EE 381V: Large Scale Optimization **Fall 2012**

Lecture 15 — October 19

Lecturer: Caramanis & Sanghavi *Scribe: Tsung-Wei Huang*

15.1 Topics Covered Last Time

- Maximum independent set - dual form;
- Weighted maximum independent Set - dual form;
- Max cut - dual form.

In the last lecture, we covered several combinatorial optimization problems, and considered the notion of taking a dual as a solution or relaxation technique. Several NP-hard combinatorial optimization problems can be relaxed as easeily solvable convex optimization problems and thus can be efficiently solved or approximated.

In today's lecture, we discuss more related applications that can solved using convex optimization.

15.2 Application 1: Classification

Classification is a fundamental problem in machine learning and statistics. The problem is as follows. We are given a finite set of labelled points in R^n. Based on this labelled set of points, often called *training points*, our goal is to learn the labeling rule, so that we can accurately label future (unlabeled) points that we see. For instance, we may want to learn to classify an e-mail as spam or not-spam. Or we might want to classify a patient exhibiting certain symptoms as stable or in critical condition. There are many diverse such examples, which is why classification, along with regression, make up such an important part of pattern recognition and problems in machine learning.

More concretely, suppose we are given points in $\mathbb{R}^n$, $X = \{x_1, x_2, \cdots, x_N\}$ and then (for now, binary) labels in $\{-1, +1\}$: $Y = \{y_1, y_2, \cdots, y_N\}$. We wish to find a mapping function $f : R^n \to R$ that is as accurate as possible, in terms of finding a labeling rule. That is, we seek a function $f \in \mathcal{F}$ such that $\text{sgn} f(x_i) = y_i$. As the notation indicates, the function f should come from a restricted class of functions, $\mathcal{F}$, otherwise the result is typically no good for prediction, i.e., for labeling future points that arrive unlabeled. Discussion of how one restricts $\mathcal{F}$ is beyond the scope of this class (but see next semester's class!).

If indeed these inequalities hold, i.e., $\text{sgn} f(x_i) = y_i$ for all i, we say that the function f *separates*, *classifies*, or *discriminates* the set of training points. We sometimes also consider weak separation, in which the weak versions of the inequalities hold, i.e., $y_i \cdot f(x_i) \geq 0$ for all i.

15.2.1 Linear Separation

In linear separation, we seek an affine function $f(x) = a^T x - b$ that separates the two point sets, as shown in Figure 15.1. We have the following formulation: Find a hyperplane, parameterized by a normal vector w and offset b, such that

$$y_i \cdot (\langle w, x \rangle - b) > 0, \qquad i = 1, \cdots, N. \tag{15.1}$$

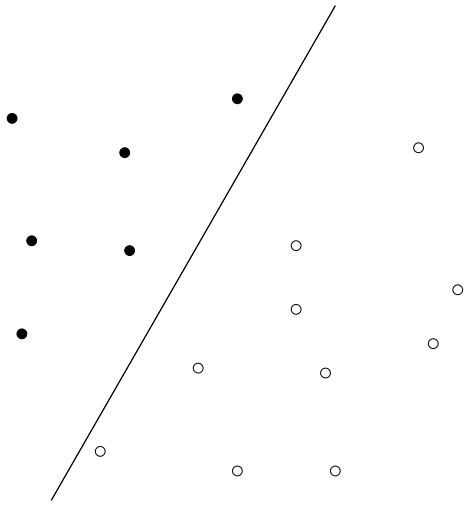

Figure 15.1. The points $\mathcal{X}_+ \triangleq \{x_i : y_i > 0\}$ and $\mathcal{X}_- \triangleq \{x_i : y_i < 0\}$ are shown as open and filled circles, respectively. These two sets are classified by an affine function f, whose 0-level set appears to be a separate line.

The geometric meaning is straightforward, and illustrated by the figure: we seek a hyperplane that separates the two point sets. Since the inequalities 15.1 are homogeneous in w and b, they are feasible if and only if the following non-strict linear inequalities hold:

$$\langle w, x \rangle - b \geq 1, \ \forall x_i \in \mathcal{X}_+, \qquad \langle w, x \rangle - b \leq -1, \ \forall x_i \in \mathcal{X}_-. \tag{15.2}$$

Take Figure 15.2 for example. We can assume the offset on b is zero, while targeting on examining the variable/vector w. Then we could have the following inequalities:

$$\begin{aligned} \Rightarrow \quad & \langle w, x \rangle \geq 1, \quad \langle w, y \rangle \leq -1 \\ \Rightarrow \quad & 8w \geq 1, \quad -3w \leq -1 \\ \Rightarrow \quad & w \geq 1/8, \quad w \geq 1/3 \\ \Rightarrow \quad & w \geq 1/3 \end{aligned} \tag{15.3}$$

From inequalities 15.3, it is desirable to maximize $1/\|w\|$. Indeed, we show below that this is proportional to the margin. Maximizing $1/\|w\|$ is the same as minimizing $\|w\|$, and

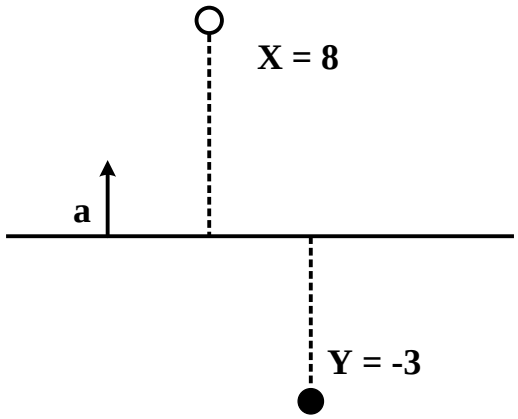

Figure 15.2. Example of finding inequalities (15.2).

therefore, we can formulate the entire linear classification problem into a convex optimization as follows:

$$\begin{aligned} \text{minimize :} \quad & \|w\|_2 && (15.4)\\ \text{s.t. :} \quad & \langle w, x_i\rangle - b \geq 1, \forall x_i \in \mathcal{X}_+ && (15.5)\\ & \langle w, x_i\rangle - b \geq -1, \forall x_i \in \mathcal{X}_-. && (15.6) \end{aligned}$$

15.2.2 Maximum Gap of Separation

We make the above assertion about margin maximization more precise here. Consider the setting depicted in Figure 15.1. There are infinitely many hyperplanes that separate the labelled points. One sensible choice is the one that *maximizes the margin*, that is, that maximizes the distance to the closest filled or empty circle (positively or negatively labelled point).

Consider any hyperplane $H = \{x : \langle w, x\rangle + b = 0\}$. It is easy to compute (try this!) the distance from the origin to the hyperplane:

$$\text{dist}(H, 0) = \frac{|b|}{\|w\|}.$$

Note that, as one would expect, this expression does not depend on the (simultaneous) scaling of w and b.

Now as mentioned above, we can always scale w and b so that the separation is given by

$$\langle w, x\rangle - b \geq 1,\ \forall x_i \in \mathcal{X}_+, \qquad \langle w, x\rangle - b \leq -1,\ \forall x_i \in \mathcal{X}_-.$$

Exercise: check that the margin is given by the following expression:

$$\text{margin} = \frac{|1-b|}{\|w\|} + \frac{|-1-b|}{\|w\|} = \frac{2}{\|w\|}.$$

Therefore, indeed, the margin is proportional to $1/\|w\|$, and therefore maximizing the margin is equivalent to minimizing $\|w\|$.

15.2.3 Non-Linear Separation

We have now examined the linear separation technique. But what happens if the data do not happen to be linearly separable? Figure 15.3 for example, illustrates such a setting, where linear separation is not possible. As the figure suggests, however, there does seem to be available a fairly simple classification rule. Indeed, labeling the center of the figure as the origin, it is straightforward to see that if we map the data *non-linearly* to an $(n+1)$-dimensional space via $x \mapsto (x, \|x\|)$, then there will be a (linear) hyperplane that separates the circles from the squares. This general idea of mapping non-linearly to a higher dimensional space where linear separation might be possible, is depicted in Figure 15.4.

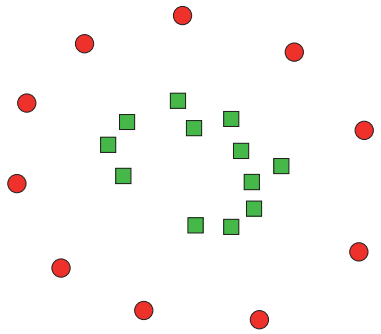

Figure 15.3. An example of non-linear data distribution in a two-dimensional (2D) plane.

This non-linear mapping, denoted by $\Phi(x)$ in Figure 15.4, is called the *kernel map*.

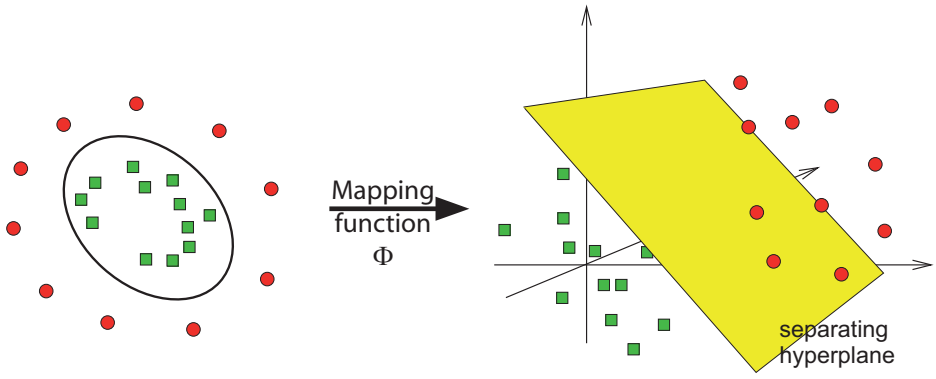

Figure 15.4. Mapping of the data to higher-dimensional space where a linear separation is possible.

In many practical applications, one often uses a mapping to a much higher dimensional – possibly even infinite dimensional – space. The immediate question is to understand how this impacts the optimization problem. Finding a hyperplane in n dimensions is an optimization problem in $(n+1)$ variables (one must find w and b, or only w if we impose a normalization and set $b = 1$). Then, if we lift to $N >> n$ dimensions, does computation time increase accordingly?

It turns out that this is not the case, as long as we use a special kind of non-linear mapping that has a special property. The property required is that it is "easy" to compute inner products in the high-dimensional space. That is, there is a kernel function K such that

$$\langle \Phi(x_i), \Phi(x_j) \rangle = K(x_i, x_j),$$

and $K(x_i, x_j)$ is easy to compute, where "easy" means that it is about as hard as computing $\langle x_i, x_j \rangle$ in the first place.

To see why this property of Φ is so important, and why when it holds it implies that we can quickly solve problems and find separating hyperplanes in much higher dimension without working any harder, we need to use the dual. Recall that the primal problem is as follows:

$$\begin{aligned} \min: \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.}: \quad & \langle w, x_i \rangle + b \geq 1, \quad \forall x_i \in \mathcal{X}_+ \\ & \langle w, x_i \rangle + b \leq 1, \quad \forall x_i \in \mathcal{X}_-. \end{aligned}$$

By writing the constraints above more compactly as

$$y_i(\langle w, x_i \rangle + b) - 1 \geq 0, \quad \forall i,$$

we can write the Lagrangian of this as (check this!):

$$L(w, b, \lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \lambda_i y_i(\langle w, x_i \rangle + b) + \sum_{i=1}^{n} \lambda_i.$$

Note that the objective function and the constraints are convex, and in particular, Slater's condition is satisfied. This means that strong duality holds. Recall that as a fundamental consequence of strong duality, we know that we have:

$$q(\lambda^*) = \max_{\lambda} \min_{w,b} L(w, b, \lambda) = \min_{w,b} \max_{\lambda} L(w, b, \lambda) = p(w^*, b^*),$$

where $p(\cdot, \cdot)$ denotes the primal function, and $q(\cdot)$ denotes the dual function. In particular, this implies, by convexity, that at (w^*, b^*, λ^*), the first order necessary conditions are satisfied:

$$\nabla_{w,b} L(w^*, b^*, \lambda^*) = 0, \quad \nabla_{\lambda} L(w^*, b^*, \lambda^*) = 0.$$

These give the conditions:

$$w = \sum_i \lambda_i y_i x_i,$$

and

$$\sum_i \lambda_i y_i = 0.$$

Substituting back in, we have derived the dual optimization problem:

$$\max: \ q(\lambda) = \max: \ \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle.$$

Notice the dependence on dimension: it is only there through the inner product, $\langle x_i, x_j \rangle$. In particular, if we were to use a nonlinear mapping Φ to map the data $\{x_i\}$ to a higher dimensional space, the resulting dual would simply be (check!)

$$\max: \ q(\lambda) = \max: \ \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \langle \Phi x_i, \Phi x_j \rangle.$$

If it happens that we have found an appropriate mapping function Φ that has the special property that $\langle \Phi x_i, \Phi x_j \rangle = K(x_i, x_i)$, for some easily computable function K, then we see that solving this optimization problem can be done essentially with no additional effort than the original one. This is the so-called *kernel trick* and it has been used in regression as well as in classification. We will learn more about kernel functions in the next semester.

One of the popular kernel mapping function is:

$$K(x_i, x_j) = e^{-\|x_i - x_j\|} \tag{15.7}$$

This kernel represents a mapping Φ to infinite dimensions. Without duality, solving for the linear separating hyperplane in infinite dimensions would be computationally hopeless.

15.3 Application 2: Congestion Control

Congestion control is an important topic in many network resource allocation problems. The allocation problem can be described as a constrained maximization of some utility function, which can be solved by convex optimization.

As shown in Figure 15.5, the flow network is a directed graph $G = V, L, C$, where V is the node set, L is the edge link, and C is the capacity value associated with each edge. We also have a set of source-sink pairs $source_s \to sink_s$ that can send a maximum amount of flow value x_s (sometimes called source rate). Also, each source-sink pair is associated with an utility U_s, which can be smooth, increasing, and concave. Then given a routing matrix $R_{ls} \in \{1, 0\}$, where the $R_{ls} = 1$ means the flow from source s use the edge link l or

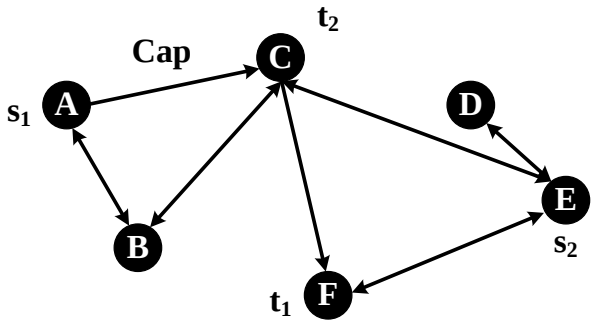

Figure 15.5. Example of a flow network.

$R_{ls} = 0$ otherwise. The objective is try to maximize the flow utility in the network. Thus, the network utility maximization problem can be formulated as follows:

$$\begin{aligned} Maximize: \quad & \textstyle\sum_s U_s(x_s) \\ s.t. \quad & \\ & RX \preceq C \\ & X \succeq 0 \end{aligned} \tag{15.8}$$

The problem given in (15.8) satisfies the condition of strong duality, i.e., the optimal primal value is equal to the optimal dual value. The Lagrangian dual form can be written as follows:

$$\begin{aligned} L(x,\lambda) &= \sum_s U_s(x_s) + \sum_l \lambda_l (c_l - \sum_s R_{ls} x_s) \\ &= \sum s[U_s(x_s) - (\sum_l R_{ls}\lambda_l)x_s] + \sum_l c_l \lambda_l \end{aligned} \tag{15.9}$$

Then the dual problem is:

$$\begin{aligned} Minimize: \quad & \textstyle\sum_s max_{x_s \geq 0}(U_s(x_s) - \lambda_s x_s) + \sum_l c_l \lambda_l \\ s.t. \quad & \lambda \succeq 0 \end{aligned} \tag{15.10}$$

Additivity of total utility and flow constraints lead to decomposition into individual source terms, a form of dual decomposition. This decomposition is called horizontal across users in network. In practice, suppose suppose each user is associated with a source-sink pair x_s. Then each user can keep his utility private and only needs to know λ_s to solve the problem $max_{x_s \geq 0}(U_s(x_s) - \lambda_s x_s)$.

15.4 Application 3: Manifold Learning

Manifold learning, also referred to as non-linear dimensionality reduction, pursuits the goal to *unfold/embed* data that originally lies in a high dimensional space into a low dimensional space, while preserving characteristic properties. This is possible since for any high dimensional data to be interesting, it must be intrinsically low dimensional.

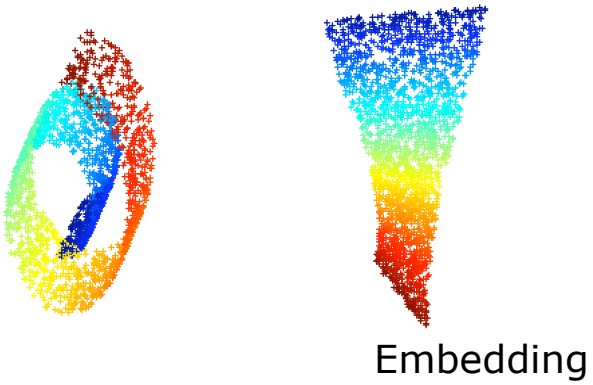

Figure 15.6. A curled data is unfolded/embedded into one dimensionality.

The idea to solve this problem is try to find approximate the one-dimensional distance to that in original space. Based on this intuition, we choose semidefinite embedding (SDE) algorithm for manifold learning. Imagining that each point is connected to its nearest nieghbors with a rigid rod. We take this structure and pull it as far apart as possible, i.e., we attempt to maximize the distance between points that are not neighbors. If the manifold looks like a curved space, then hopefully this procedure will unravel it properly.

To make this intuition a little more concrete, let us look at a simple example in Figure 15.7. Suppose we sample some points from a sine curve in R^2. The since curve is a one-dimensional manifold embedded into R^2, so let us apply to SDE idea to it to try to find a one-dimensional representation. We first attach each point to its two nearest neighbors. We then pull apart the resulting structure as far as possible. The entire steps is demonstrated in Figure 15.7. As the Figure shows, we have successfully found a one-dimensional representation of the data set. However, if the neighbors were chosen a bit differently, the procedure would have failed.

Now we examine the formulation of SDE. The primary constraint of the program is that distances between neighboring points remain or approximate the same. In other words, if x_i and x_j are neighbors then $\| y_i - y_j \| = \| x_i - x_j \|$, where y_i and y_j are the low-dimensional representatives of x_i and x_j. Translating the constraint into SDE is simple:

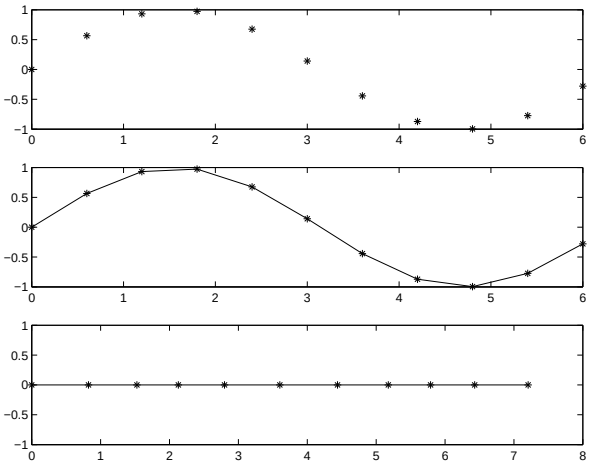

Figure 15.7. An illustration of the idea behind SDE. The top figure is a few points from a sine curve. The middle figure shows the result of connecting each point to its two neighbors. the bottom shows the result of the pulling step.

$$\begin{aligned} \| y_i - y_j \|^2 &= \| y_i \| + \| y_j \| - 2\langle y_i, y_j \rangle \\ &= \langle y_i, y_i \rangle + \langle y_j, y_j \rangle - 2\langle y_i, y_j \rangle \\ &= B_{ii} + B_{jj} - 2B_{ij} \\ &= \| x_i - x_j \|^2 \end{aligned} \tag{15.11}$$

Of course, we must have B to be semidefinite:

$$B \succeq 0 \tag{15.12}$$

Note these constraint (15.11) is only used for *neighboring* points. Then we wish to pull the remainder of the points as far apart as possible, so we maximize the inter-point distances of our configuration subject to the above constraint. The objective function is:

$$\begin{aligned} Maximize \quad : \quad & \sum_{i,j} \| y_i - y_j \|^2 \\ = \quad & \sum_{i,j} B_{ii} + B_{jj} - 2B_{ij} \\ = \quad & \sum_{i,j} B_{ii} + B_{jj} \\ = \quad & \mathbf{tr}(B) \end{aligned} \tag{15.13}$$

The semidefinite program may be solved by using any of a wide variety of software packages. One major gripe with SDE is that solving a semidefinite program requires tremendous

computational resources. State-of-the-art semidefinite programming packages typically cannot handle an instance more than 2000 points or so on a powerful desktop machine. This is a a rather severe restriction since real data sets often contain many more than 2000 points.