

# 1 Administrivia

- In HW 2, when I say “true concept”, what I mean is “plot the underlying sine curve”. The idea is that there’s a true process (the sine wave) and you’re measuring it with a noisy sensor (the  $\mathcal{N}(0, 1)$  term). You’re simply trying to reconstruct the underlying process as well as possible. You should end up with a bunch of plots that have the same sine wave and same noisy data points, but a variety of different estimates for the curve.

# 2 Linear Discriminator Functions, Cont’d: SVMs

- Recall that we formulated our maximum margin problem last time as: “find the  $\mathbf{W}$  that maximizes the minimum distance to any point in the data”, which we formalized like:
- We know that the (signed) distance from the hyperplane  $\mathbf{W}$  to a point  $X$  is given by

$$\text{dist}(X, \mathbf{W}) = \frac{\mathbf{W}^\top X}{\|\widehat{\mathbf{W}}\|}$$

- Therefore for any point in your data,  $X_i$ , with label  $y_i$ , we have that

$$d_i = y_i \frac{\mathbf{W}^\top X_i}{\|\widehat{\mathbf{W}}\|}$$

for gives the *absolute* distance of  $X_i$  to  $\mathbf{W}$ . (Note that signed distance is  $< 0$  only for  $y_i = -1$ .)

- Thus, for *separable* data, there exists some  $\mathbf{W}$  for which  $d_i = y_i \mathbf{W}^\top X_i > 0$  for all  $i$ .
- Let  $C$  be the minimal distance between hyperplane and data over the whole data set:

$$C = \min_i y_i \frac{\mathbf{W}^\top X_i}{\|\widehat{\mathbf{W}}\|}$$

- Now  $C$  is the measure of the margin we’re looking for – it measures the closest distance between *any* data point and the hyperplane. So we’d like to find the  $\mathbf{W}$  that *maximizes*  $C$ . Or, in other words,

$$\begin{aligned} & \max_{\mathbf{W}} C \\ & \text{Subject to } y_i \frac{\mathbf{W}^\top X_i}{\|\widehat{\mathbf{W}}\|} \geq C \quad \forall i \end{aligned}$$

- Which we can easily rewrite as:

$$\begin{aligned} & \max_{\mathbf{W}} C \\ & \text{Subject to } y_i \mathbf{W}^\top X_i \geq \|\widehat{\mathbf{W}}\| C \quad \forall i \end{aligned}$$

- Well... That's very nice, but how do you *solve* that for the  $\mathbf{W}$  that you want?
- For those who're familiar with linear programming/linear opt, you might recognize the form of the system that I wrote above. It looks like the general "maximize foo, subject to bar" form that we use for writing linear programs.
- It turns out that this system is actually a **quadratic program** (QP). (This isn't immediately obvious, but we'll get to it in a second.) For the moment, all you really need to know is: if you can write down the problem you want to solve as a QP, there're off-the-shelf software routines that you can just plug in that will solve it for you. It's pretty black-box, and I won't go into any more detail on how the QP solver works (*way, way* beyond the scope of this class). Suffice it to say that it's enough just to be able to write your problem as a QP.
- But our system still isn't in the right form for a QP. That's going to require something of the form:

$$\begin{array}{ll} \max_W & \text{Some function of } W^2 \\ \text{Subject to} & \text{A bunch of linear constraints} \end{array}$$

But what *we* have is something of the form

$$\begin{array}{ll} \max_W \min_i & \mathbf{W}^\top X_i \\ \text{Subject to} & \text{A bunch of linear constraints} \end{array}$$

- The trick is to notice that when the margin is maximized, then it exactly touches (at least) one point on each side of the hyperplane. Also, because rescaling  $\mathbf{W}$  doesn't change the hyperplane, we can pick a scale for  $W$  such that  $\mathbf{W}^\top X = \pm 1$  at the margins. This changes the "subject to" conditions we gave above to:

$$\text{Subject to } y_i \mathbf{W}^\top X_i \geq 1 \quad \forall i$$

- At those points exactly on the margin, we have that:

$$\begin{array}{ll} \widehat{\mathbf{W}}^\top X_i + w_0 = -1 & \text{for points "above" the plane} \\ \widehat{\mathbf{W}}^\top X_j + w_0 = -1 & \text{for points "below" the plane} \end{array}$$

The first eqn defines a plane  $(w_0 + 1)/\|\widehat{\mathbf{W}}\|$  units from the origin, the second defines a plane  $(w_0 - 1)/\|\widehat{\mathbf{W}}\|$  units from the origin. Subtracting, we find that the margin,  $C$  is  $2/\|\widehat{\mathbf{W}}\|$  units wide. Therefore, we can maximize the margin simply by maximizing  $2/\|\widehat{\mathbf{W}}\|$ , or, equivalently, minimizing  $\widehat{\mathbf{W}}^\top \widehat{\mathbf{W}}$ .

- Thus, we can write our final form as:

$$\begin{array}{ll} \min_{\widehat{\mathbf{W}}} & \widehat{\mathbf{W}}^\top \widehat{\mathbf{W}} \\ \text{subject to} & y_i \mathbf{W}^\top X_i \geq 1 \quad \forall i \end{array}$$

- The only other thing you need to know is that an off the shelf QP solver will take time that's *roughly* cubic in the number of *constraints* (i.e., “subject to”) terms in the system, *not* in the dimension. I.e., the solver doesn't care how many dimensions the data lives in, only how many data points there are. This is distinctly different from the min squared error system.
- So we have an algorithm for finding the MMH (more precisely, a meta-algorithm: write the problem down in this way, then run a QP solver on it.) That's semi-satisfying, but we're still not as cool as the MSE approach that we talked about last time – we can't handle non-separable data, for example.
- We can extend the system to handle linearly inseparable data in two ways. The first is to allow the hyperplane to make some errors. This involves changing our “subject to” criteria from

$$\text{Subject to } y_i \mathbf{W}^\top X_i \geq 1 \quad \forall i$$

to

$$\text{Subject to } y_i \mathbf{W}^\top X_i \geq 1 - \xi_i \quad \forall i$$

which says, essentially, that we allow point  $X_i$  to be on the “wrong” side of the margin by up to  $\xi_i$  units. The new, synthetic variables we have introduced are called the “slack variables” (because they add some slack in your choice of solutions).

- Clearly, we have to prevent it from being *arbitrarily* wrong – if the  $\xi_i$  could be infinitely large, then the hyperplane is completely unconstrained. So we have to add a penalty term to the objective function to force  $\xi_i$  to be as small as possible. So our *new* optimization problem is:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \widehat{\mathbf{W}}^\top \widehat{\mathbf{W}} + D \sum_i \xi_i \\ \text{subject to} \quad & y_i \mathbf{W}^\top X_i \geq 1 \quad \forall i \end{aligned}$$

Now the system is encouraged to find a  $\mathbf{W}$  that also keeps the  $\xi_i$  small.  $D$  is just a user-set parameter that says how important the slack variables are relative to  $\|\widehat{\mathbf{W}}\|$ , i.e., how harshly you should penalize classification errors.

- Now for the mindbending part. . .
- We still want to be able to handle *nonlinear* systems. It turns out that if you're a little bit clever, you can get there too.
- I *will not* prove it to you in this class, but it's possible to rewrite the previous system with some variable transformations as:

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i^\top X_j \\ \text{Subject to} \quad & 0 \leq \alpha_i \leq D \\ & \sum_i \alpha_i y_i = 0 \end{aligned}$$

(No, it's not obvious, and I told you I wouldn't prove it to you. Go look up the Burges tutorial on SVMs if you really want to understand where that came from...)

- The important thing about that expression is that it's written in terms of a *dot product* between the  $X_i$ . That is, if you know the value of the dot product, *you don't actually need to know what the  $X_i$  themselves are!*
- Why is this useful? Well, suppose that we take our *original*  $X_i$ , which live in  $\mathbb{R}^d$ , and project them into some higher dimensional space,  $\mathbb{R}^k$  via a *nonlinear* transform  $\Phi(X)$  ( $k \gg d$ ).
- It may be that  $k$  is so large that it's painful to manipulate (we have examples of  $k = \infty$ )
- But suppose that we have a convenient way to manipulate the product  $K(X_i, X_j) = \Phi(X_i)\Phi(X_j)$ . Now we could just plug in  $K$  in the place of the dot product, above, and carry through the same solution. It turns out to work just fine.
- Which begs the question of, if you can't handle  $\Phi$ , how can you get  $K$ ?
- Well, often  $K$  is easy to represent, even if  $\Phi$  is hard. E.g.,

$$K(X_i, X_j) = e^{-\frac{(X_i - X_j)^\top (X_i - X_j)}{2\sigma^2}}$$

which you should recognize as, essentially, being a Gaussian. That's a  $K$  equivalent to an infinite-dimensional  $\Phi$  (again, I won't prove this to you). Another example is:

$$K(X_i, X_j) = (X_i^\top X_j)^p$$

for some integer  $p > 1$ . It turns out that if your original  $X_i \in \mathbb{R}^d$ , then this corresponds to a  $\Phi$  that lives in a space of dimension  $\binom{d+p-1}{p}$ . If you're looking at, say,  $16 \times 16$  images ( $d = 256$ ), and  $p = 4$  (you're looking at a degree-4 polynomial expansion), then  $\Phi$  is a 183,181,376-dimensional space. Wow!