
CS 181 LECTURE 2/27/24

Scribe Notes

Contents

1	SVM (II)	3
1.1	Real-World Impact	3
1.2	Last Time: Max Margin	3
1.3	Dual Optimization Problem	3
1.4	Solve the Dual Problem	4
1.5	Intuition:	5
1.5.1	Let's Talk About Kernels	5
1.6	Concept Check	7
1.7	Concept Check Answers	8

1 SVM (II)

This will be covered on the midterm. Next lecture's content will not.

1.1 Real-World Impact

Supervised learning is the practice of prediction, specifically predicting y 's. There are two types: classification and regression. These powerful predictions can fool us. This happens because the model is so powerful that it can fit noise.

Zhang '17: Randomized labels to break the relationship between image and labels. After running the model, would still get to zero training loss. Thus, this paper shows that models can get zero loss on almost anything, even when there is a completely bogus relationship between x and y 's.

Another cause for problems is spurious correlations. There is a paper that claimed to predict sexual orientation based on profile photos. But the model was focusing on superficial features like facial hair and amount of makeup. So, rather than finding an $x \rightarrow y$ relationship, a model may find a $x \rightarrow z \rightarrow y$ relationship.

We may also alter the environment. For example, in policing. When you detect more crime in an area, then you deploy more police, but that leads to higher reported crime rates. This negative cycle can happen when you don't think about the actions of using a model.

The takeaway is that just because you can deploy a model does not mean you necessarily should.

1.2 Last Time: Max Margin

The key idea is to maximize the margin. We talked about solving using convex optimization.

We have the hard margin and soft margin.

Hard Margin:

$$\text{minimize } \frac{1}{2} \|w\|^2 \text{ subject to } y_n(w^\top \cdot x_n + w_0) \geq 1$$

Soft Margin:

$$\text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n \text{ subject to } y_n(w^\top \cdot x_n + w_0) \geq 1 - \xi_n, \quad \xi_n \geq 0$$

Solving this problem is slow when the dimensions are high.

1.3 Dual Optimization Problem

We will use Lagrangian to solve the optimization problem.

We introduce the Lagrangian variables α_n .

$$\min_{w, w_0} [\max_{\alpha} [\frac{1}{2} \|w\|_2^2 - \sum_{n=1}^w \alpha_n (y_n (w^\top x_n + w_0) - 1)]]$$

s.t. $\alpha_n \geq 0$

Let $z_n = y_n(w^\top x_n + w_0)$.

The intuition is that the maximizing over α helps keep the constraint.

1) If $z_n - 1 < 0$, then the objective value can go to infinity as α_n will go to infinity. But we have the minimization, so this is implicitly enforcing that we will not let $z_n - 1 < 0$ happen.

2) If met with slack, then $z_n - 1 > 0 \rightarrow \alpha_n = 0$.

3) $\alpha_n \neq 0$ if $z_n - 1 = 0$ which means that we are on the margin.

4) The 2) and 3) points imply **complementary slackness**: for all n , either $\alpha_n = 0$ or $y_n(w^\top x_n + w_0) = 1$.

1.4 Solve the Dual Problem

Tip: Use the strong duality property! Strong duality lets us flip the min and the max.

$$\max_{\alpha} [\min_{w, w_0} [\frac{1}{2} \|w\|_2^2 - \sum_{n=1}^N \alpha_n (y_n(w^\top x_n + w_0) - 1)]]$$

s.t. $\alpha_n \geq 0$

(You can think about it in game theory terms - outcome doesn't switch when you swap order of the players.)

Now, we take partial derivative of $\mathcal{L}$ with respect to w and set it equal to zero.

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{n=1}^N \alpha_n y_n x_n = 0$$

$$w^* = \sum_{n=1}^N \alpha_n y_n x_n$$

We plug in our optimal weights.

$$\mathcal{L}(w^*) = \frac{1}{2} \sum_{n=1}^N (\alpha_n y_n x_n)^\top (\alpha_n y_n x_n) - \sum_{n=1}^N \alpha_n (y_n(w^\top x_n + w_0) - 1)$$

$$\mathcal{L}(w^*) = \frac{1}{2} \sum_{n, n'} \alpha_n \alpha_{n'} + y_n y_{n'} x_n^\top x_{n'} + \sum \alpha_n \text{ s.t. } \sum \alpha_n y_n = 0, \alpha_n \geq 0$$

A couple of observations:

1) Our problem is still convex. We have convex optimization solvers that we can use here.

2) The α 's have to be between zero and c .

3) What is the dimension of alpha?

Now, it is an optimization problem of size N . Duality changes the size of your problem, which you can use to your advantage. For example, if we have a huge D dimensions, we would rather solve a problem with a smaller N .

1.5 Intuition:

$$w^* = \sum_{n=1}^N \alpha_n y_n x_n$$

Then, the prediction will look like

$$\hat{y} = \sum \alpha_n y_n x_n x + w_0$$

$x_n x$ is a measure of similarity y_n is a vote (the label) α_n is a weight on how important X_n is for x .

The effect of x_n and x only happens in the inner product. We never need to store these vectors, x_n and x . If we use basis function $\phi(x)$, we don't need to store the whole thing, we only need to know $\phi(x)^\top \phi(x')$ for every x and x' . We can create a kernel function, $K(x, x') = \phi(x)^\top \phi(x')$, then we only need the kernel and don't need to keep track of the basis. We can get away with storing the scalar values.

Kernel trick: replace inner product with a kernel! This is nice as long as $K(x, x')$ can be computed efficiently.

1.5.1 Let's Talk About Kernels

We like kernels because, just like phi, they let us represent data in a way that is easy to classify the data.

$\phi(x)$	$K(x, x')$
x	$x^\top x'$
$[x_1^2, x_1 x_2, x_1 x_3, \dots, x_n^2]$	$\sum_{d, d'}^D x_d x_{d'} x'_d x'_{d'}$
$// \exp(q)$	$K(x, x') = (1 + x^\top x)^q$
infinitely large ϕ	$K(x, x') = \exp(\frac{-\ x - x'\ ^2}{2\sigma^2})$

Kernels give us a very simple way to compute very powerful relationship between data-points.

Mercer's Theorem: If you consider the matrix K with entries $K_{ij} = K(x_i, x_j)$ and $z^\top K z \geq 0$ for all z , then K is a valid kernel. In simpler terms, the Mercer's theorem guarantees that if we have a valid kernel function $K(x, x')$, then there exists a corresponding feature space and feature mapping such that $K(x, x')$ can be expressed as the inner product of the feature vectors in that space.

Main takeaway: Kernels let use compute very expressive functions efficiently.

It's often more intuitive to represent similarity through a kernel than to create a basis function. In different fields, there already exist notions of similarity, ex. in biology, computer vision.

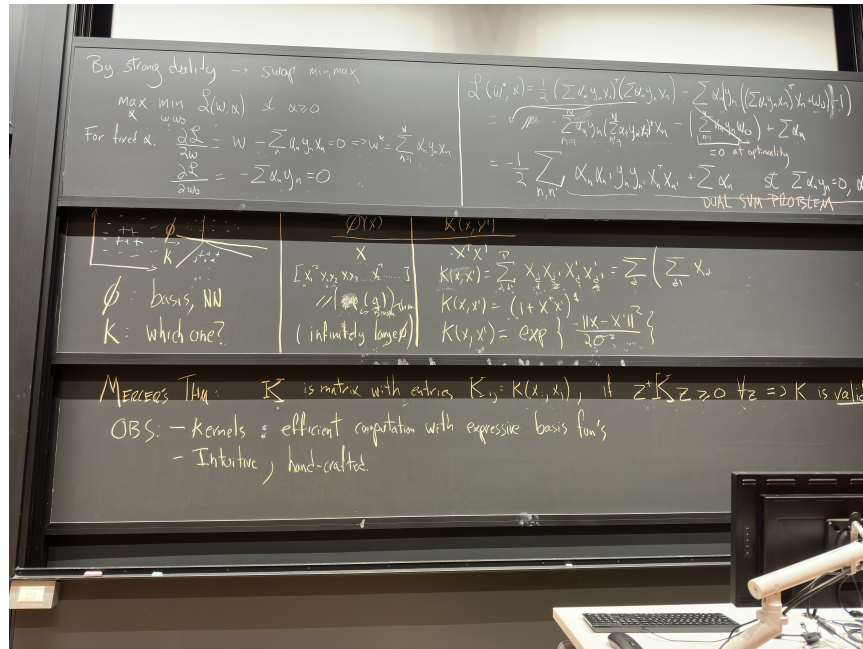


Figure 1: Pictures of board.

1.6 Concept Check

What does overfitting mean for SVM's? Can SVM's overfit?

- 1) What would happen if we use the RBF kernel with a small σ^2 ? (in terms of over and underfitting)
- 2) What would happen if we use the RBF kernel with a large σ^2 ? (in terms of over and underfitting)
- 3) How do we know if an SVM is overfitting?

Solutions on the next page.

1.7 Concept Check Answers

What does overfitting mean for SVM's? Can SVM's overfit?

1) What would happen if we use the RBF kernel with a small σ^2 ? (in terms of over and underfitting)

Only very close things are similar. In this case, we would have overfitting. We only consider to points that are very similar and this forces α_n 's that are nonzero. Every point will have a support point. So we end up with a very jagged line in our model.

2) What would happen if we use the RBF kernel with a large σ^2 ? (in terms of over and underfitting)

Now, we have fewer support points and larger margin and a flatter decision boundary. So, here we have underfitting.

3) How do we know if an SVM is overfitting?

Signs of overfitting:

- lots of support points
- lots of α_n that are not zero
- smaller margins

Cross validation can be used to check if we have overfitting.