Inverse Convex Optimization

Master's Thesis

Vassil Chatalbashev
Stanford University
vasco@cs.stanford.edu

Contents

Chapter 1

Introduction

Traditionally, machine learning has been concerned with univariate, unstructured problems. Whether it is predictions in a discrete output space (classification) or the continuous domain (regression), most machine learning techniques ignore the structure present in many real-world problems.

In this thesis we consider settings in which the output space has structure. Specifically, we assume that we know what constraints need to be satisfied in our predictions, and seek to learn the parameters of an objective function, which will produce a desired output. To do that we view the prediction task as an optimization problem, with a convex objective and convex constraints. That gives rise to an efficient convex optimization formulation of the learning problem. We consider both discrete and continuous outputs, and show experimentally that our *Inverse Convex Optimization (ICO)* method is able to exploit structure and perform better than traditional approaches.

Many important real-world problems, in a variety of domains, can be viewed as *convex optimization* problems. Tasks such as resource allocation, portfolio optimization, graphical model inference, scheduling, matching problems, and many others have natural formulations convex optimization problems. Non-convex problems, or even NP-hard problems, such as the travelling salesman, clustering, maximum-cut, and others have convex optimization relaxations which are useful in finding near-optimal solutions. Furthermore, recent developments in convex optimization [5] have led to numerical optimization algorithms which are not only applicable to a wider range of problems than was previously possible, but are efficient, reliable and enjoy strong theoretic guarantees. Such algorithms have made convex optimization problems even more attractive.

In many cases, however, the formulation of a particular task as an optimization problem is far from obvious. While one can usually narrow down their formulation to a parametric family, such as a linear or quadratic function, the quality of the solution depends critically on the choice of weights which define the optimization objective. Constructing and tuning the weights for a problem is a difficult and time-consuming process. This process is often done manually, with people tuning the objective weights until their optimal solution satisfies some goodness criteria.

In such settings one often has prototypical examples of a "goal" solution: an input $\mathbf{x}$ and output $\mathbf{y}$, such that the convex optimization formulation should produce the output $\mathbf{y}$, or something close to it when its input is $\mathbf{x}$. In those cases the general-purpose algorithm proposed here, *Inverse Convex Optimization*, can be used to learn a set of parameters that would make the output $\mathbf{y}$ optimal. When the input $\mathbf{x}$ captures important properties of the setting in which the optimization problem is applied, the parameters our method learns can generalize to similar settings, where the output should exhibit similar dependence on the input.

We take a discriminative maximum-margin approach to the task of Inverse Convex Optimization, and end up with efficient formulations for both combinatorial and continuous problems. The maximum-margin framework is a natural extension and generalization of many of the ideas in [24], [23], [25]. The optimization problems we define are convex and thus solvable efficiently and reliably using a variety of methods. Furthermore, we gain significant advantage by directly representing the prediction problem as a convex optimization problem, and modeling the constraints that must be present in its output space. For combinatorial problems we show a concrete application of our framework in associative Markov networks, and for continuous problems we provide experimental results on the task of inverse portfolio optimization. In both cases we show that our structured learning approach outperforms traditional techniques which ignore structure.

Chapter 2

Inverse Convex Optimization

2.1 Introduction

A variety of practical computational problems can be viewed as combinatorial optimization problems, including weighted bipartite matching, data clustering, and travelling salesman problems.For example, consider the task of clustering data points into coherent subsets. Given a set of weights denoting the similarity of different pairs of data points, we can formulate the clustering task as an optimization problem involving these weights. As another example, the task of assigning reviewers to papers can be formulated as a bipartite matching problem, given a set of weights denoting the “expertise” of each reviewer for each paper.

Often, however, the formulation of a particular task as an optimization problem is difficult. In both of the examples above, the quality of the solution found depends critically on the choice of weights. Constructing and tuning the weights for a problem is a difficult and time-consuming process. This process is typically done manually, where people tune the weights to try and make their algorithm return some desired “goal” solution. In this paper, we propose a general-purpose algorithm, *inverse convex optimization (ICO)*, that precisely substitutes for the manual process, by automatically training the weights of an optimization algorithm to fit a desired target output. The input to our algorithm is a set of problem instances $\{\mathcal{I}\}$, each with its target solution $\hat{\mathbf{y}}[\mathcal{I}]$. The algorithm then learns a set of parameters such that, when optimizing each instance $\mathcal{I}$, the optimal solution is $\hat{\mathbf{y}}[\mathcal{I}]$, with maximum confidence.

The general framework we propose can be applied to learn the parameters of a wide range of convex optimization problems in polynomial time, including bipartite matching, max flow, and finding the most probable assignment for any graphical model of bounded tree width, or for any probabilistic context-free grammar. In addition, many NP-hard problems, including many relaxed integer programming formulations for combinatorial optimization problems, such as clustering and travelling salesman, can be solved approximately in polynomial time by convex optimization. In such cases, our framework approximately learns the parameters of these approximation algorithms. In general, our framework effectively optimizes the weights for the algorithm that is used ultimately to perform the task. Thus, it has the effect of guiding the algorithm actually used towards the desired behavior.

2.2 Convex optimization problems

Many important problems can be solved by maximizing a concave function over a convex set. Consider for example the problem of assigning reviewers to papers at a conference, as described in Sec. 2.1. More specifically, suppose we would like to have m reviewers per paper, and that each reviewer be assigned at most k papers. For each paper and reviewer, we have an *affinity metric* c_{ij} indicating the qualification level of reviewer i for evaluating paper j. Our objective is to find an assignment for reviewers to papers that maximizes the total affinity.

This task is an instance of the general *bipartite matching problem*, which is most naturally formulated as an *integer program (IP)*, where the variables, $y_{ij} \in \{0, 1\}$, take value 1 if reviewer i is assigned to paper j, and 0 otherwise. Unfortunately, IP problems are, in general, NP-hard. In our setting, we can *relax* the integrality constraint by restricting each y_{ij} to the interval $[0, 1]$. This relaxation leads to a *linear programming (LP)* formulation:

$$\begin{aligned} \max \quad & \sum_{i,j} c_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{i} y_{ij} = m, \;\; \sum_{j} y_{ij} \le k, \qquad y_{ij} \in [0, 1]. \end{aligned} \tag{2.1}$$

Importantly, despite the relaxation, this LP always leads to integral assignments[19] to the variables y_{ij}, allowing us to solve the matching problem optimally, in polynomial time.

Bipartite matching is an example of a combinatorial optimization problem whose linear relaxation leads to an exact convex optimization formulation that can be solved efficiently. There is also a multitude of practical problems that are most naturally formulated as continuous convex optimization formulations. For example, consider the classical problem of Markowitz portfolio optimization [16]. Assume we have a set of N assets, whose returns (the relative increases in asset value) are uknown random quantities. We can estimate the mean return r_i of each stock, as well as the overall covariance of all returns $\mathbf{Q}$ (where $\mathbf{Q}_{i,j}$ is the covariance of the rerturns of assets i and j.) Intuitively, an investor would like to find an allocation of their wealth to these assets, which maximizes the expected return while keeping risk low. For an investor with a utility function of the form $u(x) = 1 - \exp(-\kappa x)$, where κ is a risk-aversion constant, it can be shown (Sec. 4.3) that the optimal expected utility portfolio can be found by solving the following quadratic program with linear constraints:

$$\begin{aligned} \max \quad & \sum_{i} r_i y_i - \frac{\kappa}{2} \sum_{i,j} y_i y_j \mathbf{Q}_{i,j} \\ \text{s.t.} \quad & \sum_{i} y_i = 1, \;\; y_i \ge 0 \end{aligned} \tag{2.2}$$

Above, y_i are the fractions of the portfolio allocated to each stock, and the positivity constraint represents the assumption that no short-selling is allowed (i.e., we cannot sell assets we do not own).

More generally, we consider optimization problems over a vector of variables $\mathbf{y}$ that maximize a concave function $f(\mathbf{y})$ over a convex set $\mathcal{Y}$. This convex set is often best described by inequality constraints $\mathbf{g}(\mathbf{y}) \preceq 0$, where $\preceq$ generalizes the inequality constraints used in LPs to the more general matrix inequalities, used, for example, in *semi-definite programming (SDP)*. We can now present the general convex optimization problem:

$$\begin{aligned} \max \quad & f(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{y}) \preceq 0, \end{aligned} \tag{2.3}$$

where we assume that this problem is convex, strictly feasible and bounded [5]. This formulation represents a very general class of problems that can be solved in polynomial time, including bipartite matching, min-cut, shortest path, and max-flow. In addition, such convex formulations allow us to obtain approximate solutions for many NP-hard problems, including many relaxed integer programming formulations for combinatorial optimization problems, such as clustering and travelling salesman. (See [19, 5, 1] for more details on optimization.)

2.3 ICO Learning

In the reviewer assignment problem described in the previous section, we assumed a known compatibility function c_{ij} between reviewers and papers. In general, this function is not known a priori. In such cases, the user typically picks a parametric form for this function. In the assignment problem, for example, we can pick *features* that depend on the words on the paper and on those on the reviewer's home page. The set of weights of this parametric function are then "tuned" until the desired output is obtained. Similarly, in portfolio optimization, we could parameterize the means and covariances of asset returns by a set of features. For example the features could include the observed means and covariances based on information in different past periods. If we are then presented with examples of portfolio allocations to a set of stocks, we can learn the weights associated with the features, such the solutions of the portfolio optimization problem (Eq. (2.2)) will produce the sample portfolio allocations.

In this section, we present an algorithm that automatically learns the weights of such objective function from examples of successfully optimized problems, e.g., the reviewer assignment in previous conferences, or previous portfolio allocations of an investor.

2.3.1 Definition

More formally, we consider optimization tasks that take a problem instance $\mathcal{I}$, associated with some set of features $\mathbf{X}[\mathcal{I}]$, and need to produce an output $\hat{\mathbf{y}}$. The objective function for this problem $f(\mathbf{y})$ is defined in terms of the features $\mathbf{X}[\mathcal{I}]$, and parameterized by a set of weights $\mathbf{w}$. Specifically, we define our parameterized objective function $f_{\mathbf{w}}$ as a linear combination of subfunctions weighted by the parameters $\mathbf{w}$:

$$f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) = \sum_i w_i f_i(\mathbf{X}, \mathbf{y}).$$

In order for $f_{\mathbf{w}}$ to be concave, we need, for each i, either that f_i is concave in $\mathbf{y}$ and $w_i \geq 0$, or that f_i is linear in $\mathbf{y}$ (in which case w_i can be unconstrained). Often, the objective is simply a linear function in $\mathbf{y}$ and $\mathbf{w}$: $f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) = \mathbf{w}^\top \mathbf{X}\mathbf{y}$. In our matching example, $f_{\mathbf{w}}$ decomposes linearly in y_{ij}, and the c_{ij} is defined by a linear combination of features of the papers and reviewers. In the portfolio optimization example, the covariance matrix of asset returns can be represented as a positive linear combination of a set of feature matrices, whereas the means of the returns will be a linear combination of a set of feature vectors.

Using this formalism, we define the *inverse convex optimization (ICO) problem* that seeks to learn weights $\mathbf{w}$ such that $\hat{\mathbf{y}}$ optimizes $f_{\mathbf{w}}(\mathbf{X}, \mathbf{y})$. Specifically, the input to our learning algorithm is a set of pairs $(\mathbf{X}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{I}])$, where $\mathcal{I}$ is an instance of our optimization task associated with a set of features $\mathbf{X}[\mathcal{I}]$, and $\hat{\mathbf{y}}[\mathcal{I}]$ is the target output (solution) for this instance over the convex set $\mathbf{g}(\mathbf{X}, \mathbf{y}) \preceq 0$. (For simplicity of notation, we omit the instance $\mathcal{I}$, when it is clear from the context.) Our goal is to find a setting $\mathbf{w}^*$ to the parameters such that maximizing $f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y})$ over the convex set will produce the desired target output $\hat{\mathbf{y}}[\mathcal{I}]$:

$$\hat{\mathbf{y}} \in \underset{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y}) \preceq 0}{\operatorname{argmax}} f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y}).$$

The above requirement can be seen as a set of strict inequalities:

$$f_{\mathbf{w}^*}(\mathbf{X}, \hat{\mathbf{y}}) > f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y}), \quad \forall \mathbf{y} : \mathbf{y} \neq \hat{\mathbf{y}},\ \mathbf{g}(\mathbf{X}, \mathbf{y}) \preceq 0$$

In other words, we require that the value of the convex optimization problem objective at $\hat{\mathbf{y}}$ is strictly greater than the value of all other feasible $\mathbf{y}$. Unfortunately, dealing with a potentially infinite set of strict inequalities is not easy. We define a *loss function* $\ell(\mathbf{y}, \hat{\mathbf{y}})$, which is strictly positive, except when $\mathbf{y} = \hat{\mathbf{y}}$ in which case it equals 0.

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = \begin{cases} 0 & \text{if } \mathbf{y} = \hat{\mathbf{y}} \\ > 0 & \text{otherwise.} \end{cases}$$

We can then transform the strict inequalities to a more manageable form:

$$f_{\mathbf{w}^*}(\mathbf{X}, \hat{\mathbf{y}}) \geq f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y}) + \ell(\mathbf{y}, \hat{\mathbf{y}}), \quad \forall \mathbf{y} :\ \mathbf{g}(\mathbf{X}, \mathbf{y}) \preceq 0$$

In their current form, the above set of constraints relies on the loss function to achieve the strict inequality of all $\mathbf{y}$ which are different from the desired output $\hat{\mathbf{y}}$. Thus, a set of weights $\mathbf{w}^*$ which satisfies the potentially infinite set of inequalities will be guaranteed to produce the output $\hat{\mathbf{y}}$. Note that the above set of constraints is equivalent to:

$$f_{\mathbf{w}^*}(\mathbf{X}, \hat{\mathbf{y}}) \geq \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y})} f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y}) + \ell(\mathbf{y}, \hat{\mathbf{y}})$$

In order to formulate a concrete learning problem, we adopt a maximum margin framework, where we maximize the "confidence margin" in the target output relative to other possible outputs. Furthermore, we might often want to ensure that an output $\mathbf{y}'$ that is very different for $\hat{\mathbf{y}}$ leads to an objective value $f_{\mathbf{w}^*}(\mathbf{X}, \mathbf{y}')$ that is significantly lower than $f_{\mathbf{w}^*}(\mathbf{X}, \hat{\mathbf{y}})$. We can achieve this by requiring that the loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ correspond to the "distance" between $\mathbf{y}$ and $\hat{\mathbf{y}}$. The margin of confidence γ should then grow as $\mathbf{y}$ becomes more distant from $\hat{\mathbf{y}}$, which we formulate as $\gamma\ell(\mathbf{y}, \hat{\mathbf{y}})$. In the maximum margin Markov networks framework of Taskar *et al.* [24], for example, $\ell(\mathbf{y}, \hat{\mathbf{y}})$ corresponds to the Hamming distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$, and the margin must grow with the number of entries $\mathbf{y}$ assigns differently from $\hat{\mathbf{y}}$. As will be discussed later, we require that the loss function is concave in y, to ensure the convexity of the formulation.

We can now define our complete ICO formulation:

$$\begin{aligned} \max_{\gamma, \mathbf{w}:\ \|\mathbf{w}\| \leq 1, \mathbf{w} \geq 0} \quad & \gamma \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) \geq \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X}, \mathbf{y}) \preceq 0} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \gamma\, \ell(\mathbf{y}, \hat{\mathbf{y}}). \end{aligned} \tag{2.4}$$

Note that the non-negativity constraint on $\mathbf{w}$ may sometimes not be present or only present for a subset of the weights.

2.3.2 Duality

By solving the ICO, we obtain weights $\mathbf{w}^*$ that guarantee that $\hat{\mathbf{y}}$ is the optimal solution to the objective function with maximum confidence γ^*. This ICO formulation is convex, and can be solved by existing interior point methods [5]. However, the maximization term in the constraints of the ICO formulation is usually not differentiable, which hinders the use of standard convex optimization packages to solve this problem. Efficient optimization is sometimes possible, especially using cutting-plane methods ([4], [1], Sec. 4.3.4), but the efficiency depends largely on particular features of the maximization problem which are not generally present. In order to derive a general formulation, we use techniques from duality theory, to convert the ICO problem in a standard form.

First, consider the Lagrangian of the convex maximization problem present in the constraints of the above formulation:

$$\mathcal{L}_{\mathbf{w},\gamma}(\mathbf{X}, \mathbf{y}, \mathbf{z}) = f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \gamma\ell(\mathbf{y}, \hat{\mathbf{y}}) - \mathbf{z}\, \mathbf{g}(\mathbf{X}, \mathbf{y}), \quad \text{s.t.} \;\; \mathbf{z} \succeq 0. \tag{2.5}$$

We can now define the dual problem in terms of the solution to the primal problem:

$$\mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X}, \mathbf{z}) = \max_{\mathbf{y}} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \gamma\ell(\mathbf{y}, \hat{\mathbf{y}}) - \mathbf{z}\, \mathbf{g}(\mathbf{X}, \mathbf{y}), \quad \text{s.t.} \;\; \mathbf{z} \succeq 0. \tag{2.6}$$

As the primal problem is convex, feasible, and bounded, there is no duality gap [5], and thus:

$$\min_{\mathbf{z}: \mathbf{z} \succeq 0} \mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X}, \mathbf{z}) = \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X}, \mathbf{y}) \preceq 0} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \gamma\, \ell(\mathbf{y}, \hat{\mathbf{y}}).$$

Furthermore, from weak duality we know that any feasible dual variables $\mathbf{z}$ will produce a dual objective value $\mathbf{h}_{\mathbf{w},\gamma}$ which upper bounds the primal constraint objective $f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \gamma\ell(\mathbf{y}, \hat{\mathbf{y}})$. Therefore we can use the dual formulation in place of the primal one in the constraints for the ICO in Eq. (2.4). In other words, we can replace the maximization problem in the constraint with simply the value of its dual objective, along with feasibility constraints for the dual variables.

$$\begin{aligned} \max_{\gamma,\mathbf{w}:\ \|\mathbf{w}\|\le 1, \mathbf{w}\ge 0} \quad & \gamma \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}}) \ge \mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X},\mathbf{z}), \qquad \mathbf{z} \succeq 0. \end{aligned} \tag{2.7}$$

This new formulation is equivalent to the original one since we established that $\mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X},\mathbf{z}) \ge \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y})\preceq 0} f_{\mathbf{w}}(\mathbf{X},\mathbf{y}) + \gamma\,\ell(\mathbf{y},\hat{\mathbf{y}})$. Because the only constraints on the dual variables is their feasibility, the optimization of Eq. (2.7) can modify their values until they produce the optimal dual value, which is possible because there is no duality gap. If at the optimum of Eq. (2.7) the dual variables of the constraint are not optimal for $\mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X},\mathbf{z})$, then note that the solution is still feasible in Eq. (2.4), and hence optimal. This is formalized below:

Theorem 2.3.1 *The optimization problem Eq. (2.7) is jointly convex in γ, $\mathbf{w}$ and $\mathbf{z}$. Furthermore, let $(\gamma^*, \mathbf{w}^*, \mathbf{z}^*)$ be an optimal solution to this problem, then $(\gamma^*, \mathbf{w}^*)$ is an optimal solution to the ICO formulation in Eq. (2.4).*

Proof Showing convexity is straightforward. The non-negativity and norm constraints on $\mathbf{w}$ are convex. $f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}})$ is linear in $\mathbf{w}$ by assumption, and from the convexity of $f_{\mathbf{w}}(\mathbf{X},\mathbf{y})$, it follows that $\mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X},\mathbf{z})$ is concave in z. Also the objective is linear in γ.

We want to show that if $(\gamma^*, \mathbf{w}^*, \mathbf{z}^*)$ is an optimal solution to the problem, then $(\gamma^*, \mathbf{w}^*)$ is an optimal solution to Eq. (2.4). Since $f_{\mathbf{w}}$ and ℓ are concave, and $\mathbf{g}$ is convex, from weak duality we know that:

$$\forall \mathbf{z} \succeq 0 : \mathbf{h}_{\mathbf{w},\gamma}(\mathbf{X},\mathbf{z}) \ge \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y})\preceq 0} f_{\mathbf{w}}(\mathbf{X},\mathbf{y}) + \gamma\,\ell(\mathbf{y},\hat{\mathbf{y}}).$$

In other words, for any feasible $\mathbf{z}$, the dual objective of the maximization problem in the constraint, $\mathbf{h}_{\mathbf{w},\gamma}$, will be an upper bound on the primal constraint objective . $\mathbf{z}^*$ is feasible for $\mathbf{h}_{\mathbf{w},\gamma}$ ($\mathbf{z} \succeq 0$ is one of the constraints in Eq. (2.7), therefore $(\gamma^*, \mathbf{w}^*)$ is feasible for Eq. (2.4), and hence optimal. ■

It is important to note that the dualization trick above leads to a closed-form analytic constraint, which does not involve maximization, only when the maximization w.r.t. $\mathbf{y}$ in Eq. (2.6) can be carried out in closed form. If that is not the case, then $\mathbf{h}_{\mathbf{w},\gamma}$ involves a maximization term, and the formulation Eq. (2.7) has no obvious computational advantages to the one in Eq. (2.4). Also note that even in the cases when we are able to carry out the maximization in Eq. (2.6) in closed form, there may be an additional set of constraints that are produced as a result. For example typically the maximization is done by explicitly setting a gradient to 0, which naturally leads to additional constraints on the primal variables. Examples of that can be seen in AMNs, and the Markowitz portfolio optimization application of ICO (Chapter 3, Chapter 4).

2.3.3 Separability

Often, it is not possible to guarantee that every training instance $\mathcal{I}$ can be optimized successfully by the chosen convex optimization problem. In such settings, we can add slack variables $\xi_{\mathcal{I}}$ for each instance $\mathcal{I}$ that allow some problems to be optimized with lower confidence margin. The optimization problem then balances maximizing the margin against minimizing these slacks, using a penalty constant C. Our final formulation, in addition to including slack variables, uses standard manipulations, such as the ones used in SVMs ([10]), to transform maximizing the margin to minimizing the norm of the weights:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\xi \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}}) \geq \mathbf{h}_{\mathbf{w}}(\mathbf{X},\mathbf{z}) - \xi, \qquad \mathbf{z} \succeq 0, \qquad \xi \geq 0. \end{aligned} \tag{2.8}$$

2.3.4 Multiple Training Examples

Typically, one has a set of training examples $\{(\mathbf{X}_i, \hat{\mathbf{y}}_i)\}$. To extend the above formulation, we simply reuse the dualization trick for each training instance and add a separate slack variable with non-negativity constraints.

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X}_i,\hat{\mathbf{y}}_i) \geq \mathbf{h}_{\mathbf{w}}(\mathbf{X}_i,\mathbf{z}_i) - \xi_i, \qquad \mathbf{z}_i \succeq 0, \qquad \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{2.9}$$

2.4 Conclusion

We now have a complete framework for Inverse Convex Optimization: We first defined the objective function of our task in terms of features of the training instances. We then formed the optimization problem for the ICO task, which featured a margin scaled by the loss function, as well as maximization constraints. Finally, we dualized this optimization problem, to formulate the ICO problem in Eq. (2.8). The benefit of the dualization is that it is possible to use standard solvers to learn the parameters of our optimization problem. Once we have learned the parameters, we use the features of a new instance to define the objective function of a solution algorithm for this new instance.

The framework relies only on the convexity of the optimization problem, for which we would like to learn the weights, as well as our ability to define a concave loss function which distinguishes between correct and incorrect outputs $\mathbf{y}$. Thus it has the potential of being applicable to a very wide range of convex optimization problems.

A number of existing maximum-margin algorithms can be seen as instances of the ICO framework. For example maximum-margin learning in structured output spaces, such as context free grammars [25], tree structured Markov networks [24], arbitrary topology associative Markov networks [23] follow the same framework of maximizing the margin of confidence and using duality and problem structure to efficiently formulate a convex optimization problem for the learning task.

In the next chapters we will see how this technique can be applied to both combinatorial optimization problems like MAP inference in a subclass of graphical models called associative Markov networks, and to continuous problems like Markowitz portfolio optimization.

Chapter 3

Combinatorial Inverse Convex Optimization

In this chapter we will focus on an important subclass of convex optimization problems: combinatorial optimization problems, where the output space can be represented as a set of *binary* assignments, usually under some constraints $\mathcal{Y}$.

$$\mathbf{y} \in \{0,1\}^n, \;\; \mathbf{y} \in \mathcal{Y} \tag{3.1}$$

Typically $\mathcal{Y}$ is a set of linear or semi-definite constraints, but the methods we develop will not explicitly depend on that. We will only require that the combinatorial problem can be solved exactly or approximately via a convex optimization formulation, which is true for a number of interesting problems such as mincut, matching, clustering, and others. We show a concrete example of an ICO formulation with Associative Markov Networks [23].

3.1 Loss Function

Recall the ICO formulation for convex optimization problems:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}}) \geq \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y})\preceq 0} f_{\mathbf{w}}(\mathbf{X},\mathbf{y}) + \ell(\mathbf{y},\hat{\mathbf{y}}) - \xi \end{aligned}$$

In this chapter, we have already restricted our attention to convex relaxations of combinatorial problems, but to ensure the convexity of the final ICO problem, we still need to define a concave *loss function* $\ell(\mathbf{y},\hat{\mathbf{y}})$. Specifically, the loss function must have the value 0 when $\mathbf{y} = \hat{\mathbf{y}}$, and be strictly positive when $\mathbf{y} \neq \hat{\mathbf{y}}$. Since we assumed that combinatorial problems have outputs $\mathbf{y} \in \{0,1\}$, a natural choice for a loss function is Hamming distance, or the number of different "bits" between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Such a loss function will have the additional desirable property that its value will grow as $\mathbf{y}$ moves further away from $\hat{\mathbf{y}}$. The effect of that will be that the weights we learn will have the property that the convex optimization objective they define, $f_{\mathbf{w}}(\mathbf{X},\mathbf{y})$, will tend to be much more "confident" in the true output $\hat{\mathbf{y}}$ as compared to a distant output $\mathbf{y}$, since it value will be proportionally higher.

Let $\mathbf{e}$ denote a vector of ones. We show that it can be expressed as a linear function. Consider:

$$\begin{aligned} \ell(\mathbf{y}, \hat{\mathbf{y}}) &= (\mathbf{e} - \hat{\mathbf{y}})^\top \mathbf{y} + (\mathbf{e} - \mathbf{y})^\top \hat{\mathbf{y}} \\ &= (\mathbf{e} - 2\hat{\mathbf{y}})^\top \mathbf{y} + \mathbf{e}^\top \hat{\mathbf{y}} \end{aligned} \tag{3.2}$$

First, notice that $\ell(\mathbf{y}, \hat{\mathbf{y}})$ is linear, and hence concave in $\mathbf{y}$, which was required by ICO. Assuming $\mathbf{y}$ and $\hat{\mathbf{y}}$ are both bit vectors, the first term counts the number of entries which are equal to 0 in $\hat{\mathbf{y}}$ and equal to 1 in $\mathbf{y}$ and the second term counts the number of entries which are equal to 1 in $\hat{\mathbf{y}}$ and equal to 0 in $\mathbf{y}$. With the concave Hamming loss in place, the ICO formulation for combinatorial problems is complete. We have:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) - \mathbf{e}^\top \hat{\mathbf{y}} + \xi \geq \max_{\mathbf{y}:\ \mathbf{g}(\mathbf{X},\mathbf{y})\preceq 0} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + (\mathbf{e} - 2\hat{\mathbf{y}})^\top \mathbf{y} \end{aligned}$$

In some applications it may be desirable to define the Hamming loss only on a subset of the output $\mathbf{y}$. Also, if the constraints on the output space are such that one bit determines a set of other bits (for example due to a summation constraint), that should be taken into account. We will see such an example in Associative Markov Networks (see Sec. 3.4.3)

The use of Hamming distance as the loss function seems like a natural choice when working in bit-space, but it also has advantageous theoretical properties. In [24] Taskar *et al.*show that in learning Markov Networks of fixed tree-width the use of Hamming distance to scale the margin leads to a novel bound on sample complexity.

3.2 Integer Programming and Linear Relaxations

While our framework applies to general convex relaxations of combinatorial problems, most often combinatorial problems are solved via linear relaxations. In this section we will characterize such relaxations precisely, and give conditions under which they are exact.

Many combinatorial problems can be solved via *integer programming (IP)*, which is in general NP-hard [19], but can be solved exactly or approximated closely in certain cases. Because *integer programming* is not convex, it does not fit in the ICO framework, but as we will see its relaxation to a linear program can sometimes be used instead, leading to a convex formulation. In this section we will discuss conditions, under which an integer linear program can be solved *exactly* via a linear program.

As a running example consider a classic combinatorial optimization problem: *bipartite matching*. As a concrete application consider the task of assigning reviewers to papers at a conference (Chapter 2). The problem can be formulated as an integer linear program:

$$\begin{aligned} \max \quad & \sum_{i,j} c_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_i y_{ij} = m, \quad \sum_j y_{ij} \leq k, \qquad y_{ij} \in \{0,1\}. \end{aligned} \tag{3.3}$$

Clearly the formulation in Eq. (3.3) is not convex due to the integer restriction of variables, but its linear relaxation is guaranteed to produce integral solutions [19].

$$\begin{aligned} \max \quad & \sum_{i,j} c_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_i y_{ij} = m, \quad \sum_j y_{ij} \leq k, \quad y_{ij} \geq 0, \quad y_{ij} \leq 1. \end{aligned} \tag{3.4}$$

More generally consider a linear program of the form:

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} \leq \mathbf{b}, \quad \mathbf{y} \geq 0 \end{aligned} \tag{3.5}$$

It is useful to be able to characterize when the LP in Eq. (3.5) will produce integral solutions. In general there is no complete condition for that, but there is an important subclass of LPs, for which we can show integrality. In order to do that we need to define the concept of a *totally unimodular* matrix.

Definition 3.2.1 *A matrix* $\mathbf{A}$ *is totally unimodular if the determinant of each square submatrix of* $\mathbf{A}$ *is either* 0, -1, *or* $+1$.

Theorem 3.2.2 *Let* $\mathbf{A}$ *be an integer* $m \times n$ *matrix, and let* $F = \{\mathbf{y} \mid \ \mathbf{A}\mathbf{y} \leq \mathbf{b}, \ \ \mathbf{y} \geq 0\}$ *be nonempty. Then* $\mathbf{A}$ *is totally unimodular if and only if for each integer vector* $\mathbf{b}$ *the region* F *has integer vertices.*

Proof See [19]. ▮

Since the solution of an LP lies at the vertices of the polyhedron, Theorem 3.2.2 implies that Eq. (3.5) will always produce integer solutions if $\mathbf{A}$ is totally unimodular, and $\mathbf{b}$ is integer.

Unfortunately, determining whether a matrix is totally unimodular is not always an easy task. Sometimes it is straightforward to verify (for example in network flow problems), but generally the only complete way of determining total unimodularity is to actually evaluate the determinant of all square submatrices, which is clearly intractable when the matrix is large. In fact, determining total unimodularity is known to be NP-hard [19].

In many cases, however, it is possible to show that a linear program will yield integer solutions. Going back to our example of bipartite matching, the constraint matrix in Eq. (3.4) can be shown to be totally unimodular since it is an instance of a network node-arc incidence matrix, which is known to be totally unimodular [19]). Thus, the solutions of the LP will always be the same as the solutions of the integer program in Eq. (3.3). Other important problems which yield integral linear relaxation include various flavors of network flow problems.

Sometimes a linear relaxation of an integer linear program will not produce integer solutions. In those cases the linear relaxation will have an optimal objective which is greater than that of the corresponding integer program, since its feasible set is a superset of the feasible set of the integer program. Still, we may be able to prove that the solution of the LP will be within a certain relative error from the optimal IP solution, which can give us some degree of confidence that it is a good approximation. In those cases the linear relaxation can still be used to formulate a combinatorial ICO problem, and even though it will lack guarantees, it may perform well in practice.

3.3 ICO for Combinatorial LP Problems

We can now define the complete ICO formulation for (exact or approximate) linear relaxations of combinatorial problems. Let $\mathbf{X}$ be a matrix of features (for a concrete example of what features may look like see Sec. 3.4), so that the combinatorial problem, given the weights $\mathbf{w}$ can be solved as:

$$\begin{aligned} \max \quad & \mathbf{w}^\top \mathbf{X}\mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} \le \mathbf{b}, \quad \mathbf{y} \ge 0 \end{aligned} \tag{3.6}$$

The complete ICO QP is then:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\ge 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{X}\hat{\mathbf{y}} - \mathbf{e}^\top \hat{\mathbf{y}} + \xi \ge \max_{\mathbf{y}:\ \mathbf{A}\mathbf{y}\le\mathbf{b},\ \mathbf{y}\ge 0} \mathbf{w}^\top \mathbf{X}\mathbf{y} + (\mathbf{e} - 2\hat{\mathbf{y}})^\top \mathbf{y} \end{aligned}$$

From LP Duality ([3]) we know that the dual of the LP in the constraint is:

$$\begin{aligned} \min \quad & \mathbf{b}^\top \mathbf{z} \\ \text{s.t.} \quad & \mathbf{A}^\top \mathbf{z} \ge \mathbf{X}^\top \mathbf{w} + (\mathbf{e} - 2\hat{\mathbf{y}}), \quad \mathbf{z} \ge 0 \end{aligned}$$

We can now plug in the value of the dual to obtain the final form:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\ge 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{X}\hat{\mathbf{y}} - \mathbf{e}^\top \hat{\mathbf{y}} + \xi \ge \mathbf{b}^\top \mathbf{z} \\ & \mathbf{A}^\top \mathbf{z} \ge \mathbf{X}^\top \mathbf{w} + (\mathbf{e} - 2\hat{\mathbf{y}}), \quad \mathbf{z} \ge 0 \end{aligned}$$

Assuming a polynomial sized matrix $\mathbf{A}$ the above formulation has a polynomial number of constraints, and therefore can be solved in polynomial time using standard methods ([3]).

In the cases when the LP in Eq. (3.6) produces integral solutions, the formulation is exact. In cases where it isn't the optimal value of Eq. (3.6) may be greater than the optimal

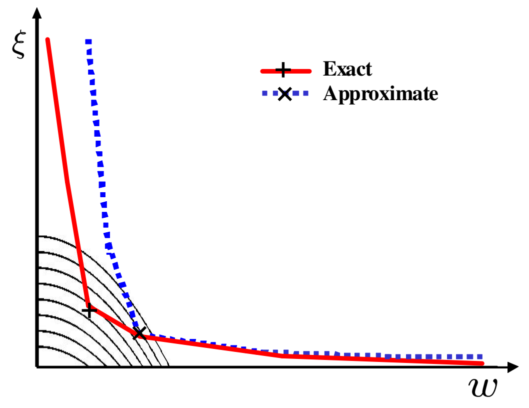

Figure 3.1: Exact and approximate constraints on the max-margin ICO formulation. The solid red line represents the constraints imposed by integer $\mathbf{y}$'s, whereas the dashed blue line represents the stronger constraints imposed by the larger set of fractional $\mathbf{y}$'s. The fractional constraints may coincide with the integer constraints in some cases, and be more stringent in others. The parabolic contours represent the value of the objective function $\frac{1}{2}\|\mathbf{w}\| + \xi$.

value of the integer program it relaxes (since its feasible set is a superset of the feasible set of the IP), and thus the constraints in Eq. (3.7) and Eq. (3.7) are potentially tighter than necessary (see Fig. 3.1). In other words, every setting of the weights $\mathbf{w}$ which produces fractionaly solutions will have tighter constraints, and the optimal solution of the LP will be better than all integer solutions, including the true one: $\hat{\mathbf{y}}$, which will inevitably result in a higher slack ξ. Because the objective of Eq. (3.7) includes a slack penalty, minimizing it should also be expected to produce $\mathbf{w}$ which tend to lead to optimal integer solutions.

3.4 Associative Markov Networks

MAP inference in associative Markov networks [23] is an example of a combinatorial problem, whose linear relaxation yields integer solutions and is thus amenable to the ICO technique. In this section we will describe the inference problem and the corresponding ICO formulation in detail. First, we briefly review Markov networks, and then we define associative Markov networks and prove important properties of the linear relaxation to the MAP inference problem.

3.4.1 Markov Networks

We restrict attention to networks over discrete variables $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, where each variable corresponds to an object we wish to classify and has K possible labels: $Y_i \in \{1, \ldots, K\}$. An assignment of values to $\mathbf{Y}$ is denoted by $\mathbf{y}$. A Markov network for $\mathbf{Y}$ defines a joint distribution over $\{1, \ldots, K\}^N$.

A Markov network is defined by an undirected graph over the nodes $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$. In general, a Markov network is a set of *cliques* $\mathcal{C}$, where each clique $c \in \mathcal{C}$ is associated with a subset Y_c of $\mathbf{Y}$. The nodes Y_i in a clique c form a fully connected subgraph (a clique) in the Markov network graph. Each clique is accompanied by a *potential* $\phi_c(Y_c)$, which associates a non-negative value with each assignment $\mathbf{y}_c$ to Y_c. The Markov network defines the probability distribution:

$$P_\phi(\mathbf{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}_c)$$

where Z is the *partition function* given by $Z = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{y}_c{}')$.

For simplicity of exposition, we focus most of our discussion on *pairwise* Markov networks. We extend our results to higher-order interactions in Sec. 3.4.2. A pairwise Markov network is simply a Markov network where all of the cliques involve either a single node or a pair of nodes. Thus, in a pairwise Markov network with edges $E = \{(ij)\}$ $(i < j)$, only nodes and edges are associated with potentials $\phi_i(Y_i)$ and $\phi_{ij}(Y_i, Y_j)$. A pairwise Markov net defines the distribution

$$P_\phi(\mathbf{y}) = \frac{1}{Z} \prod_{i=1}^{N} \phi_i(y_i) \prod_{(ij) \in E} \phi_{ij}(y_i, y_j),$$

where Z is the *partition function* given by $Z = \sum_{\mathbf{y}'} \prod_{i=1}^{N} \phi_i(y_i') \prod_{(ij) \in E} \phi_{ij}(y_i', y_j')$.

The node and edge potentials are functions of the features of the objects $\mathbf{x}_i \in \Re^{d_n}$ and features of the relationships between them $\mathbf{x}_{ij} \in \Re^{d_e}$. In hypertext classification, $\mathbf{x}_i$ might be the counts of the words of the document i, while $\mathbf{x}_{ij}$ might be the words surrounding the hyperlink(s) between documents i and j. The simplest model of dependence of the potentials on the features is a log-linear combination: $\log \phi_i(k) = \mathbf{w}_n^k \cdot \mathbf{x}_i$ and $\log \phi_{ij}(k, l) = \mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}$, where $\mathbf{w}_n^k$ and $\mathbf{w}_e^{k,l}$ are label-specific row vectors of node and edge parameters, of size d_n and d_e, respectively. Note that this formulation assumes that all of the nodes in the network share the same set of weights, and similarly all of the edges share the same weights.

We represent an assignment $\mathbf{y}$ as a set of $K \cdot N$ indicators $\{y_i^k\}$, where $y_i^k = I(y_i = k)$. With these definitions, the log of conditional probability $\log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$ is given by:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} (\mathbf{w}_n^k \cdot \mathbf{x}_i) y_i^k + \sum_{(ij) \in E} \sum_{k,l=1}^{K} (\mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}) y_i^k y_j^l - \log Z_{\mathbf{w}}(\mathbf{x}).$$

Note that the partition function $Z_{\mathbf{w}}(\mathbf{x})$ above depends on the parameters $\mathbf{w}$ and input features $\mathbf{x}$, but not on the labels y_i's.

For compactness of notation, we define the node and edge weight vectors $\mathbf{w}_n = (\mathbf{w}_n^1, \ldots, \mathbf{w}_n^K)$ and $\mathbf{w}_e = (\mathbf{w}_e^{1,1}, \ldots, \mathbf{w}_e^{K,K})$, and let $\mathbf{w} = (\mathbf{w}_n, \mathbf{w}_e)$ be a vector of all the weights, of size $d = Kd_n + K^2 d_e$. Also, we define the node and edge labels vectors, $\mathbf{y}_n = (\ldots, y_i^1, \ldots, y_i^K, \ldots)^\top$ and $\mathbf{y}_e = (\ldots, y_{ij}^{1,1}, \ldots, y_{ij}^{K,K}, \ldots)^\top$, where $y_{ij}^{k,l} = y_i^k y_j^l$, and the vector of all labels $\mathbf{y} = (\mathbf{y}_n, \mathbf{y}_e)$ of size $L = KN + K^2|E|$. Finally, we define an appropriate $d \times L$ matrix $\mathbf{X}$ such that

$$\log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \mathbf{w}^\top \mathbf{X} \mathbf{y} - \log Z_{\mathbf{w}}(\mathbf{x}).$$

The matrix $\mathbf{X}$ contains the node feature vectors $\mathbf{x}_i$ and edge feature vectors $\mathbf{x}_{ij}$ repeated multiple times (for each label k or label pair k, l respectively), and padded with zeros appropriately.

3.4.2 MAP Inference and Associative Markov Networks

A key task in Markov networks is computing the *MAP (maximum a posteriori) assignment* — the assignment $\mathbf{y}$ that maximizes $\log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$. It is straightforward to formulate the MAP inference task as an integer linear program: The variables are the assignments to the nodes y_i^k and edges $y_{ij}^{k,l}$ which must be in the set $\{0,1\}$, and satisfy linear normalization and agreement constraints. The optimization criterion is simply the linear function $\mathbf{w}^\top \mathbf{X}\mathbf{y}$, which acorresponds to the log of the unnormalized probability of the assignment $\mathbf{y}$.

In certain cases, we can take this integer program, and approximate it as a linear program by relaxing the integrality constraints on y_i^k, with appropriate constraints. For example, Wainwright *et al.*[29] provides a natural formulation of this form that is guaranteed to produce integral solutions for triangulated graphs.

We now describe one important subclass of problems for which the above relaxation is particularly useful. These networks, which we call *associative Markov networks (AMNs)*, encode situations where related variables tend to have the same value.

Associative interactions arise naturally in the context of image processing, where nearby pixels are likely to have the same label [2, 7]. In this setting, a common approach is to use a *generalized Potts model* [22], which penalizes assignments that do not have the same label across the edge: $\phi_{ij}(k,l) = \lambda_{ij},\ \forall k \neq l$ and $\phi_{ij}(k,k) = 1$, where $\lambda_{ij} \leq 1$.

For binary-valued Potts models, Greig *et al.* [13] show that the MAP problem can be formulated as a min-cut in an appropriately constructed graph. Thus, the MAP problem can be solved exactly for this class of models in polynomial time. For $K > 2$, the MAP problem is NP-hard, but a procedure based on a relaxed linear program guarantees a factor 2 approximation of the optimal solution [7, 14]. Kleinberg and Tardos [14] extend the multi-class Potts model to have more general edge potentials, under the constraints that negative log potentials $-\log \phi_{ij}(k,l)$ form a metric on the set of labels. They also provide a solution based on a relaxed LP that has certain approximation guarantees.

More recently, Kolmogorov and Zabih [15] showed how to optimize energy functions containing binary and ternary interactions using graph cuts, as long as the parameters satisfy a certain regularity condition. Our definition of associative potentials below also satisfies the Kolmogorov and Zabih regularity condition for $K = 2$. However, the structure of our potentials is simpler to describe and extend for the multi-class case. We use a linear programming formulation (instead of min-cut) for the MAP inference, which allows us to use the maximum margin estimation framework, as described below. Note however, that we can also use min-cut to perform exact inference on the learned models for $K = 2$ and also in approximate inference for $K > 2$ as in Boykov *et al.* [8]. Appendix A discusses the issue in detail.

Our associative potentials extend the Potts model in several ways. Importantly, AMNs allow different labels to have different attraction strength: $\phi_{ij}(k,k) = \lambda_{ij}^k$, where $\lambda_{ij}^k \geq 1$, and $\phi_{ij}(k,l) = 1,\ \forall k \neq l$. This additional flexibility is important in many domains, as different labels can have very diverse affinities. For example, foreground pixels tend to have locally coherent values while background is much more varied.

The linear programming relaxation of the MAP problem for these networks can be written as:

$$\max \sum_{i=1}^{N}\sum_{k=1}^{K}(\mathbf{w}_n^k \cdot \mathbf{x}_i)y_i^k + \sum_{(ij)\in E}\sum_{k=1}^{K}(\mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij})y_{ij}^k \tag{3.7}$$
$$\text{s.t. } y_i^k \geq 0, \quad \forall i,k; \qquad \sum_k y_i^k = 1, \quad \forall i;$$
$$y_{ij}^k \leq y_i^k, \qquad y_{ij}^k \leq y_j^k, \quad \forall (ij) \in E, k.$$

Note that we substitute the constraint $y_{ij}^k = y_i^k \wedge y_j^k$ by two linear constraints $y_{ij}^k \leq y_i^k$ and $y_{ij}^k \leq y_j^k$. This works because the coefficient $\mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij}$ is non-negative and we are maximizing the objective function. Hence,at the optimum $y_{ij}^k = \min(y_i^k, y_j^k)$, which is equivalent to $y_{ij}^k = y_i^k \wedge y_j^k$.

In a second important extension, AMNs admit non-pairwise interactions between variables, with potentials over cliques involving m variables $\phi(y_{i1}, \ldots, y_{im})$. In this case, the clique potentials are constrained to have the same type of structure as the edge potentials: There are K parameters $\phi(k, \ldots, k) = \lambda_{ij}^k$ and the rest of the entries are set to 1. In particular, using this additional expressive power, AMNs allow us to encode the pattern of (soft) transitivity present in many domains. For example, consider the problem of predicting whether two proteins interact [27]; this probability may increase if they *both* interact with another protein. This type of transitivity could be modeled by a ternary clique that has high λ for the assignment with all interactions present.

We can write a linear program for the MAP problem similar to Eq. (3.7), where we have a variable y_c^k for each clique c and for each label k, which represents the event that all nodes in the clique c have label k:

$$\max \sum_{i=1}^{N}\sum_{k=1}^{K}(\mathbf{w}_n^k \cdot \mathbf{x}_i)y_i^k + \sum_{c\in\mathcal{C}}\sum_{k=1}^{K}(\mathbf{w}_c^k \cdot \mathbf{x}_c)y_c^k \tag{3.8}$$
$$\text{s.t. } y_i^k \geq 0, \quad \forall i,k; \qquad \sum_k y_i^k = 1, \quad \forall i;$$
$$y_c^k \leq y_i^k, \qquad \forall c \in \mathcal{C},\ i \in c, k.$$

We will now prove important properties of the LP in Eq. (3.8).

Integrality of Binary AMN

It can be shown that in the binary case, the relaxed linear programs Eq. (3.7) and Eq. (3.8) are guaranteed to produce an integer solution when a unique solution exists.

In the subsequent discussion define, $\theta_c^k = \mathbf{w}_c^k \cdot \mathbf{x}_c$, and $\theta_i^k = \mathbf{w}_i^k \cdot \mathbf{x}_i$.

Theorem 3.4.1 *If $K = 2$, for any objective $\mathbf{w}^\top \mathbf{X}$, the linear programs in Eq. (3.7) and Eq. (3.8) have an integral optimal solution.*

Proof First define, $\theta_c^k = \mathbf{w}_c^k \cdot \mathbf{x}_c$, and $\theta_i^k = \mathbf{w}_i^k \cdot \mathbf{x}_i$. Consider any fractional, feasible $\mathbf{y}$. We show that we can construct a new feasible assignment $\mathbf{z}$ which increases the objective (or leaves it unchanged) and furthermore has fewer fractional entries.

Since $\theta_c^k \geq 0$ (because $\mathbf{w}_c^k \geq 0, \mathbf{x}_c^k \geq 0$), we can assume that $y_c^k = \min_{i \in c} y_i^k$; otherwise we could increase the objective by increasing y_c^k. We construct an assignment $\mathbf{z}$ from $\mathbf{y}$ by leaving integral values unchanged and uniformly shifting fractional values by λ:

$$\begin{array}{c} z_i^1 = y_i^1 - \lambda I(0 < y_i^1 < 1), \;\; z_i^2 = y_i^2 + \lambda I(0 < y_i^2 < 1), \\ z_c^1 = y_c^1 - \lambda I(0 < y_c^1 < 1), \;\; z_c^2 = y_c^2 + \lambda I(0 < y_c^2 < 1), \end{array}$$

where $I(\cdot)$ is an indicator function.

Now consider $\lambda^k = \min_{i:y_i^k > 0} y_i^k$. Note that if $\lambda = \lambda^1$ or $\lambda = -\lambda^2$, $\mathbf{z}$ will have at least one more integral z_i^k than $\mathbf{y}$. Thus if we can show that the update results in a feasible and better scoring assignment, we can apply it repeatedly to get an optimal integer solution. To show that $\mathbf{z}$ is feasible, we need $z_i^1 + z_i^2 = 1$, $z_i^k \geq 0$ and $z_c^k = \min_{i \in c} z_i^k$.

First, we show that $z_i^1 + z_i^2 = 1$.

$$\begin{aligned} z_i^1 + z_i^2 &= y_i^1 - \lambda I(0 < y_i^1 < 1) + y_i^2 + \lambda I(0 < y_i^2 < 1) \\ &= y_i^1 + y_i^2 = 1. \end{aligned}$$

Above we used the fact that if y_i^1 is fractional, so is y_i^2, since $y_i^1 + y_i^2 = 1$.

To show that $z_i^k \geq 0$, we prove $\min_i z_i^k = 0$.

$$\begin{aligned} \min_i z_i^k &= \min_i \left[y_i^k - (\min_{i:y_i^k > 0} y_i^k) I(0 < y_i^k < 1) \right] \\ &= \min \left(\min_i y_i^k, \min_{i:y_i^k > 0} \left[y_i^k - \min_{i:y_i^k > 0} y_i^k \right] \right) = 0. \end{aligned}$$

Lastly, we show $z_c^k = \min_{i \in c} z_i^k$.

$$\begin{aligned} z_c^1 &= y_c^1 - \lambda I(0 < y_c^1 < 1) \\ &= (\min_{i \in c} y_i^1) - \lambda I(0 < \min_{i \in c} y_i^1 < 1) = \min_{i \in c} z_i^1; \\ z_c^2 &= y_c^2 + \lambda I(0 < y_c^1 < 1) \\ &= (\min_{i \in c} y_i^2) + \lambda I(0 < \min_{i \in c} y_i^2 < 1) = \min_{i \in c} z_i^2. \end{aligned}$$

We have established that the new $\mathbf{z}$ are feasible, and it remains to show that we can improve the objective. We can show that the change in the objective is always λD for some constant D that depends only on $\mathbf{y}$ and θ. This implies that one of the two cases, $\lambda = \lambda^1$ or $\lambda = -\lambda^2$, will necessarily increase the objective (or leave it unchanged). The change in the objective is:

$$\begin{aligned} & \sum_{i=1}^{N} \sum_{k=1,2} \theta_i^k (z_i^k - y_i^k) + \sum_{c \in \mathcal{C}} \sum_{k=1,2} \theta_c^k (z_c^k - y_c^k) \\ &= \lambda \left[\sum_{i=1}^{N} (D_i^1 - D_i^2) + \sum_{c \in \mathcal{C}} (D_c^1 - D_c^2) \right] = \lambda D \\ D_i^k &= \theta_i^k I(0 < y_i^k < 1), \quad D_c^k = \theta_c^k I(0 < y_c^k < 1). \end{aligned}$$

Hence the new assignment $\mathbf{z}$ is feasible, does not decrease the objective function, and has strictly fewer fractional entries. ▮

This result states that the MAP problem in binary AMNs is tractable, regardless of network topology or clique size.

Multi-class AMNs

In the non-binary case ($K > 2$), these LPs can produce fractional solutions. We would like to formally characterize how far the fractional solutions are from a set of legal integer ones in terms of the value of the objective function.

For $K > 2$, we use the randomized rounding procedure of Kleinberg and Tardos [14] to produce an integer solution for the linear relaxation, losing at most a factor of $m = \max_{c \in \mathcal{C}} |c|$ in the objective function. The basic idea of the rounding procedure is to treat y_i^k as probabilities and assign labels according to these probabilities in phases. In each phase, we pick a label k, uniformly at random, and a threshold $\alpha \in [0,1]$ uniformly at random. For each node i which has not yet been assigned a label, we assign the label k if $y_i^k \geq \alpha$. The procedure terminates when all nodes have been assigned a label. Our analysis closely follows that of Kleinberg and Tardos [14].

Lemma 3.4.2 *The probability that a node i is assigned label k by the randomized procedure is y_i^k.*

Proof The probability that an unassigned node is assigned label k during one phase is $\frac{1}{K} y_i^k$, which is proportional to y_i^k. By symmetry, the probability that a node is assigned label k over all phases is exactly y_i^k. ▮

Lemma 3.4.3 *The probability that all nodes in a clique c are assigned label k by the procedure is at least $\frac{1}{|c|} y_c^k$.*

Proof For a single phase, the probability that all nodes in a clique c are assigned label k if none of the nodes were previously assigned is $\frac{1}{K} \min_{i \in c} y_i^k = \frac{1}{K} y_c^k$. The probability that *at least one* of the nodes will be assigned label k in a phase is $\frac{1}{K}(\max_{i \in c} y_i^k)$. The probability that *none* of the nodes in the clique will be assigned *any* label in one phase is $1 - \frac{1}{K} \sum_{k=1}^{K} \max_{i \in c} y_i^k$.

Nodes in the clique c will be assigned label k by the procedure if they are assigned label k in one phase. (They can also be assigned label k as a result of several phases, but we can ignore this possibility for the purposes of the lower bound.) The probability that all the nodes in c will be assigned label k by the procedure in a single phase is:

$$\begin{aligned} \sum_{j=1}^{\infty} \frac{1}{K} y_c^k \left(1 - \frac{1}{K} \sum_{k=1}^{K} \max_{i \in c} y_i^k \right)^{j-1} &= \frac{y_c^k}{\sum_{k=1}^{K} \max_{i \in c} y_i^k} \\ \geq \frac{y_c^k}{\sum_{k=1}^{K} \sum_{i \in c} y_i^k} &= \frac{y_c^k}{\sum_{i \in c} \sum_{k=1}^{K} y_i^k} = \frac{y_c^k}{|c|}. \end{aligned}$$

Above, we first used the fact that for $d < 1$, $\sum_{i=0}^{\infty} d^i = \frac{1}{1-d}$, and then upper-bounded the max of the set of positive y_i^k's by their sum. ∎

Theorem 3.4.4 *The expected cost of the assignment found by the randomized procedure given a solution* $\mathbf{y}$ *to the linear relaxation of the integer program in Eq. (A.1) is at least* $\sum_{i=1}^{N} \sum_{k=1}^{K} \theta_i^k y_i^k + \sum_{c \in \mathcal{C}} \frac{1}{|c|} \sum_{k=1}^{K} \theta_c^k y_c^k$.

Proof This is immediate from the previous two lemmas.

The only difference between the expected cost of the rounded solution and the (non-integer) optimal solution is the $\frac{1}{|c|}$ factor in the second term. By picking $m = \max_{c \in \mathcal{C}} |c|$, we have that the rounded solution is at most m times worse than the optimal solution produced by the LP-relaxation of Eq. (A.1). ∎

Note that the approximation factor of m applies, in fact, only to the clique potentials. Thus, if we compare the log-probability of the optimal MAP solution and the log-probability of the assignment produced by this randomized rounding procedure, the terms corresponding to the log-partition-function and the node potentials are identical. We obtain an additive error (in log-probability space) only for the clique potentials. As node potentials are often larger in magnitude than clique potentials, the fact that we incur no loss proportional to node potentials is likely to lead to smaller errors in practice. Along similar lines, we note that the constant factor approximation is smaller for smaller cliques; again, we observe, the potentials associated with large cliques are typically smaller in magnitude, reducing further the actual error in practice.

Deterministic Algorithm

We can also derandomize the randomized procedure presented in the previous section to get a deterministic algorithm with the same approximation guarantees. We use the method of conditional probabilities, similar in spirit to the approach of Kleinberg and Tardos [14].

Consider a phase of the randomized procedure. Let a denote the selected random label, and N_a denote the set of nodes assigned to the label. We define the following quantities:

- $V_0 = \sum_{i \in N_a} \theta_i^a y_i^a$: The contribution to the objective function of the assigned nodes in the phase.
- $E_0 = \sum_{c \in \mathcal{C} : i \in c \Rightarrow i \in N_a} \theta_c^a y_c^a$: The contribution to the objective function of all cliques for which each node was assigned.
- $\bar{V}_{LP} = \sum_{i \notin N_a} \sum_{k=1}^{K} \theta_i^k y_i^k$: The contribution of all unassigned nodes to the objective function. This is the same as the solution to the linear relaxation of Eq. (A.1) restricted to the set of unassigned nodes after the current phase.
- $\bar{E}_{LP} = \sum_{c \in \mathcal{C} : i \in c \Rightarrow i \notin N_a} \sum_{k=1}^{K} \theta_c^k y_c^k$: The contribution to the objective function of all cliques containing only unassigned nodes. Again, this is the same as the solution to the linear relaxation of Eq. (A.1) restricted to those cliques.

We now construct a deterministic rounding algorithm. At each step of the deterministic algorithm we select a label a and a threshold $\alpha \in [0,1]$ which lead to a non-empty set of assigned nodes N_a (where assignment is performed like in the randomized algorithm), and also maximize the quantity: $V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP}$, where $m = \max_{c\in\mathcal{C}} |c|$. We repeat until all nodes are assigned. Note that re-solving the linear relaxation of Eq. (A.1) is not required, and we simply re-use the original solution at each step. As Kleinberg and Tardos [14] note, re-solving the linear program at each step could lead to a better solution, but that is not required to get the approximation bound we are interested in.

Now let $V_{LP} = \sum_{i=1}^{N}\sum_{k=1}^{K}\theta_i^k y_i^k$ and $E_{LP} = \sum_{c\in\mathcal{C}}\sum_{k=1}^{K}\theta_c^k y_c^k$. Thus $V_{LP} + E_{LP}$ is the objective value of the linear relaxation of Eq. (A.1) for the solution $\mathbf{y}$. We first show that a phase of the above deterministic algorithm produces $V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP}$ at least equal to $V_{LP} + \frac{1}{m}E_{LP}$ in expectation.

Lemma 3.4.5 $\mathbb{E}\left[V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP}\right] \geq V_{LP} + \frac{1}{m}E_{LP}$

Proof We consider each of the above quantities in turn.

In proving Lemma 3.4.2 we saw that the probability that node i is assigned label a in on ephase is $\frac{y_i^a}{K}$. Therefore:

$$\mathbb{E}[V_0] = \sum_{i=1}^{N}\sum_{k=1}^{K}\theta_i^k\frac{y_i^k}{K} = \frac{1}{K}V_{LP}.$$

Also, in proving Lemma 3.4.3 we saw that the probability that all nodes in a clique are assigned to the same label is $\frac{1}{K}y_c^k$. Therefore:

$$\mathbb{E}[E_0] = \sum_{c\in\mathcal{C}}\sum_{k=1}^{K}\theta_c^k\frac{y_c^k}{K} = \frac{1}{K}E_{LP}.$$

The probability that a node is *not* assigned by a single phase is $\frac{K-1}{K}$ since the probabilitity that it is assigned label a is $\frac{y_i^a}{K}$, and the probability that it is assigned *any* label is $\sum_{i=1}^{K}\frac{y_i^a}{K} = \frac{1}{K}$. Then we have:

$$\mathbb{E}\left[\bar{V}_{LP}\right] = \frac{K-1}{K}\sum_{i=1}^{N}\sum_{k=1}^{K}\theta_i^k y_i^k = \frac{K-1}{K}V_{LP}.$$

The probability that all nodes in a clique c are unassigned in a phase is $1 - \frac{1}{K}\max_{i\in c} y_i^k$ and is thus at least $1 - \frac{|c|}{K}$ (by the same argument as the one in the proof of Lemma 3.4.3), which is in turn at least $1 - \frac{m}{K}$, since $m = \max_{c\in\mathcal{C}}|c|$. It follows that:

$$\mathbb{E}\left[\bar{E}_{LP}\right] \geq \left(1 - \frac{m}{K}\right)\sum_{c\in\mathcal{C}}\sum_{k=1}^{K}\theta_c^k\frac{y_c^k}{K} = \left(1 - \frac{m}{K}\right)E_{LP}.$$

Now if we sum the above expectations we get $\mathbb{E}[V_0] + \mathbb{E}[E_0] + \mathbb{E}\left[\bar{V}_{LP}\right] + \frac{1}{m}\mathbb{E}\left[\bar{E}_{LP}\right] \geq V_{LP} + \frac{1}{m}E_{LP}$.
∎

Theorem 3.4.6 *Let* $\mathbf{y}$ *be a solution of the linear relaxation of Eq. (A.1), with objective value* $V_{LP} + E_{LP}$. *The deterministic rounding algorithm finds a labeling whose corresponding objective value is at least* $V_{LP} + \frac{1}{m}E_{LP}$.

Proof We first show that a phase of the deterministic algorithm can always select a label a and threshold α such that $V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP} \geq V_{LP} + \frac{1}{m}E_{LP}$. By Lemma 3.4.5 the expectation is at least $V_{LP} + \frac{1}{m}E_{LP}$, but since choices of a and α that assign no nodes contribute exactly $V_{LP} + \frac{1}{m}E_{LP}$ to the expectation, it follows that there must exist a and α for which $V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP} \geq V_{LP} + \frac{1}{m}E_{LP}$.

We can now prove by induction on the number of phases that the labeling produced by the deterministic rounding algorithm has objective value of at least $V_{LP} + \frac{1}{m}E_{LP}$. By the inductive hypothesis the objective value for the solution obtained for the unassigned nodes remaining after a phase is at least $\bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP}$. Then the objective value of the solution is at least $V_0 + E_0 + \bar{V}_{LP} + \frac{1}{m}\bar{E}_{LP}$, which is in turn at least $V_{LP} + \frac{1}{m}E_{LP}$, as we showed above. ∎

Taking $\mathbf{y} = \hat{\mathbf{y}}$ (the optimal solution) in Theorem 3.4.6 it follows that the deterministic rounding algorithm loses at most a factor m of the optimal objective value.

3.4.3 ICO Learning

With the development of the LP for MAP inference in AMNs, learning the max-margin weights is straightforward using our established ICO framework.

Loss Function

Since we would like the margin to scale with the number of mistakes in the node predictions between $\mathbf{y}$ and $\hat{\mathbf{y}}$, we will use Hamming distance on the entries in $\mathbf{y}$ corresponding to node labels: y_i^k. Furthermore, the constraint $\sum_k y_i^k = 1$ requires us to scale the Hamming distance by $\frac{1}{K}$ because each bit difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$ will automatically result in K errors due to the summation constraint. Therefore, we define our loss as:

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = N - \mathbf{y}_n^\top \hat{\mathbf{y}}_n$$

Notice that the above simply counts the number of mistakes by subtracting from the number of nodes N all the nodes for which the two labelings agree (i.e. $y_i^k = \hat{y}_i^k$.)

ICO Formulation

Let $\mathcal{Y}'$ denote the feasible space of the AMN MAP inference LP Eq. (3.7):

$$\mathcal{Y}' = \{\mathbf{y} : y_i^k \geq 0;\ \sum_k y_i^k = 1;\ y_{ij}^k \leq y_i^k;\ y_{ij}^k \leq y_j^k\}$$

Using the ICO formalism and the above loss function the max-margin QP for estimating the weights that produce the sample labeling $\hat{\mathbf{y}}$ is:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{X}\hat{\mathbf{y}} - N + \xi \geq \max_{\mathbf{y}\in\mathcal{Y}'} \mathbf{w}^\top \mathbf{X}\mathbf{y} - \hat{\mathbf{y}}_n \cdot \mathbf{y}_n; \\ & \mathbf{w}_e \geq 0. \end{aligned} \tag{3.9}$$

Recall that AMNs require that $\mathbf{w}_e^{k,k\top}\mathbf{x}_{ij} \geq 0$, which we enforce by requiring that $\mathbf{x}_{ij} \geq 0$ and $\mathbf{w}_e \geq 0$.

We now need to find the dual of the inference LP used to represent the interior max. Specifically, $\max_{\mathbf{y}\in\mathcal{Y}'} \mathbf{w}^\top \mathbf{X}\mathbf{y} - \hat{\mathbf{y}}_n \cdot \mathbf{y}_n$ is a feasible and bounded linear program in $\mathbf{y}$, with a dual given by:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} z_i \\ \text{s.t.} \quad & z_i - \sum_{(ij),(ji)\in E} z_{ij}^k \geq \mathbf{w}_n^k \cdot \mathbf{x}_i - \hat{y}_i^k, \quad \forall i, k; \\ & z_{ij}^k + z_{ji}^k \geq \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij}, \qquad z_{ij}^k, z_{ji}^k \geq 0, \quad \forall (ij) \in E, k. \end{aligned} \tag{3.10}$$

In the dual, we have a variable z_i for each normalization constraint in Eq. (3.7) and variables z_{ij}^k, z_{ji}^k for each of the inequality constraints.

Substituting this dual into Eq. (3.9), we obtain:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{X}\hat{\mathbf{y}} - N + \xi \geq \sum_{i=1}^{N} z_i; \qquad \mathbf{w}_e \geq 0; \\ & z_i - \sum_{(ij),(ji)\in E} z_{ij}^k \geq \mathbf{w}_n^k \cdot \mathbf{x}_i - \hat{y}_i^k, \quad \forall i, k; \\ & z_{ij}^k + z_{ji}^k \geq \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij}, \qquad z_{ij}^k, z_{ji}^k \geq 0, \quad \forall (ij) \in E, k. \end{aligned} \tag{3.11}$$

For $K = 2$, the LP relaxation is exact, so that Eq. (3.11) learns *exact* max-margin weights for Markov networks of *arbitrary* topology. For $K > 2$, the linear relaxation leads to a strengthening of the constraints on $\mathbf{w}$ by potentially adding constraints corresponding to fractional assignments $\mathbf{y}$. Thus, the optimal choice $\mathbf{w}, \xi$ for the original QP may no longer be feasible, leading to a different choice of weights. However, as our experiments show, these weights tend to do well in practice.

Also note that by taking the dual of the QP in Eq. (3.11) we can show that the node features are kernelizable. Unfortunately the positivity constraint on the edge parameters $\mathbf{w}_e$ makes it impossible to kernelize the edge features. For more details see [23].

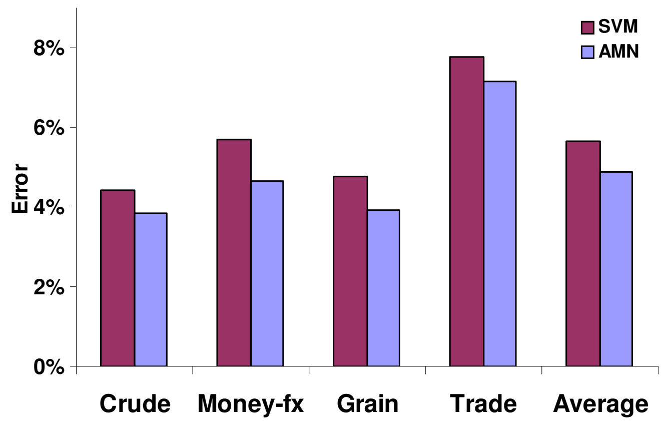

Figure 3.2: Comparison of test error of SVMs and AMNs on four categories of Reuters articles, averaged over 7-folds;

3.4.4 Experimental Results

We evaluated our approach on two tasks: newswire text classification and 3D terrain recognition.

Reuters

We ran our method on the ModApte set of the Reuters-21578 corpus. We selected four categories containing a substantial number of documents: crude, grain, trade, and money-fx. We eliminated documents labeled with more than one category, and represented each document as a bag of words. The resulting dataset contained around 2200 news articles, which were split into seven folds where the articles in each fold occur in the same time period. The reported results were obtained using seven-fold cross-validation with a training set size of ~ 200 documents and a test set size of ~ 2000 documents.

The baseline model is a linear kernel SVM using a bag of words as features. Since we train and test on articles in different time periods, there is an inherent distribution drift between our training and test sets, which hurts the SVM's performance. For example, there may be words which, in the test set, are highly indicative of a certain label, but are not present in the training set at all since they were very specific to a particular time period (see [26]).

Our AMN model uses the text similarity of two articles as an indicator of how likely they are to have the same label. The intuition is that two documents that have similar text are likely to share the same label in any time period, so that adding associative edges between them would result in better classification. Such positive correlations are exactly what AMNs represent. In our model, we linked each document to its two closest documents as measured by TF-IDF weighted cosine distance. The TF-IDF score of a term was computed as: $(1 + \log tf) \log \frac{N}{df}$ where tf is the term frequency, N is the number of total documents, and df is the document frequency. The node features were simply the words in the article corresponding to the node. Edge features included the actual TF-IDF weighted cosine distance, as well as

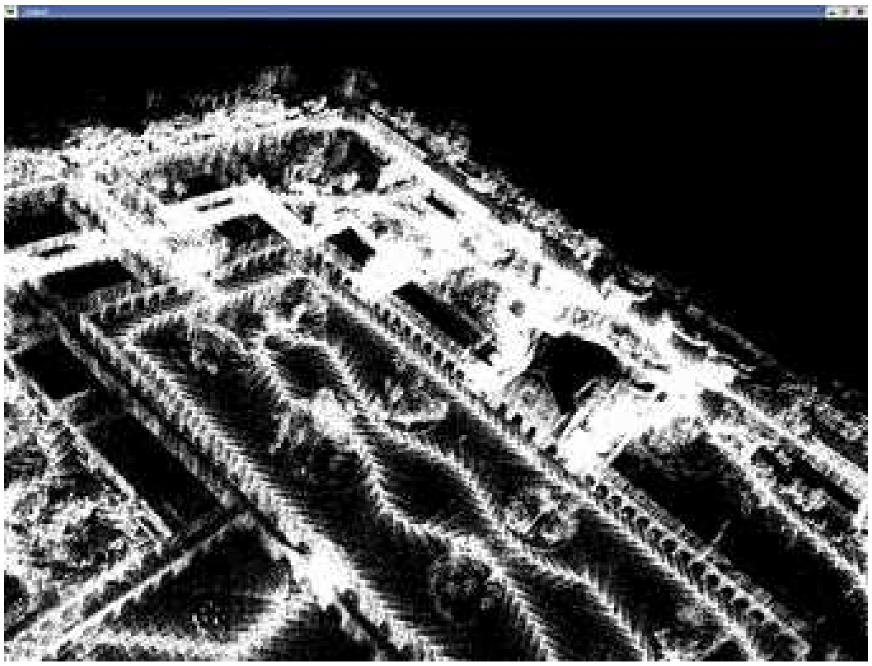
Figure 3.3: Overhead view of the Stanford Quad laser range data collected by the Segbot

the bag of words consisting of union of the words in the linked documents.

We trained both models (SVM and AMN) to predict one category vs. all remaining categories. Fig. 3.2(a) shows that the AMN model achieves a 13.5% average error reduction over the baseline SVM, with improvement in every category. Applying a paired t-test comparing the AMN and SVM over the 7 folds in each category, crude, trade, grain, money-fx, we obtained p-values of 0.004897, 0.017026, 0.012836, 0.000291 respectively. These results indicate that the positive interactions learned by the AMN allow us to correct for some of the distribution drift between the training and test sets.

Terrain Classification

Terrain classification is very useful for autonomous mobile robots in real-world environments. It can be useful to aid path planning, target detection, and as a pre-processing step for other perceptual tasks.

The Stanford Segbot Project provided us with a laser range maps of the Stanford campus collected by a Segway scooter-based moving robot equipped with SICK2 laser sensors. The data consists of around 35 million points, represented as 3D coordinates in an absolute frame of reference. Thus, the only available information is the location of points. Fig. 3.3 shows snapshot of an overhead view of the Stanford Quad.

Our goal is to classify the laser range points into four classes: *ground, building, tree, and shrubbery.* Since classifying *ground* points is trivial given their absolute z-coordinate, we classify ground points deterministically by thresholding the z coordinate at a value close to 0. After we do that, we are left with approximately 20 million non-ground points.

Features

Recall that each point is represented simply as a location in an absolute 3D coordinate system. That's why the features we use require pre-processing to infer properties of the local neighborhood of a point, such as how planar it is, or how much of the neighbors are close to the ground. The features described below are used in both flat and joint classification.

The features we use should be invariant to rotation in the x-y plane, as well as the density of the range scan, since scans tend to be sparser in regions farther from the robot.

1. **PCA-based features:** For each point we sample 100 points in a ball of radius 0.5 m. We then run PCA, to get the plane of maximum variance (spanned by the first two principal components.) We then form a 3x3x3 cube around the neighborhood of the point oriented with respect to the principal plan, and compute the percentage of points lying in the various sub-cubes. We use a number of features derived from the cube such as the percentage of points in the central column, the outside corners, the central plane, etc. These features capture the local distribution well and are especially useful in finding planes.

2. **Vertical features:** For each point we also take a cylinder of radius 0.2 m, which extends vertically to include all the points in a "column" of the range scan. We then compute what percentage of the points lie in various segments of this vertical column: e.g., between 2m and 2.5m, etc.

3. **Threshold feature** We also use an indicator feature of whether or not a point lies within 2m of the ground. This feature is especially useful in classifying shrubbery.

We use a quadratic kernel over all the features in both flat and joint classification.

Flat Model

The flat model is a multi-class SVM with a quadratic kernel over the above features. Fig. 3.4 shows a sample result:

We can see that the flat model achieves reasonable performance in a lot of places, but fails to enforce local consistency of the classification predictions. For example arches on buildings and other less planar regions are consistently confused for trees, even though they are surrounded entirely by buildings.

Voted Model

A possible improvement to the flat model is to take its predictions and smooth them using voting. For each point we look at its local neighborhood (we varied the radius to get the best possible results) and assign to the point the label of the majority of its neighbors. Fig. 3.5 shows a a sample result.

We can see that the voted model performs slightly better than flat: for example, it smoothes out trees and some areas of the buildings. Still it fails in areas like arches of buildings where the flat classifier has a locally consistent wrong prediction.

AMN Model

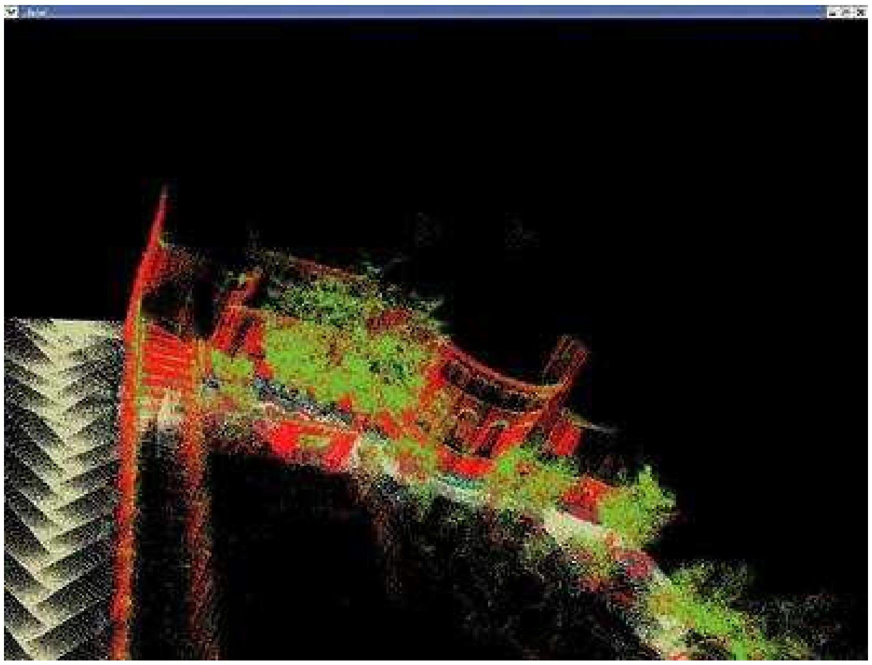

Figure 3.4: Typical results of flat multi-class SVM classifier. Note the overall lack of smoothness.

As we saw above there is room for improvement in increasing the local coherence of predictions. To do that we use a pairwise AMN over laser scan points, with associative potentials to ensure smoothness. Each point is connected to 6 of its neighbors: 3 of them are sampled randomly from the local neighborhood in a sphere of radius 0.5m, and the other 3 are sampled at random from the vertical cylinder column of radius 0.2m. It is important to ensure vertical consistency since frequently, the flat classifier is wrong in areas that are higher of the ground (due to the decrease in point density) or because objects tend to look different as we vary their z-coordinate (for example tree trunks and tree crowns look different). While we experimented with a variety of edge features including various distances between points, we found that even using a single "bias" feature which is always equal to 1 performed well.

For training we use CPLEX as the ICO QP solver. We select roughly 30,000 points that represent the classes well: a segment of a wall, a tree, some bushes. The pieces don't form a coherent segment of a scene.

Fig. 3.6 shows sample results achieved with the AMN model. We can see that the predictions are much smoother: for example building arches and tree trunks are predicted correctly.

A VRML fly-through animation of the same results is available at `http://ai.stanford.edu/~vasco/3d_results/`.

3.5 Conclusion

In this chapter we described how ICO can be applied to combinatorial problems and their convex optimization relaxations. We exploited the combinatorial nature of the output space

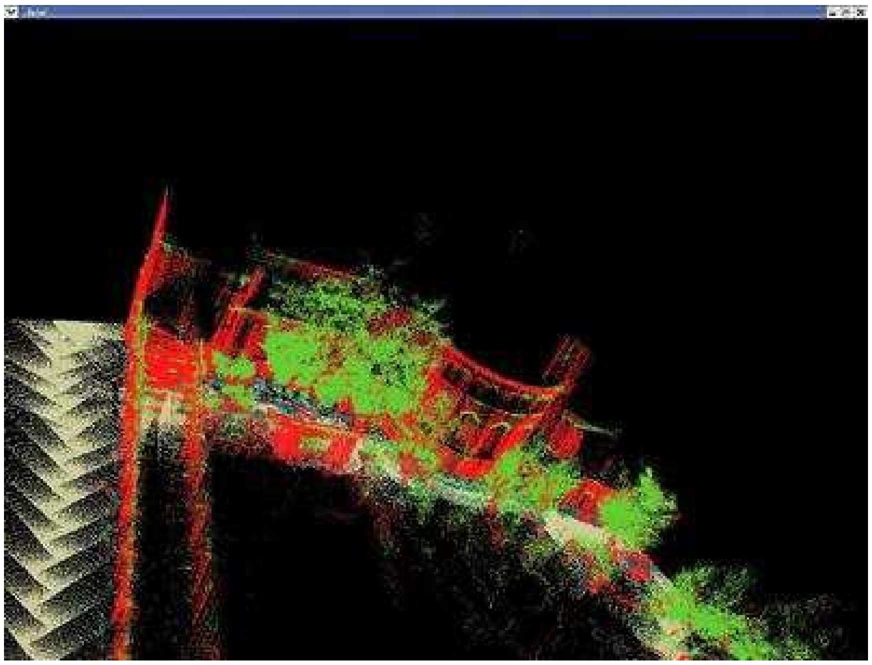

Figure 3.5: Typical results of the vote-smoother flat predictions. While the labels are somewhat smoother, there are still errors in areas where the flat classfier was consistently wrong.

to define a loss function based on Hamming distance, and examined a concrete application of ICO in associative Markov networks [23]. The efficiency of our formulation depended explicitly on our ability to compactly define Hamming distance as a linear function, and on the ability to formulate a convex relaxation to a combinatorial problem. While in associative Markov networks we observed that even relaxations which are not exact tend to produce good solutions in practice, it will be interesting to attempt to formally relate the quality of the learned weights to the quality of the approximation yielded by the combinatorial problem relaxation.

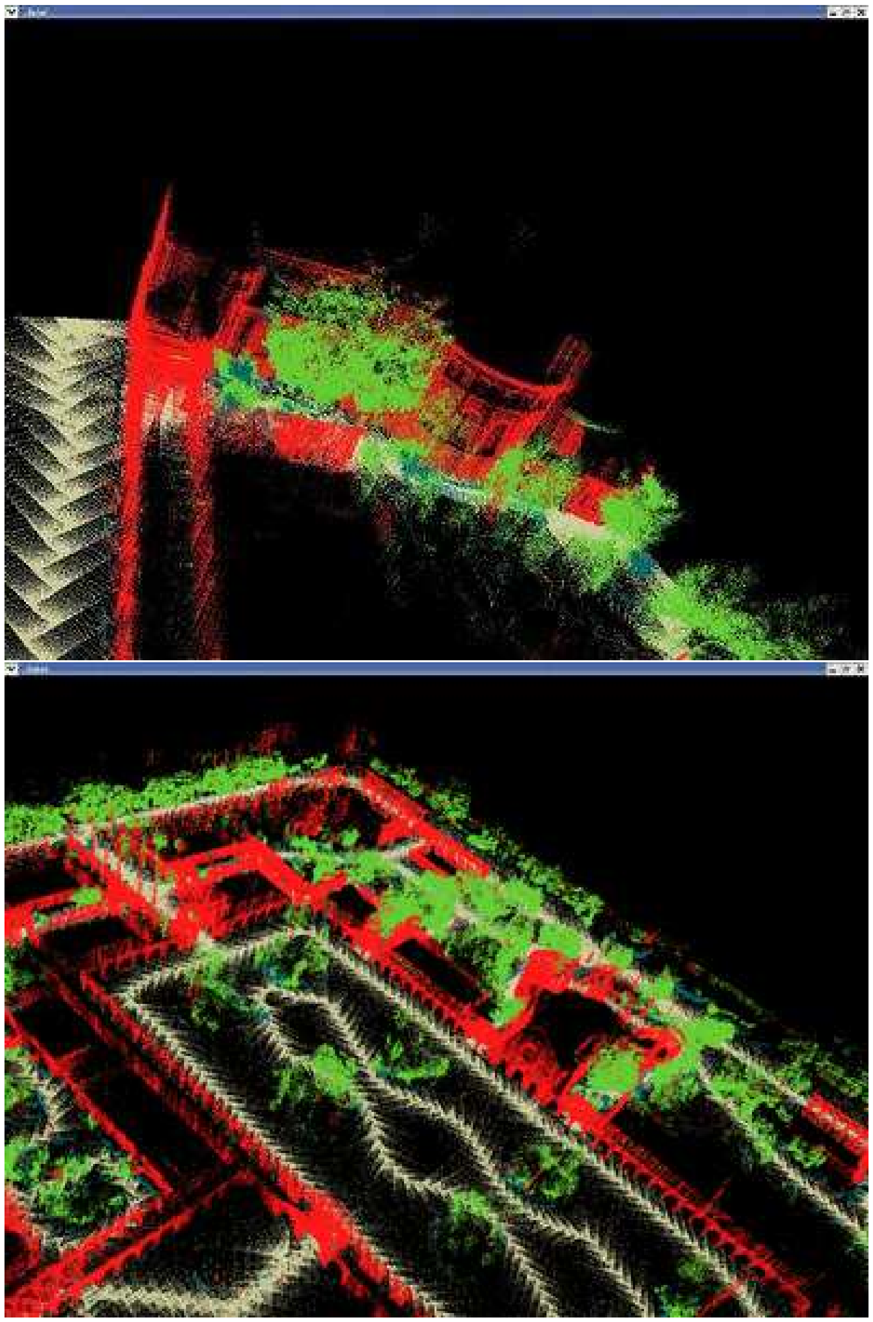

Figure 3.6: Results of the AMN model. We can see that predictions are consistently smoother across the board.

Chapter 4

Continuous Inverse Convex Optimization

In this chapter we turn to the problem of learning to optimize an objective over a continuous space. Thus in this case, for each instance $\mathcal{I}$, the target solution $\hat{\mathbf{y}}[\mathcal{I}] \in \mathrm{I\!R}^N$, for some N. We will see that the continuous nature of the target labels makes it harder to define a concave *loss function* $\ell(\mathbf{y}, \hat{\mathbf{y}})$ efficiently. We will also present experimental results in inverse Markowitz portfolio optimization showing the feasibility of the proposed algorithm, and its advantages over a simpler model which ignores the constraints of the target space.

4.1 Loss Function

Recall that in the combinatorial case, it was possible to define a loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ based on Hamming Distance. At first glance, a natural extension of that to the continuous case would be to simply use $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_1$ as the loss function, since it is a good measure of how far apart two different labelings are. However, that formulation will not work, since the ICO framework required a concave loss function, and L_1 norm is clearly not concave. Therefore, we need to select a different loss function.

4.1.1 Max-Norm

Consider $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_\infty$. For this loss function, we can provide a linear formulation, albeit one which is computationally more complex, by expressing the constraint $f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) \geq \max_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \Delta(\hat{\mathbf{y}}, \mathbf{y}) - \xi$ by a set of constraints, whose size is linear in the size of $\hat{\mathbf{y}}$. We add the following constraints:

$$f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) \quad \geq \quad \max_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + d_i(\hat{y}_i - y_i) - \xi, \quad \forall i, \; d_i \in \{1, -1\}$$

This set of $2N$ constraints is equivalent to a single constraint written in terms of $\|\hat{\mathbf{y}} - \mathbf{y}\|_\infty$. By letting d_i be either $+1$ or -1 we ensure that in one of the two cases the margin $d_i(\hat{y}_i - \hat{y}_i)$ is equal to $|\mathbf{y}_i - \hat{\mathbf{y}}_i|$ (the other one is strictly less than that). The combined effect of the $2N$

constraints is to ensure that the optimal $\mathbf{y}$ under $\mathbf{w}$ is at least $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \max_i |y_i - \hat{y}_i|$ better than any other one.

The resulting ICO QP is:

$$\begin{aligned}
\min_{\mathbf{w}:\ \mathbf{w} \geq 0} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \\
\text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) \geq \max_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + d_i(\hat{y}_i - y_i) - \xi, \quad \forall i,\ d_i \in \{1, -1\}
\end{aligned} \tag{4.1}$$

It can now be solved by using the technique discussed in Sec. 2.3.2: i.e., taking the dual of each of these $2N$ constraints, and using the dual objective in place of the primal maximization one.

4.1.2 Generalization

At first sight it seems counterintuitive that we can use L_∞ norm as the loss function, but not L_1 norm. In fact, the above construction could be adapted to work for an L_1 norm loss function also, but the computational cost would be prohibitive, since we would need to add an exponential number of constraints. We can view the L_1 norm as:

$$\|\mathbf{y} - \hat{\mathbf{y}}\|_1 = \sum_i |y_i - \hat{y}_i| = \max_{\mathbf{d}: d_i \in \{1, -1\}} \sum_i d_i(y_i - \hat{y}_i) \tag{4.2}$$

As before, in Eq. (4.2) we essentially allow d_i to control whether each term in the sum is positive or negative, and it is easy that the maximizing d_i would result in the expression $d_i(y_i - \hat{\mathbf{y}}_i)$ taking on its absolute value. If we enumerate all $\mathbf{d} \in \{-1, 1\}^N$, we can add the following constraints to the ICO problem:

$$f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}) \quad \geq \quad \max_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \mathbf{d}^\top(\hat{\mathbf{y}} - \mathbf{y}) - \xi \quad \forall i,\ \mathbf{d} \in \{\mathbf{1}, \mathbf{-1}\}^{\mathbf{N}}$$

Clearly the above construction is not computationally feasible since it would involve having 2^N constraints, but the ideas behind it are a generalization of the constraints we used to formulate the max-norm loss. In the max-norm construction, rather than enumerating each $\mathbf{d} \in \{-1, 1\}^N$, we simply restrict our attention to the set of vectors in which only one d_i is present, and other entries are 0: $\mathbf{d} = [0, \ldots, \{1, -1\}, \ldots, 0] : 1 \leq i \leq N$. The size of this set is linear, as opposed to exponential in the L_1 norm case.

4.1.3 ε-Window Loss

As an alternative to the max-norm loss function we can use an unstructured loss, analogous to the "hinge" loss ([10]) used in SVMs. Whereas before the L_∞ norm loss resulted in a margin which scaled with the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$, now we will penalize all $\mathbf{y}$ that are "different" from $\hat{\mathbf{y}}$ equally. Because we are working in continuous space, the "difference" will only be ensured up to some component-wise precision ε, which will only guarantee (in the separable case) that the weights we learn generate the true labeling $\hat{\mathbf{y}}$ up to the same precision ε. In order to accomplish that we need to add the following constraints:

$$f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}}) \;\geq\; \max_{\mathbf{y}\in\mathcal{Y},\; d_i(y_i-\hat{y}_i)\geq\varepsilon} f_{\mathbf{w}}(\mathbf{X},\mathbf{y}) + 1 - \xi \quad \forall i,\; d_i \in \{-1,1\}$$

The maximization constraint $d_i(y_i - \hat{y}_i) \geq \varepsilon$ reduces to either $y_i \geq \hat{y}_i + \varepsilon$ or $y_i \leq \hat{y}_i - \varepsilon$, depending on d_i, and effectively ensures that we only deal with $\mathbf{y}$ which are ε or more away from $\hat{\mathbf{y}}$ in at least one component. Thus, the full set of constraints above have the effect of ensuring that outside a window of ε around each component of the true labeling $\hat{y}_i$, the true label is better than all else by a margin of 1. By adding a constraint for each i, the ε-window holds for each y_i. By setting ε to be sufficiently small we can make sure that up to ε precision the learned weights will generate the true labeling $\hat{\mathbf{y}}$. The final ICO QP using the ε-window formulation has the form:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X},\hat{\mathbf{y}}) \geq \max_{\mathbf{y}\in\mathcal{Y},\; d_i(y_i-\hat{y}_i)\geq\varepsilon} f_{\mathbf{w}}(\mathbf{X},\mathbf{y}) + 1 - \xi \quad \forall i,\; d_i \in \{-1,1\} \end{aligned} \tag{4.3}$$

4.2 ICO for QPs

In this section we will give a complete ICO formulation for a general class of optimization problems: linearly constrained quadratic programs (QP). Such problems are often encountered in practice and are the next most widespread class of optimization problems, after linear programs (which are technically a type of QP). While we will work with the ε-window formulation for simplicity it is straightforward to derive the same formulation for the case of a max-norm.

4.2.1 QP Form

The specific problems we consider have a quadratic objective, and linear constraints.

As usual, we that the objective is linear in $\mathbf{w}$. We let $\mathbf{R}$ denote the features corresponding to the linear part of the objective, and $\mathbf{Q}_i$ denote one of the M_q feature matrices corresponding to the quadratic part of the objective. For a concrete example of such features see Sec. 4.3.2.

We will split up our weights vector as $\mathbf{w} = [\dot{\mathbf{w}}, \ddot{\mathbf{w}}]$, where $\dot{\mathbf{w}}$ are the weights corresponding to the linear part of the objective, and $\ddot{\mathbf{w}}$ are the weights corresponding to the quadratic part. $\ddot{w}_i$ is the ith entry of the $\ddot{\mathbf{w}}$ vector.

The QP problem in general form allows both equality and inequality linear constraints via the matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\begin{aligned} \max \quad & \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \left[\sum_{i=1}^{M_q} \ddot{w}_i \mathbf{Q}_i\right] \mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} \geq \mathbf{b} \\ & \mathbf{B}\mathbf{y} = \mathbf{c} \end{aligned} \tag{4.4}$$

From here on let $\mathbf{Q}(\ddot{\mathbf{w}}) = \left[\sum_{i=1}^{M_q} \ddot{w}_i \mathbf{Q}_i\right]$. Note that for the above problem to be convex, we require that $\mathbf{Q}(\ddot{\mathbf{w}}) \succeq 0$. To ensure that, we assume: $\mathbf{Q}_i \succeq 0$ and $\ddot{\mathbf{w}} \geq 0$.

4.2.2 ICO Formulation

We will now develop the ICO formulation for the case of the ε-window margin.

Let $\mathcal{Y}_{d_i} = \{\mathbf{A}\mathbf{y} \leq \mathbf{b},\ \mathbf{B}\mathbf{y} = \mathbf{c},\ d_i(y_i - \hat{y}_i) \geq \varepsilon\}$. Here is the ICO problem for the QP in Eq. (4.4):

$$\begin{aligned} \min_{\mathbf{w},\xi:\ \ddot{\mathbf{w}} \geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \dot{\mathbf{w}}^\top \mathbf{R}\hat{\mathbf{y}} - \frac{1}{2}\hat{\mathbf{y}}^\top \mathbf{Q}(\ddot{\mathbf{w}})\hat{\mathbf{y}} \geq \max_{\mathbf{y}\in\mathcal{Y}_{d_i}} \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y} + 1 - \xi \quad \forall i,\ d_i \in \{-1, 1\} \end{aligned} \tag{4.5}$$

While the above problem is convex, and thus solvable general algorithms ([5]), we would like to dualize the maximization in the constraint explicitly and convert the problem to a more standard form.

Dual

The primal problem in the maximization of each constraint is similar to the one in Eq. (4.4) except that we also have a restriction on y_i from the ε constraint. Let δ_{d_i} denote a vector of zeros with a d_i in the ith entry. The primal problem of each constraint is then:

$$\begin{aligned} \max \quad & \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} \geq \mathbf{b} \\ & \delta_{d_i}\mathbf{y} \geq \delta_{d_i}\hat{\mathbf{y}} + \varepsilon \\ & \mathbf{B}\mathbf{y} = \mathbf{c} \end{aligned} \tag{4.6}$$

We now form the Lagrangian. We introduce the vector of non-negative dual variables λ_{d_i} for the first set of constraints, the non-negative scalar σ_{d_i} for the second constraint, and the vector of free variables μ_{d_i}. We will subscript the dual variables by d_i to highlight the fact that the primal problem for the constraint depends explicitly on the exact value of $d_i \in \{-1, 1\}$.

$$\begin{aligned}\mathcal{L}^{d_i}(\mathbf{y}, \lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \;&=\; \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y} + {\lambda_{d_i}}^\top (\mathcal{A}\mathbf{y} - \mathbf{b}) + \\ &\quad {\mu_{d_i}}^\top (\mathbf{B}\mathbf{y} - \mathbf{c}) + \sigma_{d_i}\left(\delta_{d_i}\mathbf{y} - \delta_{d_i}\hat{\mathbf{y}} - \varepsilon\right)\end{aligned} \tag{4.7}$$

The dual function can be found by maximizing $\mathcal{L}^{d_i}(\mathbf{y}, \lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ with respect to the primal variables $\mathbf{y}$.

$$\nabla_y \mathcal{L}^{d_i}(\mathbf{y}, \lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \;=\; \mathbf{R}^\top \dot{\mathbf{w}} - \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y} + \lambda_{d_i}\mathbf{A} + \mu_{d_i}\mathbf{B} + \sigma_{d_i}\delta_{d_i} = 0 \tag{4.8}$$

$$y \;=\; \left(\mathbf{R}^\top \dot{\mathbf{w}} + \lambda_{d_i}\mathbf{A} + \mu_{d_i}\mathbf{B} + \sigma_{d_i}\delta_{d_i}\right)\mathbf{Q}(\ddot{\mathbf{w}})^{-1} \tag{4.9}$$

Let $\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) = \left(\mathbf{R}^\top \dot{\mathbf{w}} + \lambda_{d_i}\mathbf{A} + \mu_{d_i}\mathbf{B} + \sigma_{d_i}\delta_{d_i}\right)$. Note that $\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ is linear in both the dual variables and the weights $\mathbf{w}$.

Plugging in $\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ in the Lagrangian $\mathcal{L}^{d_i}(\mathbf{y}, \lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$, we can derive the dual function:

$$\mathbf{h}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \;=\; \frac{1}{2}\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})^\top \mathbf{Q}(\ddot{\mathbf{w}})^{-1}\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) - \lambda_{d_i}^\top \mathbf{b} - \mu_{d_i}\mathbf{c} - \sigma_{d_i}(\delta_{d_i}\hat{\mathbf{y}} + \varepsilon)$$

Eq. (4.10) is a quadratic form in $\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$, but we still face a problem. Recall that $\mathbf{Q}(\ddot{\mathbf{w}}) = \left[\sum_{i=1}^{M_q} \ddot{w}_i \mathbf{Q}_i\right]$. Unfortunately, we have no way of explicitly taking the inverse and thus exposing its dependence on the weights $\ddot{\mathbf{w}}$. Furthermore it is still not clear that the formulation of the dual function is convex in both the dual variables σ_{d_i}, λ_{d_i}, μ_{d_i}, and the weights $\dot{\mathbf{w}}$, $\ddot{\mathbf{w}}$. To establish convexity we first need to review the definition of Schur complement.

Schur Complement

In this section we give a brief review of Schur complement. For more details see [5].

Definition 4.2.1 *Consider a matrix X, partitioned as:*

$$X = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}.$$

If $\det A \neq 0$*, the matrix*

$$S = C - B^\top A^{-1} B$$

is called the Schur complement *of A in X.*

Theorem 4.2.2 *If $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.*

Proof See [5] ▮

Convexity of Dual

We can use Definition 4.2.1 and Theorem 4.2.2 to establish the convexity of Eq. (4.10), by considering the dual in epigraph form. The *epigraph* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as:

$$\mathbf{epi} f = \{(x,t) |\ x \in \mathbf{dom} f, f(x) \leq t\}$$

where $\mathbf{dom} f$ is the domain of the function f. A function is convex if and only if its epigraph is convex. We will directly show the convexity of the epigraph, by defining it via a set of convex inequalities.

Let $\hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ be the quadratic part of $\mathbf{h}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$.

$$\hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) = \frac{1}{2}\bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})^\top \mathbf{Q}(\ddot{\mathbf{w}})^{-1} \bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$$

Now, consider its epigraph:

$$\mathbf{epi}\ \ \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \quad = \quad \left\{ \left(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}, t^{d_i}\right) \mid \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \leq t^{d_i} \right\} \tag{4.10}$$

The condition $\hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \leq t^{d_i}$ can be rewritten as:

$$\left\{ \begin{bmatrix} \mathbf{Q}(\ddot{\mathbf{w}}) & \bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \\ \bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})^\top & t^{d_i} \end{bmatrix} \succeq 0, \ \sum_{j=1}^{M_q} \ddot{w}_j \mathbf{Q}_j \succ 0 \right\} \tag{4.11}$$

The Schur complement of the composite matrix in Eq. (4.11), is simply $t^{d_i} - \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$. The requirement that the matrix is positive semi-definite is equivalent to its Schur complement being positive semi-definite (by Theorem 4.2.2), which in turn implies that $t^{d_i} - \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \geq 0$. We are not guaranteed that the matrix $\mathbf{Q}(\ddot{\mathbf{w}})$ will be positive definite because the weights $\ddot{\mathbf{w}}$ could all be 0, but we can ensure positive definiteness by including a very small multiple of the indentity matrix in the sum (since we already have $\ddot{\mathbf{w}} \geq 0$.).

What is interesting is that the semi-definite inequality in Eq. (4.11) is simply a linear matrix inequality in all the variables we care about: $(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}, t^i)$, and therefore $\mathbf{epi}\ \ \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ is convex, and hence so is $\hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ ([5]).

Furthermore recall that $\mathbf{h}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) = \hat{\mathbf{h}}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) - \lambda_{d_i}^\top \mathbf{b} - \mu_{d_i}\mathbf{c} - \sigma_{d_i}(\delta_{d_i}\hat{\mathbf{y}} + \varepsilon)$, so it is also convex.

Final Formulation

We have established that the dual function $\mathbf{h}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ of the maximization in each constraint is convex. We now show the final ICO formulation. Define:

$$\mathbf{g}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \quad = \quad t^{d_i} - \lambda_{d_i}^\top \mathbf{b} - \mu_{d_i}\mathbf{c} - \sigma_{d_i}(\delta_{d_i}\hat{\mathbf{y}} + \varepsilon) \geq \mathbf{h}^{d_i}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \tag{4.12}$$

The inequality follows from the fact that when the constraints in Eq. (4.11) hold t^{d_i} as a tight upper bound on the quadratic part of $\mathbf{h}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ (tightness is achieved when the semi-definite constraint is active).

From Theorem 2.3.1 we know that for a feasible $(\lambda_{d_i}, \sigma_{d_i}, \mu_{d_i})$, we have

$$\mathbf{h}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \geq \max_{\mathbf{y} \in \mathcal{Y}_{d_i}} \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y}$$

Putting that together with Eq. (4.12) we get:

$$\mathbf{g}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \geq \max_{\mathbf{y} \in \mathcal{Y}_{d_i}} \dot{\mathbf{w}}^\top \mathbf{R}\mathbf{y} - \frac{1}{2}\mathbf{y}^\top \mathbf{Q}(\ddot{\mathbf{w}})\mathbf{y}$$

We are going to use $\mathbf{g}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ in place of $\mathbf{h}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ to get the final ICO formulation, since $\mathbf{g}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ is an upper bound on $\mathbf{h}_{\mathbf{w}}^{d_i}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})$ which is in turn an upper bound on the maximum of the primal objective in the constraint.

Here is the resulting ICO problem:

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \qquad\qquad (4.13)\\
\text{s.t.} \quad & \left.\begin{array}{l}
\ddot{\mathbf{w}} \geq 0 \\
\dot{\mathbf{w}}^\top \mathbf{R}\hat{\mathbf{y}} - \frac{1}{2}\hat{\mathbf{y}}^\top \mathbf{Q}(\ddot{\mathbf{w}})\hat{\mathbf{y}} - 1 + \xi \geq t^{d_i} - \lambda_{d_i}^\top \mathbf{b} - \mu_{d_i}\mathbf{c} - \sigma_{d_i}(\delta_{d_i}\hat{\mathbf{y}} + \varepsilon) \\
\begin{bmatrix} \mathbf{Q}(\ddot{\mathbf{w}}) & \bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i}) \\ \bar{\mathbf{y}}_{\mathbf{w}}(\lambda_{d_i}, \mu_{d_i}, \sigma_{d_i})^\top & t^{d_i} \end{bmatrix} \succeq 0 \\
\lambda_{d_i} \geq 0, \quad \sigma_{d_i} \geq 0
\end{array}\right\} \forall i, \ d_i \in \{1, -1\}
\end{aligned}$$

There is a total of $2 \times N$ constraints, which give rise to $\mathcal{O}(N^2)$ variables because the dualization of each constraint in Eq. (4.5) led to a linear number of dual variables. The optimization problem involves minimizing a quadratic function over linear and semi-definite constraints, and can be transformed into a general conic program, which can be solved using standard methods ([5, 1]).

4.3 Markowitz Portfolio Optimization

In this section we present experimental results on the problem of inverse Markowitz portfolio optimization which demonstrate that our formulation is able to learn weights whose resulting objective performs similarly to the convex objective which generated the training examples, even in the presence of noise.

4.3.1 Background

First we review the problem of portfolio selection. When one is faced with investing in assets there is always a risk-return tradeoff. For example, in order to obtain greater returns on an investment, one must often be willing to take on additional risk. While there are certain risk-free investments, in practice one cannot expect to achieve a useful return without taking on risk. Portfolio theory assumes that for a given level of risk, investors prefer higher expected returns to lower returns. Similarly, for a fixed level of expected return, investors should prefer less risk to more. For more details see [16].

Assume we have N assets available for consideration. We can measure returns as relative price changes over a certain period, and since they are random quantities we will need to estimate their means and covariances. Let r_i denote the mean return of asset i ($\mathbf{r}$ denotes the vector of mean returns), and let the matrix $\mathbf{Q}$ encode the covariance of the returns: i.e., $\mathbf{Q}_{i,j}$ is the covariance of the returns of the assets i and j. Also, let y_i be the fraction of the portfolio wealth we allocate to asset i, and $\mathbf{e}$ be the vector of ones. If we fix a desired expected level of return α, the classical portfolio optimization problem, introduced by Markowitz ([18], [17]) becomes:

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{y}^\top \mathbf{Q}\mathbf{y} && (4.14) \\ \text{s.t.} \quad & \mathbf{r}^\top \mathbf{y} \geq \alpha, \ \ \mathbf{e}^\top \mathbf{y} = 1, \ \ \ \mathbf{y} \geq 0 && (4.15) \end{aligned}$$

Eq. (4.14) finds the portfolio that minimizes risk, subject to the constraint that it achieves some minimum rate of return α. The positivity constraint on $\mathbf{y}$ requires that we not hold *short positions* (i.e., the obligation to buy the asset at the end of the period, resulting from our belief that its price will go down). The summation constraint corresponds to the fact that we allocate our entire portfolio to the assets, and is reasonable when we have a risk-free asset with positive return.

Similarly to Eq. (4.14) one can fix a maximum desired level of risk σ, and maximize expected return subject to that quadratic constraint. However, we would like to maximize return and minimize risk at the same time. In order to do that, it will be useful to assume that an investor has a certain *utility* for money, which determines exactly how much risk he is willing to take in order to obtain a certain return in expectation. We will assume an utilitity function of the form $u(x) = 1 - \exp(-\kappa x)$, where $\kappa > 0$ is a risk-aversion coefficient. We further assume that returns follow a normal distribution with mean $\mathbf{r}$ and covariance $\mathbf{Q}$. The return on the portfolio z is therefore also normal with mean $\mathbf{r}^\top \mathbf{y}$ and covariance $\mathbf{y}^\top \mathbf{Q}\mathbf{y}$.

We can now find the expected utility of the portfolio. Let $f(x)$ denote the probability density function.

$$\begin{aligned} \mathbb{E}[u] &= \int_{-\infty}^{\infty} (1 - \exp(-\kappa x)) f(x) dx \\ &= 1 - \frac{1}{\mathbf{y}^\top \mathbf{Q}\mathbf{y}\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\kappa x - \frac{1}{2}\left(\frac{x - \mathbf{r}^\top \mathbf{y}}{\mathbf{y}^\top \mathbf{Q}\mathbf{y}} \right)^2 \right) dx \\ &= 1 - \exp\left(-\kappa \mathbf{r}^\top \mathbf{y} + \frac{1}{2}\kappa^2 \left(\mathbf{y}^\top \mathbf{Q}\mathbf{y}\right)^2 \right) \end{aligned}$$

If we maximize the log of the expected utility with respect to the optimal portfolio $\mathbf{y}$ we get:

$$\begin{aligned} \max \quad & r^\top \mathbf{y} - \frac{\kappa}{2}\mathbf{y}^\top \mathbf{Q}\mathbf{y} \\ \text{s.t.} \quad & \mathbf{e}^\top \mathbf{y} = 1, \quad \mathbf{y} \geq 0 \end{aligned} \tag{4.16}$$

Thus the portfolio that maximizes the expected utility for an investor with a risk aversion coefficient κ can be found by solving the QP in Eq. (4.16) [21].

4.3.2 Feature-based Parameterization

In this section we will describe how to parameterize the portfolio selection problem based on a linear combination of *features*, so that we can later apply the ICO technique for learning the feature weights.

Mean Returns Features

We assume asset mean returns can be parameterized by a set of M_l features, such that the mean return of an asset is a linear combination of them. Let $\mathbf{R}_{j,i}$ be the value of the jth feature for the ith asset. We will also let $\dot{\mathbf{w}}$ denote the vector of weights associated with the return means features, with $\dot{w}_j$ being its jth entry. The mean return for the ith asset is then:

$$r_i = \sum_{j=1}^{M_l} \dot{w}_j \mathbf{R}_{j,i} \tag{4.17}$$

For example a feature could be the average return of an asset over the last T time periods, and we could have different features for different time windows T. Another feature could take on a value 1 if an asset is of a certain type (for example a stock in the finance industry) and a value 0 otherwise. It is possible to come up with many interesting features, whose linear combination can yield one's belief about the mean return of an asset. A reasonable way an investor might approach estimating return means could be to look at past returns, and combine it with an estimate of how assets of different types might move in the future. For example, if one believes that returns averaged over the last 10 time periods are a good prediction of future returns, and also that stocks in the finance industry are likely to suffer in the future, one can have a high weight for the feature corresponding to the average of returns over the last 10 time periods, and a negative weight for the feature which denotes whether or not an asset is a stock in the finance industry.

Letting $\mathbf{R}$ denote a matrix whose rows correspond to the feature values for each asset, and whose columns correspond to the different features, we can see that $\mathbf{r}^\top = \dot{\mathbf{w}}^\top \mathbf{R}$.

Return Covariance Features

We also assume that the covariance matrix of returns $\mathbf{Q}$ can be parameterized via a positive linear combination of M_q positive semi-definite feature matrices:

$$\mathbf{Q} = \sum_{j=1}^{M_q} \ddot{w}_j Q_j, \quad \ddot{w}_j \geq 0, \quad Q_j \succeq 0$$

We require that the weights be non-negative in order to ensure that the final covariance matrix $\mathbf{Q}$ is positive semi-definite. Examples of covariance matrices include covariance estimated from past returns of different time windows T, or outer product of indicators denoting which industry a stock is in. As an example of the latter feature consider whether or not an asset represents a stock in the finance industry. If we have N assets, we can denote that by an $N \times 1$ vector a, where $a_i \in \{0, 1\}$. We can then let one of our covariance matrix features be $Q_j = aa^\top$, which can be easily shown to be a positive semi-definite matrix.

Feature-based Portfolio Optimization

With the above definitions we can rewrite Eq. (4.16) as follows (we have dropped the risk-aversion coefficient since its effect can be absorbed in all the features):

$$\begin{aligned} \max \quad & \dot{\mathbf{w}}^\top \mathbf{R} \mathbf{y} - \frac{1}{2} \mathbf{y}^\top \left[\sum_{j}^{M_q} \ddot{w}_j \mathbf{Q}_j \right] \mathbf{y} \\ \text{s.t.} \quad & \mathbf{e}^\top \mathbf{y} = 1, \quad \mathbf{y} \geq 0 \end{aligned} \tag{4.18}$$

Notice that the above objective is linear in $\dot{\mathbf{w}}$ and $\ddot{\mathbf{w}}$, which is what we required for $f_{\mathbf{w}}(\mathbf{X}, \mathbf{y})$ in the ICO formulation.

4.3.3 ICO Formulation

We can apply the results of Sec. 4.2 by letting $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = \mathbf{e}$, $\mathbf{b} = 0$, $\mathbf{c} = 1$. We end up with an optimization problem involving $\mathcal{O}(N^2)$ variables and $\mathcal{O}(N)$ semi-definite and linear constraints.

4.3.4 Algorithm

In the experimental section we employ a cutting plane algorithm rather than the SDP formulation due to the limitations of current standard SDP packages in dealing with a large number of semi-definite constraints and a large number of variables.

The algorithm is an instance of a cutting plane method, and it relies on the existence of a *separation oracle* ([4], [1], [5]). A separation oracle is an abstract entity, which can be used to localize a point $\mathbf{x}$ in a convex set $\mathcal{X}$. When queried, it either asserts that $\mathbf{x} \in X$ or returns a separating hyperplane, i.e., $\mathbf{b}$ and d, so that $\mathbf{b}^\top \mathbf{z} \leq d, \quad \mathbf{z} \in \mathcal{X})$ and $\mathbf{b}^\top \mathbf{x} \geq d$.

The algorithm applies to a general ICO program of the following form:

```
1. Initialize $\mathcal{A} = \{\}$ , $\mathbf{w} = 0$, $\xi = 0$.
2. Set $\mathbf{w}, \xi := \arg\min_{\mathbf{w},\xi\in\mathcal{A},\ \mathbf{w}\geq 0} \frac{1}{2}\|\mathbf{w}\| + C\xi$
3. Set $violate := 0$
4. For each $j \in \mathcal{C}$,
    4.1. if $f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}_j) < \max_{\mathbf{y}\in\mathcal{Y}_j} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \ell_j(\hat{\mathbf{y}}_i, \mathbf{y}_i) - \xi$,
         add $\mathbf{w}^\top \mathbf{c}_j + \xi \geq b_j$ to $\mathcal{A}$, and set $violate := 1$
5. if $violate := 1$ goto 2.
5. Return $\mathbf{w}$, $\xi$.
```

Figure 4.1: The cutting plane algorithm for ICO programs. $\mathbf{c}_j$ and b_j result from the linearization of the jth constraint: i.e., keeping the violating $\bar{\mathbf{y}}_j$ as a constant, and retaining only the weights $\mathbf{w}$ and slack variable as variables.

$$\begin{aligned} \min_{\mathbf{w}:\ \mathbf{w}\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & f_{\mathbf{w}}(\mathbf{X}, \hat{\mathbf{y}}_j) \geq \max_{\mathbf{y}\in\mathcal{Y}_j} f_{\mathbf{w}}(\mathbf{X}, \mathbf{y}) + \ell_j(\hat{\mathbf{y}}_i, \mathbf{y}_i) - \xi \quad \forall j \in \mathcal{C} \end{aligned} \tag{4.19}$$

In Eq. (4.19) we use $\mathcal{C}$ to denote the set of all constraints, and index the specific loss function, and feasible set $\mathcal{Y}_j$ by j. This formulation is simply a more convenient way to capture both the max-norm, and ε-window margin, as well as problems with multiple training instances, which result in multiple margin constraints (which is why we also index $\hat{\mathbf{y}}_j$ by j.)

The algorithm is described in Fig. 4.1. It works by repeatedly adding linear constraints to a working set of constraints $\mathcal{A}$, and re-solving the linearly constrainted quadratic program: $\mathbf{w}, \xi := \arg\min_{\mathbf{w},\xi\in\mathcal{A},\ \mathbf{w}\geq 0} \frac{1}{2}\|\mathbf{w}\| + C\xi$. At every iteration, it then checks whether each constraint $j \in \mathcal{C}$ is satisfied with the current $\mathbf{w}$ and ξ, and adds a linearized version of the constraint to the set $\mathcal{A}$ if the constraint is violated. Checking whether a constraint holds involves carrying out the maximization over $\mathbf{y}$, yielding a $\bar{\mathbf{y}}_j$, which causes the violation if there is one. The terms $\mathbf{c}_j$ and b_j in Fig. 4.1 result from the linearization of the jth constraint. The linearization of the constraint simply uses the violating $\bar{\mathbf{y}}_j$ as a constant, making the constraint linear in $\mathbf{w}$ and ξ (recall that by assumption $f_{\mathbf{w}}$ is linear in $\mathbf{w}$). The algorithm stops when the current $\mathbf{w}$ and ξ satisfy all constraints.

The convergence of the algorithm follows from the fact that the constraints in Eq. (4.19) are convex (since $\mathcal{Y}_j$ is convex, and $f_{\mathbf{w}}$ and ℓ_j are concave in $\mathbf{y}$ by assumption). The *separation oracle* in this case is the maximization we carry out for each constraint. See [4] for a more in-depth discussion of cutting plane methods.

From a computational perspective, the algorithm is especially useful when checking whether a constraint is satisfied can be done efficiently. Since our constraints involve maximization, that might be problematic: for example in the case of portfolio optimization we

Train Set	CVX, HPQ, HAL, GM, F, MSFT, C, AAPL, BUD
Test Set	AMD, AMR, BAC, DB, ING, MC, NET, RTN, SNE

Table 4.1: Stock ticker symbols used in the train and test sets.

need to solve a linearly constrained QP at each constraint. That's why it is important to design the constraint checking part carefully, and reuse as much existing information as possible. For example in our experiments with inverse portfolio optimization, in each iteration we keep track of all violating $\bar{\mathbf{y}}_j$ along with the violating constraint maximization objective values. Then before rerunning the constraint checking QP for constraint j, we check if there is an existing $\bar{\mathbf{y}}_j$ which is feasible for the particular maximization problem in the constraint (i.e., $\bar{\mathbf{y}}_j \in \mathcal{Y}_j$), and has a corresponding objective value which will cause a violation. If so, we declare the constraint is violated and add the corresponding separating hyperplane (which is not necessarily the same as some previously added one if the loss terms $\ell(\mathbf{y}, \hat{\mathbf{y}})_j$ are different for different constraints. In cases when we need to run the constraint checking QP, we initialize the QP solver with the optimal $\mathbf{y}$ given the current weights $\mathbf{w}$ (ignoring the additional constraints that may be present for the different $\mathcal{Y}_j$.) It is also straightforward to parallelize the constraint checking part, which may be especially useful when dealing with large problems. It is possible to only go through the constraints until a violation is found, as opposed to adding the hyperplanes corresponding to all violations, but we found that such an approach has slower performance.

4.3.5 Results

We performed experiments using real stock return data, comparing the ICO approach to a linear regression approach which ignores the constraints on the output space and only projects its prediction onto the feasible set of portfolios (which is a probability simplex) after a prediction has been made.

We downloaded real stock return data of a set of 18 stocks from the Dow Jones Industrial Index. The ticker symbols we used are shown in Table 4.1. We converted weekly closing stock prices into relative returns, so in the following discussion a time period is a week. The stock data was split evenly into a train and test set with the data for half the stocks belonging to the train set and the data for the other half belonging to the test set.

For return means features we used several different features: the average observed return based over the last $\{2, 5, 10, 15\}$ time periods, the value of the Dow Jones Index, the values of the NASDAQ Index, as well as indicators of whether a stock is in one of the following general sectors: oil-related, computer-related, finance.

For covariance features we also used the covariance of stocks estimated based over the last $\{2, 5, 10, 15\}$ time periods, the identity matrix, as well as the outer products of the industry sector indicators: i.e., for each of the industry sectors above, we formed a matrix whose (i, j)-th entry represented whether the i-th and j-th stock are both in that industry sector.

We then randomly sampled weights 18 times, which we used to generate training example portfolios. The weights corresponding to the return means features were sampled from a normal distribution with mean 1, and standard deviation 0.5, and the weights corresponding to the covariance matrix features were sampled from a normal distribution with mean 3, and standard deviation 3 (we clipped negative covariance feature weights). The sampling distributions were selected so that they generate portfolios that cover more than one stock. For each weight sample, we generated a number of portfolios by solving the portfolio optimization problem using the weights and features derived from data from different time periods in the training set (spread out, so as to make the portfolios more independent.) We then ran ICO using a max-norm margin and constraint generation to learn a set of weights, which was then tested on data from the test set 7 different times (the choice of that number was motivated by limited data.) In order to make the learning problem more challenging, we added random Gaussian noise of mean 0 and varying standard deviation to each portfolio entry, and then projected the portfolio back to the feasible set, since addition of noise violates the summation and non-negativity constraints.

We compared the ICO approach against a linear regression model which ignores the constraints on the output, and simply projects its predictions after they are made to the set of feasible portfolios. Because the ICO model uses one set of weights to predict all stocks, we used the same regression model for predicting the fraction of the portfolio allocated to each stock. Thus the regression model's training examples consisted of a portfolio fraction, and features for each stock in a portfolio, for each example in the training set. The features included all the returns means features from above for the given stock, and the sum of the returns means features for the other stocks. In order to give the regression model some information about the covariance features, for each ICO covariance feature matrix, we included the diagonal entry corresponding to the given stock, as well as the sum of the other entries in the row.

Fig. 4.3.5 shows the results for different training set sizes and different noise levels. As we can see ICO outperforms the linear regression model significantly at all training set sizes. We can see that the regression model performs very poorly with just one portfolio example, whereas the ICO model benefits from directly modeling the constraints on the output space.

4.3.6 Conclusion

In this chapter we presented inverse convex optimization for continuous problems. Continuous problems present a challenge because it is more difficult to efficiently define a loss function for the margin. We presented two formulations of a margin: a max-norm based one, and one with a constant margin. We also showed experimental results in the domain of portfolio optimization which present the advantages of our approach over traditional unstructured approaches. Currently our approach yields quadratic-sized convex optimization problems, but it is an interesting area of future research to come up with more efficient formulations in which the number of constraints is linear in the problem size.

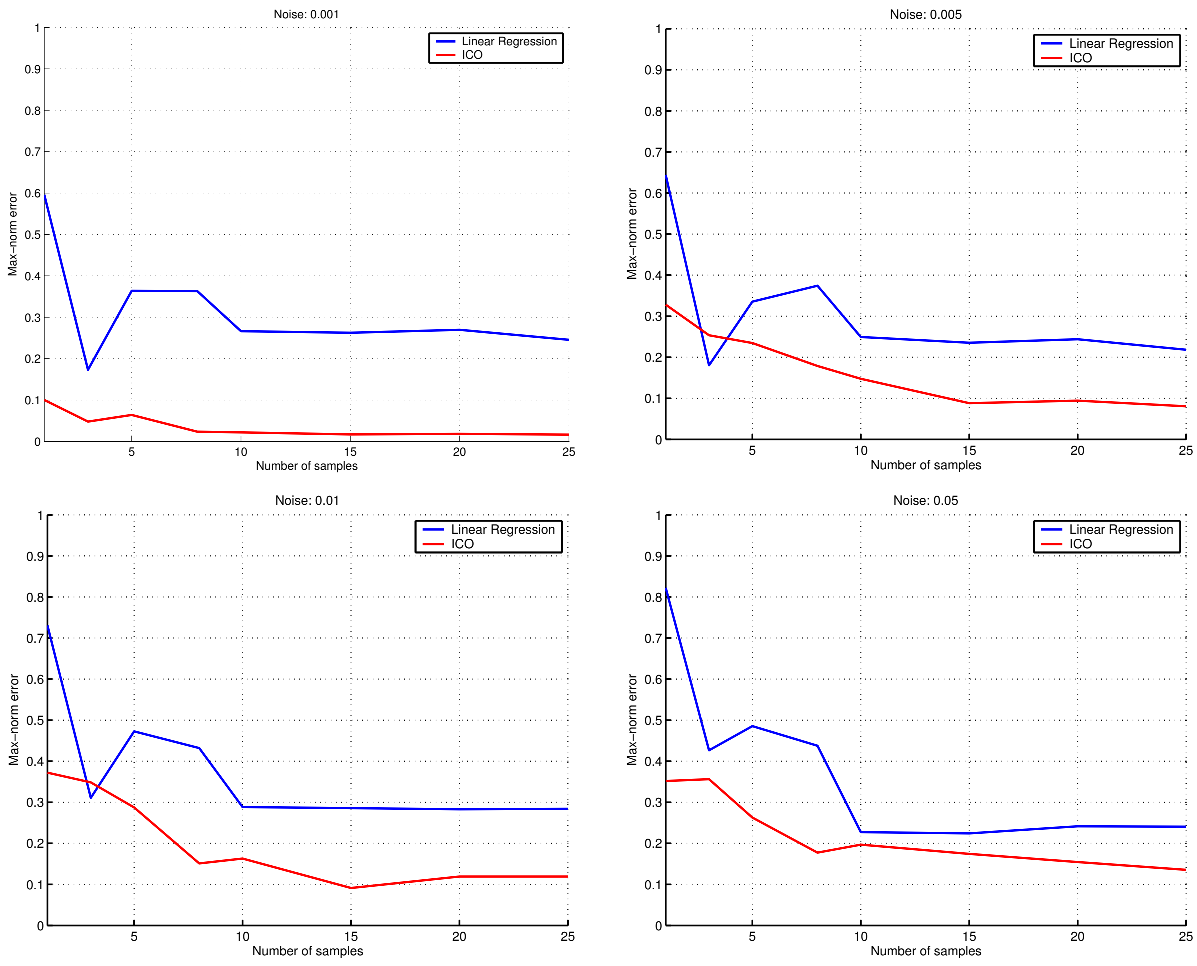

Figure 4.2: Results for different noise levels. Noise sampled from a normal distribution with the given standard deviation was added to each of the portfolios in the training set. The vertical axis shows the max-norm error: i.e., the maximum difference across all stocks between the predicted portfolio and the portfolio generated using the weights that also generated the training set.

Chapter 5

Conclusion

In this thesis we looked at the problem of *Inverse Convex Optimization*. We took a maximum-margin discriminative approach, and developed convex optimization formulation for the problem of learning the objective function of both continuous and combinatorial convex optimization problems. Our formulation relied on two major requirements: convexity of the optimization problem whose objective is learned, and the ability to define a concave loss function which serves as a measure of "distance" in the output space of the optimization problem. The generality of our framework makes it very attractive since a variety of practical problems are either naturally formulated as convex optimization problems or have convex relaxations. Furthermore our Inverse Convex Optimization framework yields a learning optimization problem which is convex, and thus solvable efficiently and reliably.

For combinatorial problems, our learning formulation exploited the binary nature of the output space by defining a Hamming distance loss function, which can be expressed as a linear function. Specifically we considered the associative Markov networks as an instance of Inverse Convex Optimization and showed that even when the combinatorial problem whose objective we are learning does not have an exact convex relaxation, our framework produces good results in practice. We also demonstrated an efficient graph-cut based inference algorithm based on past work in the vision community. The algorithm performs approximate inference by iterative graph cuts, and its most attractive properties are its near-linear performance and provable convergence to a near-optimal solution. The most important future research direction in ICO for combinatorial problems is relating the approximation guarantees of the convex relaxation of the combinatorial problem, to guarantees about the performance of the learning algorithm. It will also be beneficial to design special-purpose optimization algorithms which exploit the structure of specific ICO learning problems, such as associative Markov Networks.

For continuous optimization problems, the ICO framework used two different loss functions: a max-norm based one, and an absolute 0-1 loss within a certain precision ε. We presented experimental results on the task of inverse portfolio optimization in which our algorithm performed better than traditional approaches, which ignore problem structure. The main current limitation of the approach is that the number of constraints in the learning problem grows quadratically with the problem size. Thus it will be a very useful research direction to find more efficient formulations.

Acknowledgements

Most of this work was done in close collaboration with Ben Taskar and it is a generalization and extension of much of his past work on maximum margin methods. Thanks to Dinkar Gupta for his work on 3D terrain recognition, and Daphne Koller for helpful comments.

Appendix A

Fast AMN Inference

A.1 Overview

While in principle linear programming can be used for the MAP inference task in AMN, in this chapter we will describe a much more efficient graph-based inference algorithm for the case of pairwise AMNs. We use an existing algorithm, first analyzed in the vision community by Zabih *et al.*[8], and show experimentally that we are able to perform fast inference on very large models.

A.2 Background

A.2.1 Graphical Models Inference

Recall that a major task in probabilistic graphical models is the task of inference. When the model defines a probability distribution, there are a number of queries one can be interested in. For example we might need to find the (unconditional) marginal probability of a random variable, or a conditional probability given a set of evidence [20]. In the max-margin setting the most interesting inference query is the Maximum-A-Posteriori (MAP) inference: i.e. what is the most likely value of a random variable (perhaps conditional on some evidence), and that's what the rest of this chapter will focus on.

Both MAP and posterior probability inference can be shown to be NP-hard in a general graphical model [20]. The proof is omitted and relies on a reduction to the 3-SAT problem, by forming a suitable graph.

Still, as we have seen already, in AMNs MAP inference is tractable. We have already seen that MAP inference can be formulated as a linear program, and is hence efficiently solvable in polynomial time ([3]). In this chapter we will develop a different perspective on the problem, and use existing results to present an efficient graph-based algorithm for pairwise AMNs.

A.2.2 Mincut

Since our algorithm will use mincut as the main optimization tool, we will briefly review it.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with *non-negative* weights $c(u, v)$ associated with each edge (u, v). We add to the graph two special vertices (often referred to as terminal nodes): the source s and the sink t. An s-t cut is a partition of the set of vertices $\mathcal{V}$ into two disjoint sets S and T, such that $s \in S$ and $t \in T$. The cost of the cut is the sum of costs of all edges connecting nodes in S and T.

$$c(S,T) = \sum_{u \in S, v \in T, (u,v) \in \mathcal{E}} c(u,v)$$

The problem of finding the minimum-cost s-t-cut is commonly refferred to as simply *mincut*, and is solvable in polynomial time. The standard algorithm is due to Ford and Fulkerson [12], and relies on the duality the maximum flow between the source s and the sink t, and the cost of the minimum cut. Multiple other algorithms have also been proposed, each having its advantages in specific domains. For a more complete overview see [9].

There are a generalizations of the s-t mincut problem that handle multiple terminals (and thus produce a cut that splits the graph vertices into more than two subsets) but they are NP-hard [11].

The intuition used in the rest of the chapter is that if we restrict our attention to binary AMNs, we can treat the two terminals s and t as the 0 and 1 class, and thus interpret the cut as a binary labeling.

A.3 Efficient AMN Inference

A.3.1 Binary Case

We first consider the case of binary AMNs, and later show how to use the local search algorithm developed by Zabih *et al.*[8] to perform (approximate) inference in the general multi-class case. The fact that we will not be able to perform exact inference in the multi-class case should not come as a surprise since that problem is known to be NP-hard.

Formulation

Recall the integer programming formulation of MAP inference in AMNs:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{N} \sum_{k=1}^{K} \theta_i^k y_i^k + \sum_{(ij) \in E} \sum_{k=1}^{K} \theta_{ij}^{k,k} y_{ij}^k \qquad \text{(A.1)} \\
\text{s.t.} \quad & y_i^k \in \{0,1\}, \quad \forall i,k; \qquad \sum_k y_i^k = 1, \quad \forall i; \\
& y_{ij}^k \le y_i^k, \qquad y_{ij}^k \le y_j^k, \qquad \forall (ij) \in E, k.
\end{aligned}$$

We have already multiplied features and weights so that $\theta_i^k = \mathbf{w}_n^k \cdot \mathbf{x}_i$ and $\theta_{ij}^{k,k} = \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij}$. Since we assumed that $\mathbf{w}_e^{k,k} \ge 0$ and $\mathbf{x}_{ij} \ge 0$, we now know that $\theta_{ij}^{k,k} \ge 0$,which will be important later in constructing the mincut graph.

Consider the binary case, i.e., $K = 2$. The objective of the above integer program becomes:

$$\max \sum_{i=1}^{N} (\theta_i^0 y_i^0 + \theta_i^1 y_i^1) + \sum_{(ij)\in E} (\theta_{ij}^{00} y_{ij}^0 + \theta_{ij}^{11} y_{ij}^1)$$

Graph Construction

We will now show how to construct a graph for the above binary objective, whose *mincut* will correspond to the optimal MAP labeling. Our objective satisfies certain conditions proposed by Kolmogorov and Zabih ([15]) and thus the approach we use leverages existing techniques to construct the appropriate graph.

First, let's recast the object as minimization by simply reversing the signs on the value of each θ.

$$\min \sum_{i=1}^{N} (\theta_i^0 y_i^0 + \theta_i^1 y_i^1) - \sum_{(ij)\in E} (\theta_{ij}^{00} y_{ij}^0 + \theta_{ij}^{11} y_{ij}^1)$$

Since there was no restriction on the sign of θ_i^k, we simply hid the change of sign in the node terms, by negating the value of each θ_i^k. However, because we required that $\theta_{ij}^k \geq 0$, we will now subtract the edge terms, rather than adding it. The minimization formulation is made explicit because mincut performs minimization, and it will be easier to interpret the construction with it.

We now present the graph construction. The graph will consist of a vertex for each node in the AMN, along with the s and t terminals. In the final (S, T) cut, the S set will correspond to label 0, and the T set will correspond to label 1. We will show how to deal with the node terms (those depending only on a single variable) and the edge terms (those depending on a pair of variables), and then how to combine the two.

Node Terms

Consider a node term of the form $\theta_i^0 y_i^0 + \theta_i^1 y_i^1$. Such a term corresponds to the node potential contribution to our objective function for node i. Recall that θ_i as used here is the negative of the original θ_i since we cast the maximization as a minimization. For each node term corresponding to node i we add a vertex to the mincut graph v_i. We then look at $\Delta_i = \theta_i^0 - \theta_i^1$, and create an edge of weight $|\Delta|$ from v_i to either s or t, depending on the sign of Δ. The reason for that is that the final mincut graph must consist of only positive weights. An example is presented in Fig. A.3.1.

It is easy to see from Fig. A.3.1 that if the AMN consisted of only node potentials (which would correspond to flat classification), the graph construction above would add an edge from each node to its less likely label. Thus if we run mincut, we would simply get a cut with cost 0, since for each introduced vertex we have only one edge of positive weight to either source or sink, and we would always choose not to cut any edges.

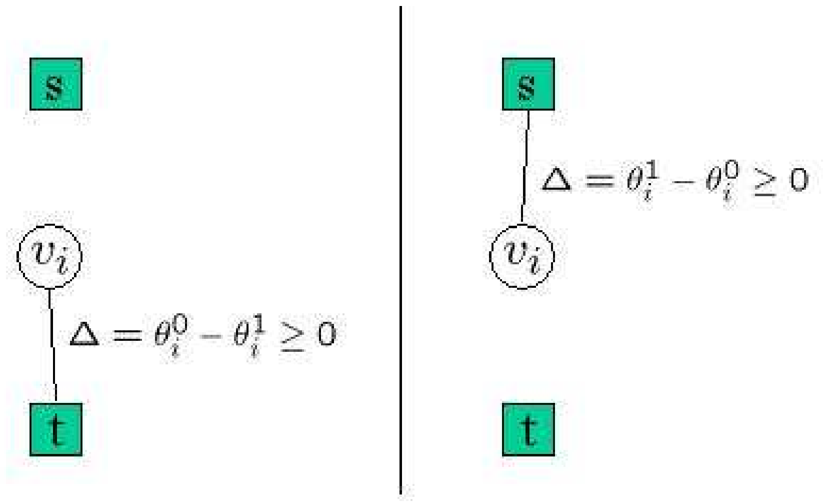
s
v_i
$\Delta = \theta_i^0 - \theta_i^1 \geq 0$
t
s
$\Delta = \theta_i^1 - \theta_i^0 \geq 0$
v_i
t

Figure A.1: Graph construction of node terms. Notice that if $\theta_i^0 \geq \theta_i^1$ we connect to the t node with weight $|\Delta|$.

Edge Terms

Now consider an edge term of the form $-\theta_{ij}^{00} y_{ij}^0 - \theta_{ij}^{11} y_{ij}^1$. Recall that the negative signs came about in recasting the original objective as minimization, and that $\theta_{ij} \geq 0$ by assumption. To construct a mincut graph for the edge term we will introduce two vertices v_i and v_j. We will connect vertex v_i to s with an edge of weight θ_{ij}^{00} and connect v_j to t with an edge of weight θ_{ij}^{11}. Fig. A.3.1 shows an example.

Observe what happens when both nodes are on the T side of the cut: the value of the mincut is θ_{ij}^{00}, which must be less than θ_{ij}^{11} or the mincut would have placed them both on the S side. Since the T side corresponds to label 1, this is exactly what we should expect. When looking at edge terms in isolation, a cut that places both nodes in different sets will not occur since $\theta_{ij}^{00} + \theta_{ij}^{11} \geq \theta_{ij}^{11}$ and $\theta_{ij}^{00} + \theta_{ij}^{11} \geq \theta_{ij}^{00}$, but when we combine the graphs for node terms and edge terms, such cuts will be possible.

Combining Terms

Interestingly we can take the individual graphs we created for node and edge terms and merge them by adding edge weights together (and treating missing edges as edges with weight 0). It can be shown that the resulting graph will represent the same objective (in the sense that running mincut on it will optimize the same objective) as the sum of the objectives of each graph. Since our MAP-inference objective is simply a sum of node and edge terms, merging the node and edge term graphs will result in a graph, whose mincut will correspond to the MAP labeling. The proof is omitted, but can be found in [15] as the *Additivity Theorem.*

Thus, by following the above procedure and then merging subgraphs, we can form a graph whose mincut will correspond to the optimal MAP labeling. The construction works

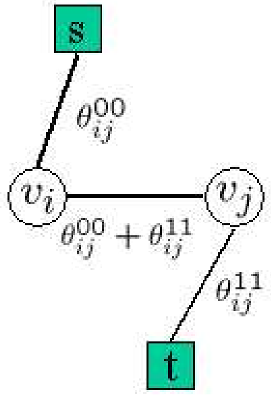

Figure A.2: Graph construction of edge terms. Observe what happens in the different possible cuts.

for the binary, pairwise case. While in the next section we will see how to handle multiple classes, it is still an open question how to do this efficiently for any clique size.

A.3.2 Multi-class Case

The graph construction above finds the best MAP labeling for the binary case, but in practice we would often like to handle multiple classes in AMNs. For example in automatic e-mail foldering we would like to assign an e-mail to one of several folders, or in laser range terrain classification we might be interested in classifying laser scan points as belonging to different objects such as trees, buildings or shrubbery.

One of the most effective algorithms for minimizing energy functions, which form a "metric" on a set of labels, is the α-expansion algorithm proposed by Boykov, Veksler and Zabih ([8]). The algorithm performs a series of "expansion" moves each of which involves optimization over two labels, and it can be shown that it converges to within a constant factor of the global minimum.

Expansion Algorithm

As we have shown the MAP objective we are minimizing is a metric on the set of labels, and thus we can apply the α-expansion algorithm proposed by Boykov, Veksler, Zabih (for a more detailed treatment see [8] or [28]). Note that the original paper required that the function being minimized is a metric over the set of labels, but that requirement was only necessary in constructing the mincut graph. As we saw in the previous section, using the more recent results of [15] we can construct a mincut graph, even though our objective is not a metric over the set of labels, so we will ignore that requirement.

Consider a current labeling y and a particular label $\alpha \in 1, \dots K$. Another labeling y' is called an "α-expansion" move from y if $y'_i \neq \alpha$ implies $y'_i = y_i$ (where y_i is the label of the ith node in the AMN.) In other words, an α-expansion from a current labeling allows each label to either stay the same, or change to α.

The α-expansion algorithm cycles through all labels α in either a fixed or random order, and finds the new labeling whose objective has the lowest value. It terminates when there

```
1. Begin with arbitrary labeling $y$
2. Set $success := 0$
3. For each label $\alpha \in \{1, \dots K\}$
    3.1 Compute $\hat{y} = \arg\min E(y')$ among $y'$ withing one $\alpha$-expansion of $y$.
    3.2 If $E(\hat{y}) < E(y)$, set $y := \hat{y}$ and success := 1
4. If $success = 1$ goto 2.
5. Return $y$
```

Figure A.3: Boykov, Veksler's and Zabih's α-expansion algorithm. $E(y)$ denotes the value of the AMN MAP objective in its minimization form for a particular labeling y.

is no α-expansion move for any label α that has a lower objective than the current labeling (Fig. A.3).

The key part of the algorithm for us becomes computing the best α-expansion labeling for a fixed α and a fixed current labeling . Fortunately for us, the mincut construction from earlier can allow us to do exactly that since an α-expansion move essentially minimizes a MAP-objective over two labels: it either allows a node to retain its current label, or switch to the label α. In this new binary problem we will let label 0 represent a node keeping its current label and label 1 will denote a node taking on the new label α. In order to construct the right coefficients for the new binary objective we need to consider several factors. Below, let $\theta_i'^k$ and $\theta_{ij}'^{k,k}$ denote the node and edge coefficients associated with the new binary objective:

1. **Node Potentials** For each node i in the current labeling whose current label is not α, we let $\theta_i'^0 = \theta_i^{y_i}$, and $\theta_i'^1 = \theta_i^{\alpha}$, where y_i denotes the current label of node i, and θ^{y_i} denotes the coefficient in the multiclass AMN MAP objective. Note that we ignore nodes with label α altogether since an α-expansion move cannot change their label.

2. **Edge Potentials** For each edge $(i, j) \in E$ whose nodes have labels different from α, we add a new edge potential, with weights $\theta_{ij}'^1 = \theta_{ij}^{\alpha,\alpha}$. If the two nodes of the edge currently have the same label, we set $\theta_{ij}'^0 = \theta_{ij}^{y_i,y_j}$, and if the two nodes currently have different labels we let $\theta_{ij}'^0 = 0$.

   For each edge $(i, j) \in E$ in which exactly one of the nodes has label α in the current labeling, we add $\theta_{ij}^{\alpha,\alpha}$, to the node potential $\theta_i'^1$ of the node whose label is different from alpha.

After we have constructed the new binary MAP objective as above, we can apply the mincut construction from before (after flipping signs of potentials to transform the problem to a minimization) to get the optimal labeling within one α-expansion from the current one, by using the fact that nodes with label 0 (i.e., the ones assosiated with the S side of the cut) retain their current label, and the other nodes change their label to α.

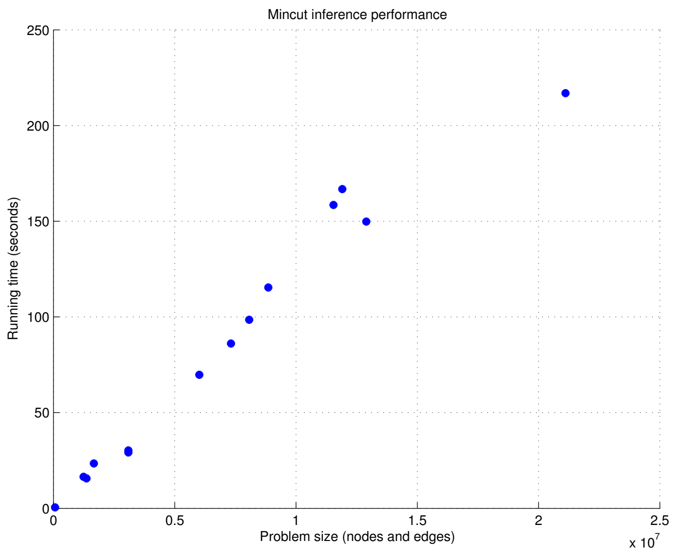

Figure A.4: The running time (in seconds) of the mincut-based inference algorithm for different problem sizes. The problem size is the sum of the number of nodes and the number of edges. Note the near linear performance of the algorithm and its efficiency even for large models.

Veksler ([28]) shows that the α-expansion algorithm converges in $O(N)$ iterations where N is the number of nodes. As noted in [8] and as we have observed in our experiments, the algorithm terminates only after a few iterations with most of the improvement occurring in the first couple of expansion moves.

It can also be shown that the minimum the algorithm converges to lies within a factor of 2 of the global minimum. The original proof presented in [8] and [28] shows that the local minimum is within a multiplicative factor which also depends on the function being minimized, but the proof can be adapted quite easily for the AMN MAP problem, and provide a constant factor approximation. This result is similar to the result of Kleinberg and Tardos [14], and the proof in Sec. 3.4.2.

A.4 Experiments

In this section we illustrate the scalability of the mincut-based inference algorithm, which was used in the task of 3D terrain recognition. The experimental setup is described in Sec. 3.4.4. We split up the dataset into 16 square (in the xy-plane) regions, the largest of which contains around 3.5 million points. The inference over each segment is performed using mincut with α-expansion moves as described above. The implementation is largely dominated by I/O time, with the actual mincut taking less than two minutes even for the largest segment. The performance is summarized in Fig. A.4, and as we can see, it is roughly linear in the size of the problem (number of nodes and number of edges). As a result we are able to handle very large networks, and it is an interesting challenge to apply the algorithm to even larger models, and come up with efficient distributed versions of it.

We used Kolmogorov's implementation of a modified mincut algorithm, which uses bidirectional search trees for augmenting paths (see [6]).

A.5 Summary

In this appendix we presented an graph cut based inference algorithm for pairwise AMNs, which is an alternative to linear programming. The algorithm is based on past results developed in the vision community, and its running time is practically linear for the task of 3D terrain recognition making it very attractive for very large models. We are able to perform inference on models with 3 million nodes and 18 million edges in under 4 minutes, which makes associative Markov networks suitable for large datasets. It is still an open question how to perform inference efficiently for larger clique sizes.

Bibliography

[1] Dimitri Bertsekas. *Nonlinear Programming.* Athena Scientific, 1995.

[2] J. E. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48, 1986.

[3] Dimitris Betsimas and John Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[4] Stephen Boyd and Lieven Vandenberghe. Localization and cutting plane methods. http://www.stanford.edu/class/ee392o, 2003.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[6] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. In *PAMI*, 2004.

[7] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR*, 1998.

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV*, 1999.

[9] C. S. Chekuri, A. V. Goldberg, D. R. Karger, M. S. Levine, and C. Stein. Experimental study of minimum cut algorithms. In *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 324–333, 1997.

[10] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[11] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proc. ACM Symp. Theory of Computing*, pages 241–251, 1992.

[12] L. Ford and D. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[13] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *J. R. Statist. Soc. B*, 51, 1989.

[14] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *FOCS*, 1999.

[15] V. Kolmogorov and R. Zabih. What energy functions can be minimized using graph cuts? In *PAMI*, 2002.

[16] David Luenberger. *Investment Science*. Oxford University Press, 1997.

[17] H. Markowitz. *Mean-Variance Analysis in Portfolio Choice and Capital Market*. Basil Blackwell, 1990.

[18] H. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Basil Blackwell, 1991.

[19] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. J. Wiley, New York, 1999.

[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[21] A. F. Perold. Large-scale portfolio optimization. *Management Science*, 30, 1984.

[22] R. B. Potts. Some generalized order-disorder transformations. *Proc. Cambridge Phil. Soc.*, 48, 1952.

[23] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. ICML*, 2004.

[24] B. Taskar, C. Guestrin, and D. Koller. Max margin markov networks. In *Proc. NIPS*, 2003.

[25] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proc. EMNLP*, 2004.

[26] B. Taskar, M.F. Wong, and D. Koller. Learning on the test data: Leveraging unseen features. In *Proc. ICML*, 2003.

[27] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology*, 6, 2003.

[28] O. Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, 1999.

[29] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Allerton Conference on Communication, Control and Computing*, 2002.