**COS 511: Theoretical Machine Learning**

Lecturer: Rob Schapire
Scribe: Charles Marsh

Lecture #13
March 25, 2014

# 1 Introduction to SVMs

## 1.1 Motivation

As we saw, the boosting technique wasn't defined so as to maximize margins, but we did end up using margins to analyze its performance. This begs the question: What if we derive an algorithm whose goal is, in fact, to maximize margins?

## 1.2 Assumptions

With boosting, we required the weak learning assumption. In this setting, we instead require that all features can be converted to real numbers, regardless of whether they're categorical, discrete, continuous, etc. This allows us to deal with feature vectors as if they're represented by points in Euclidean space, which we will assume for the remainder of the notes.

## 1.3 Intuition

To start, consider $m$ labeled points $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$ points in the plane, with $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, as seen in Figure 1.

To classify these points, the tendency is to draw some line between them and consider "above the line" to be labeled $+1$ and "below the line", $-1$. But there are an infinite number of such lines that cleanly divide the data, two of which are seen in Figure 1. Which do we prefer?

Some of these lines are intuitively "better" than others. For example, while both the green and orange lines cleanly separate the data, the green line appears to be "better" than the orange because it provides wider margins. For example, if we receive a new data point located at the black dot, the orange line will classify it as a negative example when in fact it's quite close to a positive example. In other words, the orange line just barely separates the data consistently, while the green line leaves some margin for error.

Intuitively, if we consider a ball of radius $\delta$ around every example, we'd hope that all points in this $\delta$-ball are classified with the same label as the given example. Thus, we want a hyperplane that not only classifies the training data correctly, but is also as far away as possible from all of the training points.

# 2 The Support Vector Machines Algorithm

The goal is to maximize the *margin* $\delta$ between the linear separator and the training data (note that in this setting, margins are not exactly the same as in boosting, although the two concepts are connected). There will always be examples that are exactly $\delta$ away from the separator, and these are known as the **support vectors**. In Figure 2, the support vectors are bolded with respect to the dotted linear separator.

Figure 1: Labeled examples in the plane

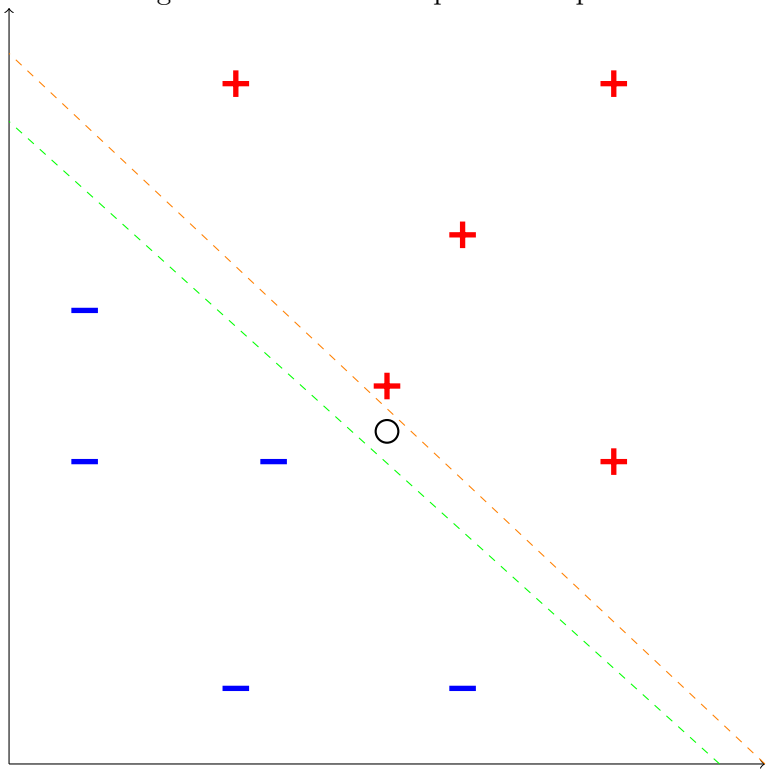
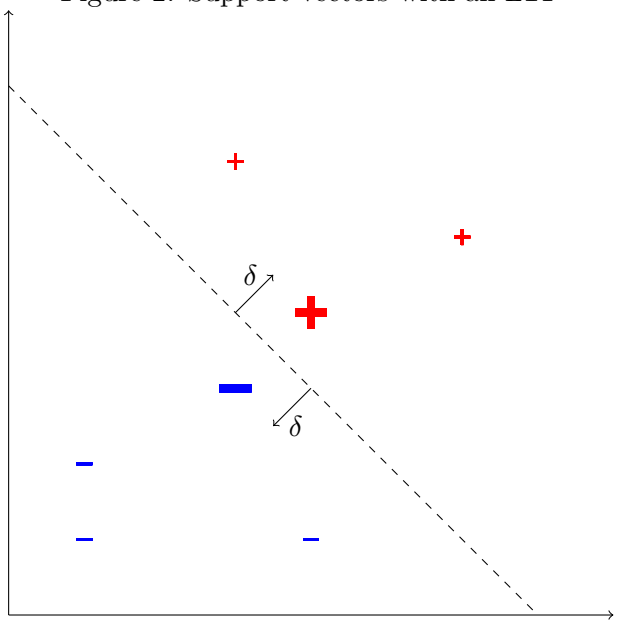


Figure 2: Support vectors with an LTF



### 2.1 VC-Dimension

Why is this a good approach? Can we find a more formal justification for maximizing the margin?

Recall that the VC-Dimension of an $n$-dimensional linear threshold function (LTF) through the origin is $n$. This may lead to acceptable generalization error bounds in low dimensions, but with SVMs, we'll often be working in spaces with hundreds of thousands of dimensions, so we'd like to do a lot better.

Assuming that all of our examples are contained within the unit ball ($\forall i : ||\mathbf{x}_i|| \leq 1$), we have the following nice result:

$$\text{VC-Dim(LTF with margin } \delta) \leq \frac{1}{\delta^2}$$

Note that if all our examples are contained with a ball of size $R$, we can normalize the data and the bound above becomes $(\frac{R}{\delta})^2$.

Why is this a nice result?

- It proves to us that larger margins leads to a smaller VC-Dimension, which in turn leads to better bounds on the generalization error.
- It's independent of the number of dimensions.

The proof of the bound on VC-Dimension will be given in a later lecture. An alternative bound on the generalization error can be achieved using Rademacher Complexity, the proof of which is very similar to that which was given for boosting.

## 3 How do we find these hyperplanes?

We start by formalizing the problem. Given that we're assuming our hyperplanes go through the origin, they can be defined by a single unit normal $\mathbf{v}$. That is, the hyperplane will be defined as all points orthogonal to this vector $\mathbf{v}$, with the additional constraint that $||\mathbf{v}|| = 1$.

For any given point $\mathbf{x}$, we'll also be interested in solving for the distance from the hyperplane to $\mathbf{x}$, which is simply:

$$\mathbf{v} \cdot \mathbf{x}$$

The scenario is laid out in Figure 3.

Next, note that this is a signed distance, such that:

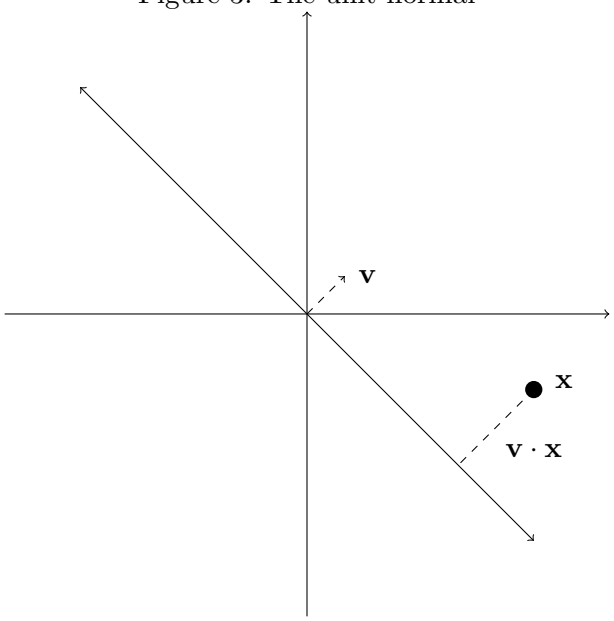
$$\begin{aligned} \mathbf{v} \cdot \mathbf{x} &> 0 \text{ if } \mathbf{x} \text{ is above the plane} \\ &< 0 \text{ if } \mathbf{x} \text{ is below the plane} \\ &= 0 \text{ if } \mathbf{x} \text{ is on the plane} \end{aligned}$$

As a result, we can classify a new point $\mathbf{x}$ simply by returning the sign of $\mathbf{v} \cdot \mathbf{x}$. Formally:

$$h(\mathbf{x}) = sign(\mathbf{v} \cdot \mathbf{x})$$

So under this setting, what problem are we trying to solve?

Figure 3: The unit normal




### 3.1 Formalizing the Problem

The problem statement is as follows:

Given $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$ with $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, maximize $\delta$ such that:

$$\begin{aligned} ||\mathbf{v}|| &= 1 \\ \forall i : \mathbf{v} \cdot \mathbf{x}_i &\geq \delta \quad \text{if } y_i = +1 \\ \mathbf{v} \cdot \mathbf{x}_i &\leq -\delta \text{ if } y_i = -1 \end{aligned}$$

We'll re-write this optimization problem several times.

#### 3.1.1 Rewrite #1

First, we can combine the two primary constraints and rewrite the problem as follows:

Maximize $\delta$ such that:

$$\begin{aligned} \forall i : y_i(\mathbf{v} \cdot \mathbf{x}_i) &\geq \delta \\ ||\mathbf{v}|| &= 1 \end{aligned}$$

#### 3.1.2 Rewrite #2

Next, define $\mathbf{w} = \frac{\mathbf{v}}{\delta}$. This implies that $||\mathbf{w}|| = \frac{1}{\delta}$ and, subsequently, $\delta = \frac{1}{||\mathbf{w}||}$. Maximizing $\delta$ is now akin to minimizing $||\mathbf{w}||$. Thus, the optimization problem can now be written as:

Minimize $||\mathbf{w}||$ such that:

$$\begin{aligned} \forall i : y_i(\frac{\mathbf{v}}{\delta} \cdot \mathbf{x}_i) &\geq \frac{\delta}{\delta} \\ y_i(\mathbf{w} \cdot \mathbf{x}_i) &\geq 1 \end{aligned}$$

This is ideal because the constraint on $\mathbf{v}$ can now be ignored. Further, the problem is defined solely in terms of a single vector $\mathbf{w}$ and has no dependence on $\delta$.

To avoid dealing with square roots, we redefine the objective to be:

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{Such that } \forall\, i : y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1$$

### 3.2 The Lagrangian

As this problem is defined on a convex function subject to convex constraints, it is known as a "Convex Program". We'll proceed using standard techniques for solving such programs.

First, we rewrite our constraints as:

$$b_i(\mathbf{w}) \geq 0$$
$$\text{where: } b_i(\mathbf{w}) = y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1$$

Next, we define the Lagrangian function $L(\mathbf{w}, \boldsymbol{\alpha})$ such that:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i b_i(\mathbf{w})$$

**Claim.** *The previous formulation is identical to solving:*

$$\min_{\mathbf{w} \in \mathbb{R}^n} [\max_{\alpha_i \geq 0} [L(\mathbf{w}, \boldsymbol{\alpha})]]$$

*Proof.* Consider the optimization problem to be a game between two players Mindy and Max. Mindy's goal is to minimize the above function $L(\mathbf{w}, \boldsymbol{\alpha})$, and she gets to pick $\mathbf{w}$. Max, knowing Mindy's choice of $\mathbf{w}$, aims to maximize the above function by picking $\boldsymbol{\alpha}$ subject to the constraint that all $\boldsymbol{\alpha}_i \geq 0$.

Say Mindy picks $\mathbf{w}$ such that $b_i(\mathbf{w}) < 0$ for some $i$. Then, Max will pick $\alpha_i = \infty$, as he wants to maximize the function and $-b_i(\mathbf{w})\infty = \infty$ in this case. But this would be a terrible result for Mindy, who wants to minimize the function. As such, she will never choose $b_i(\mathbf{w}) < 0$, lest she allow Max the opportunity to earn a payoff of $\infty$.

Consider the two remaining cases:

- $b_i(\mathbf{w}) = 0 \implies \alpha_i$ is irrelevant as it contributes nothing to the Lagrangian function.
- $b_i(\mathbf{w}) > 0 \implies \alpha_i = 0$. Recall that $\alpha_i \geq 0$ by definition. Max's only choices are to play $\alpha_i = 0$ or $\alpha_i > 0$. If he chooses $\alpha_i > 0$, he'll be decreasing the value of the function. Thus, the best he can do is play $\alpha_i = 0$.

In either case, it holds that $\alpha_i b_i(\mathbf{w}) = 0$. Thus, the sum on the right will disappear and the optimization problem will resolve to minimizing $\frac{1}{2}||\mathbf{w}||^2$, just as in the previous formulation. □

### 3.3 The Convex Dual

For any function on two parameters, it can be shown that:

$$\min_{\mathbf{w}}[\max_{\boldsymbol{\alpha}}[L(\mathbf{w}, \boldsymbol{\alpha})]] \geq \max_{\boldsymbol{\alpha}}[\min_{\mathbf{w}}[L(\mathbf{w}, \boldsymbol{\alpha})]]$$

Under certain conditions, we can go further and show that the two expressions are in fact equal. The most important of these conditions is that the function $L$ is convex in $\boldsymbol{\alpha}$ and concave in $\mathbf{w}$. As it turns out, our choice of $L$ indeed satisfies this along with other necessary conditions, giving us equality:

$$\min_{\mathbf{w}}[\max_{\boldsymbol{\alpha}}[L(\mathbf{w}, \boldsymbol{\alpha})]] = \max_{\boldsymbol{\alpha}}[\min_{\mathbf{w}}[L(\mathbf{w}, \boldsymbol{\alpha})]]$$

The RHS of this equation is known as the **convex dual**. It too is a convex optimization problem. The dual is often easier to solve, but more importantly, it can reveal important facts about the structure of the problem itself.

Define:

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}}[\min_{\mathbf{w}}[L(\mathbf{w}, \boldsymbol{\alpha})]]$$
$$\mathbf{w}^* = \arg\min_{\mathbf{w}}[\max_{\boldsymbol{\alpha}}[L(\mathbf{w}, \boldsymbol{\alpha})]]$$

Then, we have the following series of equations:

$$\begin{aligned} L(\mathbf{w}^*, \boldsymbol{\alpha}^*) &\leq \max_{\boldsymbol{\alpha}} L(\mathbf{w}^*, \boldsymbol{\alpha}) \\ &= \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha}) \\ &= \max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}) \\ &= \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}^*) \\ &\leq L(\mathbf{w}^*, \boldsymbol{\alpha}^*) \end{aligned}$$

### 3.4 Saddle Points

Examining two of the equations above, we have:

$$\min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}^*) = L(\mathbf{w}^*, \boldsymbol{\alpha}^*)$$

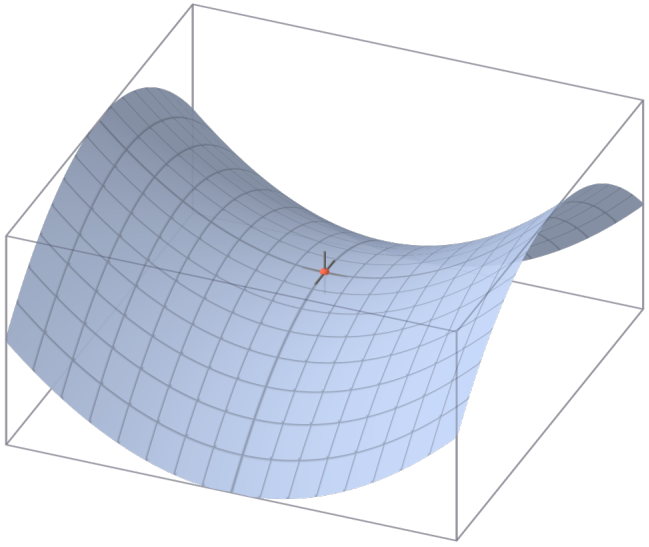
In other words, $\mathbf{w}^*$ minimizes $L$ when the choice of $\boldsymbol{\alpha}*$ is fixed. Similarly, $\boldsymbol{\alpha}^*$ maximizes $L$ when the choice of $\mathbf{w}$ is fixed. This implies that the pair of solutions $(\mathbf{w}^*, \boldsymbol{\alpha}^*)$ form a saddle point, as seen in Figure 4.

### 3.5 KKT Conditions

We already saw that:

$$\begin{aligned} &\forall i : b_i(\mathbf{w}^*) \geq 0 \\ &\alpha_i^* \geq 0 \\ &\alpha_i^* b_i(\mathbf{w}^*) = 0 \end{aligned}$$

Figure 4: Saddle point



In addition, if $\mathbf{w}^*$ is in fact the minimizer, then the partial derivatives of the Lagrangian with respect to each component $w_j$ should also be zero. This imposes an extra condition:

$$\forall j : \frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*)}{\partial w_j} = 0$$

Together, these are known as the Karush-Kuhn-Tucker (KKT) conditions.

We can solve this expression as follows:

$$\begin{aligned} \frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*)}{\partial w_j} &= 0 \\ &= w_j - \sum_{i=1}^{m} \alpha_i y_i x_{ij} \\ \implies \mathbf{w} &= \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \end{aligned}$$

The $\mathbf{w}$ derived above is the weight vector that minimizes $L$ for *fixed* $\boldsymbol{\alpha}$. The simple formula implies that for any $\boldsymbol{\alpha}$, we can find the appropriate weight vector. In particular, if we can find $\boldsymbol{\alpha}^*$, we can find $\mathbf{w}^*$.

As a next step, we can plug $\mathbf{w}$ back into $L$, which gives us the following:

$$\begin{aligned} L(\mathbf{w}, \boldsymbol{\alpha}) &= L(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, \boldsymbol{\alpha}) \\ &= \sum_i \alpha_i - \frac{1}{2} \sum_i^m \sum_j^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x_j}) \end{aligned}$$

This represents the minimum over $\mathbf{w}$. To finish solving, we need to take the maximum over $\boldsymbol{\alpha}$ of the above expression subject to $\alpha_i \geq 0$. In effect, this represents the dual of the previous problem.

To find $\boldsymbol{\alpha}^*$, we can use iterative methods along the lines of the above until we reach the desired saddle point. And, as previously demonstrated, finding $\mathbf{w}^*$ given $\boldsymbol{\alpha}^*$ is merely a matter of plugging $\boldsymbol{\alpha}^*$ into the equation $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$.

### 3.6 Support Vectors & Generalization Error

All the conditions outlined above will still hold at our solution $(\mathbf{w}^*, \boldsymbol{\alpha}^*)$. In particular:

$$\alpha_i^* b_i(\mathbf{w}^*) = 0$$
$$\alpha_i^* [y_i(\mathbf{w}^* \cdot \mathbf{x}_i) - 1] = 0$$

Therefore, if $\alpha_i^* \neq 0$, it follows that $y_i(\mathbf{w}^* \cdot \mathbf{x}_i) = 1$. This implies that example $(\mathbf{x}_i, y_i)$ is a support vector, as we can substitute $\mathbf{w} = \frac{\mathbf{v}}{\delta}$ to get:

$$y_i(\mathbf{v} \cdot \mathbf{x}_i) = \delta$$

This satisfies exactly the definition of a support vector in that example $(\mathbf{x}_i, y_i)$ is exactly $\delta$ away from the hyperplane.

Further, if $(\mathbf{x}_i, y_i)$ is *not* a support vector, then working backwards, you can see that $b_i(\mathbf{w}^*) \neq 0$, which implies that $\alpha_i = 0$. As a result, the solution $\mathbf{w}$ will be a linear combination of the support vectors! Therefore, our output hypothesis can be defined just in terms of these support vectors, a subset of the overall training data.

Say there are $k$ such support vectors and our algorithm outputs hypothesis $h$. As our output hypothesis is a function of a subset of the training data, we can use the result of Homework 2, Problem 4 to bound the generalization error:

$$err(h) \leq \tilde{O}(\frac{k + \ln(\frac{1}{\delta})}{m})$$

This result is particularly appealing as it is independent of the number of dimensions.

## 4 Linear Inseparability: Soft Margins

Until now, we've assumed that our data is linearly separable. That is, we've assumed that there exists a valid half plane that can divide the data consistently. Often, however, this assumption does not hold.

It might be the case, instead, that we have a few examples that are incorrectly classified by our LTF, as in Figure 5. We'd like to somehow *move* the bad examples to the other side of the hyperplane. But for this, we'd have to pay a price.

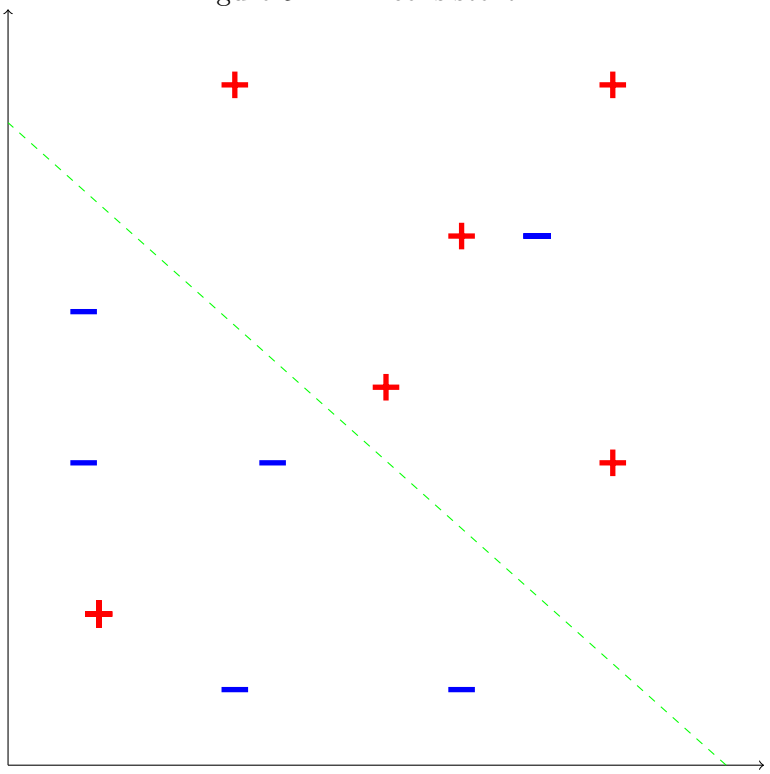
In the previous setting, our goal was to minimize $\frac{1}{2}||\mathbf{w}||^2$ subject to the constraint:

$$\forall i: \; y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1$$

However, in this new setting (in which we're allowed to nudge our examples around slightly), we pay a different price. The problem setting is as follows:

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$
$$\text{Such that } \forall\, i: \; y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

Figure 5: An Inconsistent LTF



Essentially, $\xi_i$ is the amount which we move example $i$, and $C$ is some positive constant. This approach to linear inseparability is known as the "Soft Margins" technique.