CIS5200: Machine Learning Spring 2025

Lecture 9: Support Vector Machines

Date: February 13, 2025

In this lecture, we'll cover our next machine learning algorithm: the **Support vector machine (SVM).** SVMs are one of the most successful linear models of all time, and there's a particular notion of "quality" under which they are in fact the *optimal* linear classifier.

Lecture Roadmap:

1. **The Margin:** A geometric measure of classifier quality
2. **Hard Margin SVMs:** Finding the optimal separating hyperplane
3. **Soft Margin SVMs:** Handling non-separable data
4. **Connection to ERM:** SVMs as regularized optimization

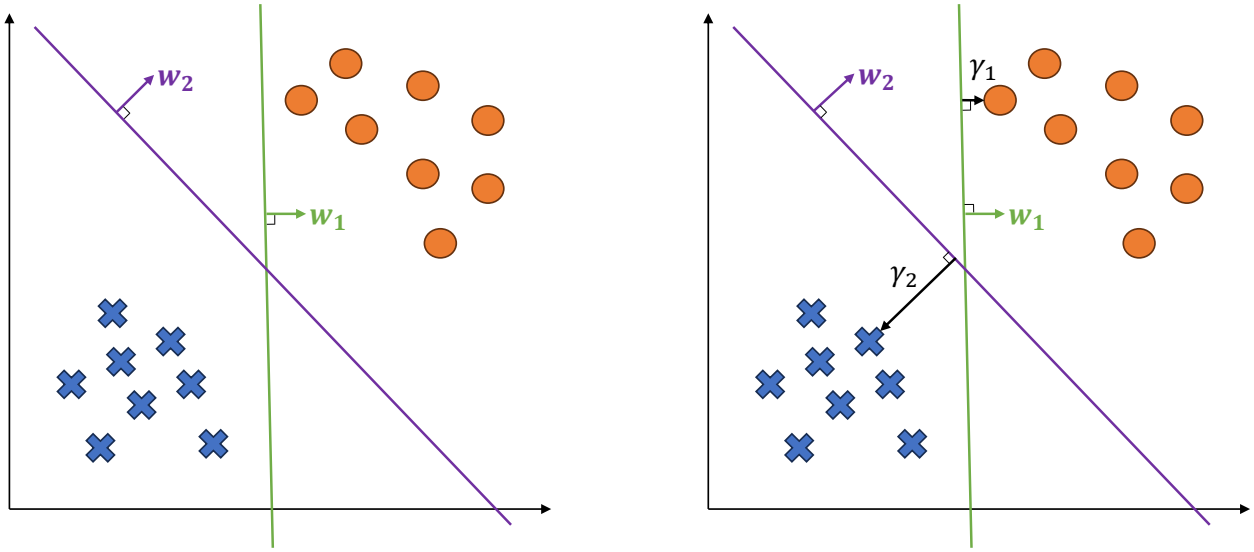

Figure 1: **Left:** Intuitively, the purple hyperplane parameterized by $\mathbf{w}_2$ is a "better" classifier than the hyperplane parameterized by $\mathbf{w}_1$, because it seems to leave more room for error. **Right:** the margin–a concept we've briefly touched on before–seems to formalize this concept nicely. The green hyperplane has a much smaller margin.

To begin, consider the dataset in the figure above. Intuitively, it seems like the purple hyperplane parameterized by $\mathbf{w}_2$ is "better" than the green one parameterized by $\mathbf{w}_2$. The reason for this is that there are orange circles very close to the green hyperplane – it's not implausible to think that there could be orange circle test points just to the left of the given training data that we would classify incorrectly. Put another way, the purple hyperplane is better because it seems to "leave

more room for error." On the right, we see that we can formalize this intuition using a quantity we've already seen before: the *margin*. The purple hyperplane is "better" because it has a larger margin. SVMs are linear models with the maximum possible margin given a dataset.

Note: Recall that the Perceptron algorithm guaranteed to find a hyperplane that separates the data as long as one exists. In fact, since we assumed separation with margin, there are infinitely many hyperplanes that separate the data, and the Perceptron may find any one of these. In particular, Perceptron algroithm did not provide any guarantees on the margin of the hyperplane found, even though we assumed that the training dataset had margin γ with respect to the true separator. However, in Homework 1, we saw a simple modification of the Perceptron to the Margin Perceptron which guaranteed an approximate margin of $\gamma/3$. SVM will guarantee maximum margin of γ.

1 The Margin Revisited

Recall from the perceptron that we defined the margin $\gamma(\mathbf{w}, S)$ of a hyperplane $\mathbf{w}$ and dataset S as the smallest distance between any point in S and the hyperplane. Since the margin is going to be such a critical quantity to our discussion of SVMs, let's talk about how to derive it. This time, we won't assume that the hyperplane is defined by a normalized vector $\mathbf{w}$ – instead we're given some arbitrary vector $\mathbf{w}$ that defines a hyperplane. As we'll see, we get the "right thing" for normalized vectors anyways.

To derive the margin, we make the following observations about the distance from a point $\mathbf{x}$ to some hyperplane parameterized by $\mathbf{w}$:

1. For any point $\mathbf{x}$, there is a point $\mathbf{x}_p$ that is the perpendicular projection from the hyperplane to the point.
2. Because the vector pointing from $\mathbf{x}_p$ to $\mathbf{x}$, $\mathbf{d} = \mathbf{x} - \mathbf{x}_p$, is perpendicular to the hyperplane, it is parallel to $\mathbf{w}$. As a result, it must be the case that $\mathbf{d} = \alpha\mathbf{w}$.
3. Since $\mathbf{x}_p$ is on the hyperplane, we know that $\mathbf{w}^\top\mathbf{x}_p = 0$ by definition of the hyperplane parameterized by $\mathbf{w}$.

These facts are enough to compute the vector pointing from the hyperplane to the point $\mathbf{x}$:

$$\mathbf{w}^\top\mathbf{x}_p = 0 \qquad (\mathbf{x}_p \text{ is on the hyperplane})$$

$$\mathbf{w}^\top(\mathbf{x} - \mathbf{d}) = 0 \qquad (\mathbf{d} = \mathbf{x} - \mathbf{x}_p \text{ is the distance vector})$$

$$\mathbf{w}^\top(\mathbf{x} - \alpha\mathbf{w}) = 0 \qquad (\mathbf{d} = \alpha\mathbf{w} \text{ since } \mathbf{d} \parallel \mathbf{w})$$

$$\mathbf{w}^\top\mathbf{x} - \alpha\mathbf{w}^\top\mathbf{w} = 0 \qquad (\text{expand})$$

$$\alpha = \frac{\mathbf{w}^\top\mathbf{x}}{\mathbf{w}^\top\mathbf{w}} = \frac{\mathbf{w}^\top\mathbf{x}}{\|\mathbf{w}\|_2^2} \qquad (\text{solve for } \alpha)$$

$$\mathbf{d} = \alpha\mathbf{w} = \frac{\mathbf{w}^\top\mathbf{x}}{\|\mathbf{w}\|_2^2}\mathbf{w} \qquad (\text{distance vector in terms of } \mathbf{w})$$

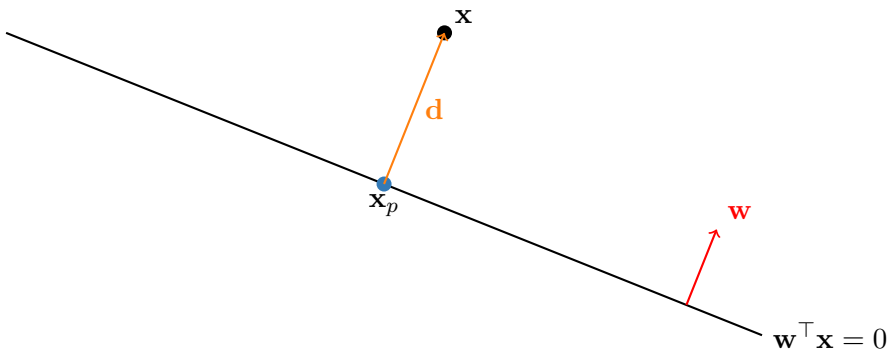

Figure 2: Geometric interpretation of margin calculation. The point $\mathbf{x}_p$ is the perpendicular projection of $\mathbf{x}$ onto the hyperplane. The distance vector $\mathbf{d}$ is parallel to the normal vector $\mathbf{w}$, and its length gives the distance from the point to the hyperplane.

Given this vector $\mathbf{d}$, we compute its length to get the distance from $\mathbf{x}$ to the hyperplane:

$$\|\mathbf{d}\|_2 = \left\| \frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|_2^2} \mathbf{w} \right\|_2 = \frac{|\mathbf{w}^\top \mathbf{x}|}{\|\mathbf{w}\|_2^2} \|\mathbf{w}\|_2 = \frac{|\mathbf{w}^\top \mathbf{x}|}{\|\mathbf{w}\|_2} \tag{1}$$

Note that if $\mathbf{w}$ is a unit vector (i.e., $\|\mathbf{w}\|_2 = 1$), then this distance simplifies to $|\mathbf{w}^\top \mathbf{x}|$. The margin $\gamma(\mathbf{w}, S)$ is defined as the minimum distance to any point in the dataset:

$$\gamma(\mathbf{w}, S) = \min_i \frac{|\mathbf{w}^\top \mathbf{x}_i|}{\|\mathbf{w}\|_2} \tag{2}$$

2 (Hard Margin) Support Vector Machines: Maximizing the Margin

Now that we've recapped our expression for the margin, and we have the intuition that we'd like a linear classifier with a *large* margin, we might try first using our nice numerical optimization tools to simply maximize the margin:

DANGER: Bad Idea Zone

$$\mathbf{w}_* = \max_{\mathbf{w}} \gamma(\mathbf{w}, S)$$

The problem with this is, of course, that it says nothing about actually *fitting the data!* Indeed, it is trivial to simply maximize the margin: just place the hyperplane as far away from all of your data as you can get away with. What we need to do is not just find a maximum margin hyperplane, but a maximum margin *separating hyperplane* (which we'll assume exists for the moment).

Recall that a point $(\mathbf{x}_i, y_i)$ is on the "right" side of the hyperplane if $y_i \mathbf{w}^\top \mathbf{x}_i > 0$. What we can do is modify the above optimization problem to be subject to the constraint that all training data points are on the "right" side of the hyperplane:

A Better Idea

$$\mathbf{w}_* = \max_{\mathbf{w}} \gamma(\mathbf{w}, S) \qquad \underbrace{\text{s.t. } \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0}_{\mathbf{w} \text{ needs to be a separating hyperplane}}$$

Pros of this optimization problem: it probably gives us a really good linear model! Cons of this optimization problem: it has constraints, so it doesn't immediately look like we can apply gradient descent.

The rest of what we need to do in this section is to fix that con. First, let's plug in the definition of the margin above to make things even worse for a moment:

$$\mathbf{w}_* = \max_{\mathbf{w}} \min_i \frac{|\mathbf{w}^\top \mathbf{x}_i|}{\|\mathbf{w}\|_2} \text{ s.t. } \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0$$

Let's play a few tricks to clean up this optimization problem. First of all, remember that the hyperplane is scale invariant: for any hyperplane parameterized by $\mathbf{w}$, $c\mathbf{w}$ for $c \neq 0$ parameterizes the same hyperplane. This is because $(c\mathbf{w}^\top)\mathbf{x} = 0 \iff \mathbf{w}^\top \mathbf{x} = 0$. So let's pick a *really* convenient scale for c. Given the optimal solution $\mathbf{w}_*$ to the above optimization problem let's find c such that $\min_i |(c\mathbf{w}_*^\top)\mathbf{x}_i| = 1$. Since $\mathbf{w}_*$ and $c\mathbf{w}_*$ define the same hyperplane, why not just search for this scaled version directly by adding this as a constraint so that we find $c\mathbf{w}_*$ instead of $\mathbf{w}_*$ in the first place?

$$\begin{aligned} \mathbf{w}_* = \max_{\mathbf{w}} \min_i \frac{|\mathbf{w}^\top \mathbf{x}_i|}{\|\mathbf{w}\|_2} \\ \text{s.t. } \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0 \\ \min_i |\mathbf{w}_*^\top \mathbf{x}_i| = 1 \end{aligned}$$

The claim is that the above optimization problem, which just has a new constraint, is equivalent to the first one. Why does this help us? Well, let's first pull a $\frac{1}{\|\mathbf{w}\|_2}$ out of the minimization over i since it doesn't depend on the data:

$$\begin{aligned} \mathbf{w}_* = \max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|_2} \min_i |\mathbf{w}^\top \mathbf{x}_i| \\ \text{s.t. } \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0 \\ \min_i |\mathbf{w}_*^\top \mathbf{x}_i| = 1 \end{aligned}$$

But at the minimum, we know by the *constraint* that $\min_i |\mathbf{w}^\top \mathbf{x}_i| = 1$, so we can drop it!

$$\begin{aligned} \mathbf{w}_* = \max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|_2} \\ \text{s.t. } \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0 \\ \min_i |\mathbf{w}_*^\top \mathbf{x}_i| = 1 \end{aligned}$$

Now, we'll make two observations. The first is simple, the second is complex and we'll prove it. First the maximization problem can be converted to a simpler minimization problem,

$$\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|_2} = \max_{\mathbf{w}} \frac{1}{\sqrt{\mathbf{w}^\top \mathbf{w}}} \iff \min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w}$$

This is because maximizing a fraction is the same as making the denominator as small as possible, and since $\sqrt{\cdot}$ is monotonic we can just drop it.

Next, the big part. the claim is that the following optimization problems are equivalent:

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } \begin{cases} \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0 \\ \min_i |\mathbf{w}_*^\top \mathbf{x}_i| = 1 \end{cases} \iff \min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$$

Let's be clear about what we mean by equivalent. We are claiming that the optimal solution $\mathbf{w}_*$ is the same for both optimization problems. Equivalently, $\mathbf{w}_*$ satisfies the left two constraints **if and only if** it satisfies the right constraint. Let's prove this.

Left implies right. Let's start by showing that if the left two constraints are satisfied by $\mathbf{w}_*$, then the right one is. The first constraint implies that $y_i \mathbf{w}^\top \mathbf{x}_i$ is positive for every i. Since $y_i \in \{+1, -1\}$, the multiplication by y_i only changes the sign of $\mathbf{w}^\top \mathbf{x}_i$, it cannot change the magnitude. Therefore,

$$y_i \mathbf{w}^\top \mathbf{x}_i = |\mathbf{w}^\top \mathbf{x}_i|$$

But by the second constraint on the left, we know that the *smallest* value of $|\mathbf{w}^\top \mathbf{x}_i|$ is 1. Therefore:

$$y_i \mathbf{w}^\top \mathbf{x}_i = |\mathbf{w}^\top \mathbf{x}_i| \geq 1$$

Thus, if the left two constraints are satisfied, the right constraint is also satisfied.

Right implies left. To finish the proof that these constraint sets are equivalent, we just need to show that the right constraint implies both of the left two constraints. First of all, it is obvious that the first constraint is implied:

$$y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 \implies y_i \mathbf{w}^\top \mathbf{x}_i > 0$$

Clearly if for every i the quantity is greater than 1, it's also greater than 0. Now we consider the second constraint. Suppose for contradiction that we indeed solved the right optimization problem:

$$\begin{aligned} \mathbf{w}_* = \min_{\mathbf{w}} \; & \mathbf{w}^\top \mathbf{w} \\ \text{s.t. } & \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 \end{aligned}$$

and found that the second left constraint was violated. In other words, we found that $\min_i |\mathbf{w}_*^\top \mathbf{x}_i| = c$ for $c \neq 1$. Clearly, c cannot be less than 1, otherwise we wouldn't have $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$. Therefore, $c > 1$.

Set $\mathbf{w}_{**} = \frac{\mathbf{w}_*}{c}$. I claim that $\mathbf{w}_{**}$ is at least a *feasible* solution to the optimization problem above. To see this, observe that, by definition,

$$y_i \mathbf{w}_{**}^\top \mathbf{x}_i = \frac{1}{c} y_i \mathbf{w}_*^\top \mathbf{x}_i$$

Since $y_i \mathbf{w}_*^\top \mathbf{x}_i$ and c are both positive, we certainly haven't changed the sign – the above is still positive. Therefore:

$$\frac{1}{c} y_i \mathbf{w}_*^\top \mathbf{x}_i = \frac{1}{c} |\mathbf{w}_*^\top \mathbf{x}_i|$$

But by definition of c, we know that $\min_i |\mathbf{w}_*^\top \mathbf{x}_i| = c$. Therefore,

$$\min_i \frac{1}{c} |\mathbf{w}_*^\top \mathbf{x}_i| = \frac{c}{c} = 1$$

Since the *smallest* value of $y_i \mathbf{w}_{**}^\top \mathbf{x}_i$ is 1, certainly for all i this quantity is greater than or equal to 1. Therefore, $\mathbf{w}_{**}$ is a feasible solution.

But what about the objective value?

$$\mathbf{w}_{**}^\top \mathbf{w}_{**} = \left(\frac{1}{c}\mathbf{w}_*\right)^\top \left(\frac{1}{c}\mathbf{w}_*\right) = \frac{1}{c^2} \mathbf{w}_*^\top \mathbf{w}_*$$

Since $c > 1$, this is **less than the objective value we achieved with $\mathbf{w}_*$**. This is a contradiction, because $\mathbf{w}_*$ was supposed to be the optimal solution!

Therefore, reiterating things, we do indeed have that:

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } \begin{cases} \forall i \, y_i \mathbf{w}^\top \mathbf{x}_i > 0 \\ \min_i |\mathbf{w}_*^\top \mathbf{x}_i| = 1 \end{cases} \iff \min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$$

This right optimization problem is exactly what we'll call the **hard margin SVM**:

The Hard Margin SVM

We call the linear model specified by $h(\mathbf{x}) = \mathbf{w}_*^\top \mathbf{x}$ parameterized by $\mathbf{w}_*$ a "hard margin" SVM if it is a solution to the optimization problem:

$$\mathbf{w}_* = \min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$$

2.1 Understanding Support Vectors

Let's see how the optimization constraint $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$ creates margin boundaries. For positive examples ($y_i = +1$), this constraint means $\mathbf{w}^\top \mathbf{x}_i \geq 1$. For negative examples ($y_i = -1$), it means $\mathbf{w}^\top \mathbf{x}_i \leq -1$.

This naturally creates three parallel regions in space: the decision boundary at $\mathbf{w}^\top \mathbf{x} = 0$, and two margin boundaries at $\mathbf{w}^\top \mathbf{x} = 1$ and $\mathbf{w}^\top \mathbf{x} = -1$. At the optimal solution, each point must either lie exactly on one of these margin boundaries (where $y_i \mathbf{w}^\top \mathbf{x}_i = 1$) or beyond their respective margin boundary (where $y_i \mathbf{w}^\top \mathbf{x}_i > 1$).

The points that lie exactly on the margin boundaries, where $y_i \mathbf{w}^\top \mathbf{x}_i = 1$, are called *support vectors*. These points have a remarkable property: they alone determine the optimal hyperplane. If we removed them, we could make the margin wider. All other points could be moved while maintaining their labels without affecting the solution.

This sparsity is what makes SVMs special: even though we used all the data to find the optimal hyperplane, in the end, only the support vectors matter. We could throw away all other points and get exactly the same classifier!

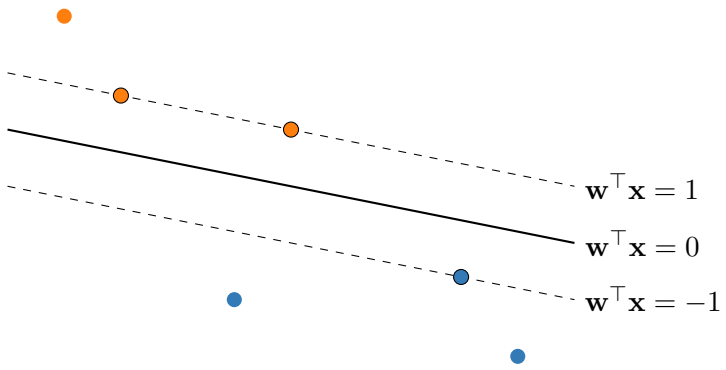

Figure 3: The support vectors (circled points) lie exactly on the margin boundaries where $y_i \mathbf{w}^\top \mathbf{x}_i = \pm 1$.

3 Soft margin Support Vector Machines

The formulation above has two problems:

1. It's still a constrained optimization problem. It's a really *nice* kind of constrained optimization problem called a quadratic program, so it's actually solvable, but we still don't like constraints.
2. The hard margin SVM only works if the data is linearly separable – otherwise, there is no feasible solution to the optimization problem.

In this section, we'll solve both of these problems with the same trick. To do this, we'll introduce the notion of **slack variables** to the constraint. In particular, we make the following observation: for *any* $\mathbf{w}$, if $\mathbf{w}$ is not a feasible solution above, it's because there's some i for which $y_i \mathbf{w}^\top \mathbf{x}_i$.

The idea of slack variables is to explicitly *allow* this kind of violation, but make the optimizer pay a penalty for it. The resulting optimization problem might look like this:

$$\mathbf{w}_* = \min_{\mathbf{w}, \boldsymbol{\xi}} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i$$

$$\text{s.t.} \quad \underbrace{y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 - \xi_i}_{\xi_i \text{is constraint violation for } \mathbf{x}_i}$$

$$\xi_i \geq 0$$

Here, ξ_i measures the amount by which the constraint for $(\mathbf{x}_i, y_i)$ is violated – e.g., $y_i \mathbf{w}^\top \mathbf{x}_i$ is less than 1. In English, what this new optimization problem does is allow each $\mathbf{x}_i$ to violate the constraint by some ξ_i, but we pay a penalty of $C\ \xi_i$ (where C is a constant chosen by you) in the objective. Our goal is to minimize the total penalty.

So far, we seem to have solved the "hard margin" problem, but not the constrained optimization problem. To achieve this final goal, first rearrange the above optimization problem:

$$\begin{aligned} \mathbf{w}_* = \min_{\mathbf{w},\boldsymbol{\xi}} \mathbf{w}^\top \mathbf{w} + C \sum_i \xi_i \\ \text{s.t. } \xi_i \geq 1 - y_i \mathbf{w}^\top \mathbf{x}_i \\ \xi_i \geq 0 \end{aligned}$$

observe that:

$$\xi_i = \begin{cases} 1 - y_i \mathbf{w}^\top \mathbf{x}_i & y_i \mathbf{w}^\top \mathbf{x}_i < 1 \\ 0 & y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 \end{cases}$$

Why is this true? Well, if $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$, clearly we can set $\xi_i = 0$. This satisfies both constraints, but adds the minimum possible additional loss to the objective. On the other hand, if $y_i \mathbf{w}^\top \mathbf{x}_i < 1$, the minimum value of ξ_i we can choose to satisfy the constraint that $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1 - \xi_i$ is just $1 - y_i \mathbf{w}^\top \mathbf{x}_i$, which we can see by rearranging the constraint itself.

Thus, for any choice of $\mathbf{w}$, instead of treating ξ_i as a variable to be optimized over, we can just compute it as above. A compact form for this is just:

$$\xi_i = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

To recap, we know that (1) at the optimum, ξ_i takes exactly the value above, and (2) this definition of ξ_i always satisfies both constraints. Thus, we can plug this definition of ξ_i into the loss and drop both constraints!

$$\mathbf{w}_* = \min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$

We're essentially done – the above is a perfectly fine optimization problem, and we could stop here. However, to make it look more like the optimization problems we've seen in class, let's do three "clean up" steps:

1. First, write $\mathbf{w}^\top \mathbf{w}$ as $\|\mathbf{w}\|_2^2$.
2. Second, since C is a free parameter that you will have to specify anyways, let's instead define $\lambda = \frac{1}{Cm}$ and multiply the whole equation by that.
3. Finally, let's swap the order of the terms.

The result is the following:

The Soft Margin SVM

We call the linear model specified by $h(\mathbf{x}) = \mathbf{w}_*^\top \mathbf{x}$ parameterized by $\mathbf{w}_*$ a "soft margin" SVM if $\mathbf{w}_*$ is a solution to the optimization problem:

$$\mathbf{w}_* = \min_{\mathbf{w}} \frac{1}{m} \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i) + \lambda \|\mathbf{w}\|_2^2$$

3.1 The Empirical Risk Minimization View

Part of the reason we did those "clean up" steps is that now the above minimization problem looks pretty familiar. We're minimizing the following function of $\mathbf{w}$:

$$\hat{R}(\mathbf{w}) = \frac{1}{m}\sum_i \underbrace{\max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)}_{\text{loss function}} + \underbrace{\lambda \|\mathbf{w}\|_2^2}_{\ell_2\text{-regularizer}}$$

Indeed, the soft margin SVM is just a plain old empirical risk minimization problem like linear or logistic regression, just with a particular loss function and an ℓ_2 regularizer!

We call this specific loss the *hinge loss*:

$$\ell_{\text{hinge}}(h(\mathbf{x}), y) = \max(0, 1 - y_i h(\mathbf{x}_i))$$

The hinge loss $\max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$ has an interesting property: it not only penalizes misclassification but also enforces a margin. Points correctly classified but within the margin ($0 < y_i h(\mathbf{x}_i) < 1$) still incur a penalty, pushing the decision boundary away from the data.

A soft margin SVM therefore just uses the hinge loss plus ℓ_2 regularization. Here's a plot comparing the $0-1$ loss, the hinge loss, and the logistic loss:

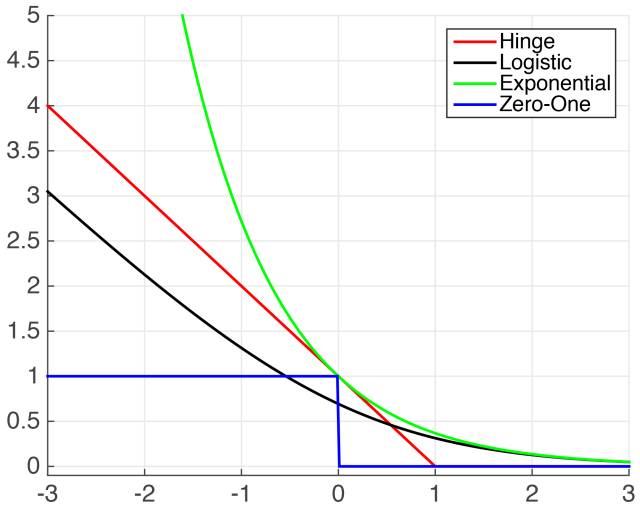

Figure 4: Three classification losses we've learned about so far (plus the exponential loss). These are plotted as a function of $y_i h(\mathbf{x}_i)$, because that quantity measures how "bad" the fit is – negative values mean we are classifying incorrectly.

By looking at these loss functions, we can start to get a better sense of the behavior of these algorithms. What do you think this will mean if there are outliers?