Lecture 12: Support Vector Machines
Support Vector Machines

Assume labels $y_i \in \{-1, +1\}$

Then $\|y - Xw\|_2^2 = \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} (1 - y_i x_i^T w)^2$

If this is $\geq 1$, then we incur loss even if point is correctly classified.

Ex: predict whether someone plays basketball based on height.

$y_i = +1 \implies$ plays bball
$y_i = -1 \implies$ doesn't play bball

4 training samples:

$X_i = \begin{bmatrix} 70 & 71' & 73' & 82' \end{bmatrix}$
$Y_i = \begin{bmatrix} -1 & -1 & +1 & +1 \end{bmatrix}$

\[
\hat{w}_L = (X^T X)^{-1} X^T Y = 0.15 \implies \hat{y} = +1 \text{ if } (\text{height} - \text{mean}) > 0.15
\]

LS decision boundary misclassifies a point even though perfect classifier exists.
Loss functions

- O(1) loss: \( I \{ y \neq \text{sign}(x^T w) \} = I \{ y x^T w < 0 \} \) = "ideal loss."
- Quadratic loss: \( (1 - y x^T w)^2 \)
- Hinge loss: \( (1 - y x^T w)_+ \)

Hinge loss mimics the ideal loss but is convex and easy to minimize.

\[
\ell(w) = \sum_{i=1}^{n} (1 - y_i x_i^T w)_+
\]

\[
\nabla_w \ell = \sum_{i=1}^{n} I \{ y_i x_i^T w < 1 \} (y_i x_i)
\]

This is technically called "subgradient" because \( \ell(w) \) not differentiable.

\[\text{If we minimize } \sum_{i=1}^{n} (1 - y_i x_i^T w)_+ + \lambda \|w\|_2^2 \]

or the kernel version \[\sum_{i=1}^{n} (1 - y_i \phi(x_i)^T w), + \lambda \|w\|_2^2\]

this is called a support vector machine.
Let $\hat{w}_{svm} = \text{argmin} \sum_{i=1}^{n} (1 - y_i x_i^T w)_+ + \lambda w_i^2$ or $\hat{w}_{svm} = \text{argmin} \sum_{i=1}^{n} (1 - y_i \phi(x_i)^T w)_+ + \lambda w_i^2$.

Just like with kernel ridge regression, it is possible to show $\hat{w}_{svm} = X^T \alpha$ for some $\alpha$ (different from least squares $\alpha$) or $\hat{w}_{svm} = \Phi^T \alpha$.

To see this:

Imagine $\hat{w} = X^T \beta + w^d = \sum_{j=1}^{n} \alpha_j x_j + w^d$ (where $w^d$ is some vector orthogonal to the $x_j$'s).

Then $\min_{\alpha, w^d} \sum_{i=1}^{n} (1 - y_i x_i^T \left( \sum_{j=1}^{n} \alpha_j x_j + w^d \right))_+ + \lambda \left\| \sum_{j=1}^{n} \alpha_j x_j \right\|_2^2$

$= \min_{\alpha, w^d} \sum_{i=1}^{n} (1 - y_i \left[ \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle + x_i^T w^d \right])_+ + \lambda \left[ \left\| \sum_{j=1}^{n} \alpha_j x_j \right\|_2^2 + \left\| w^d \right\|_2^2 \right]$  

This is always zero.

$= \min_{\alpha, w^d} \sum_{i=1}^{n} (1 - y_i \left[ \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle \right])_+ + \lambda \left[ \left\| \sum_{j=1}^{n} \alpha_j x_j \right\|_2^2 + \left\| w^d \right\|_2^2 \right]$  

This must be 0 at optimum!

Alternatively, we can apply the 'kernel trick' and replace inner products with kernels:

$\hat{\alpha} = \text{argmin} \sum_{i=1}^{n} (1 - y_i \sum_{j=1}^{n} \alpha_j K(x_i, x_j))_+ + \lambda \sum_{j=1}^{n} \alpha_j \sum_{i=1}^{n} \alpha_j K(x_i, x_j)$

There is no closed form solution to this optimization problem.  
$\Rightarrow$ need to use gradient descent or other numerical optimization methods.
Why is this called a "support vector machine"?

Typically, $\mathbf{\alpha}$ is sparse—most $\alpha_i = 0$

Recall $\mathbf{\hat{w}} = \sum_{j=1}^{\hat{N}} \alpha_j \phi(x_j)$

$\Rightarrow \mathbf{\hat{w}}$ is a linear combination of only a few training samples (in feature space)

those $x_i$'s are called support vectors.
\[ \lambda = 10^{-4} \]

Training Error: 0.270  
Test Error: 0.288  
Bayes Error: 0.210

\[ \lambda = 100 \]

Training Error: 0.26  
Test Error: 0.30  
Bayes Error: 0.21

---

ideal (Bayes) decision boundary — gives smallest test errors on average

points \( x \) where \( \hat{y} = x^T \hat{w}_{\text{Bayes}} \pm 1 \)

SVM decision boundary
How do kernels help?

no good linear classifier exists

\[ X_i = \begin{bmatrix} X_{i1} \\ X_{i2} \end{bmatrix} \]

in high-dimensional feature space, a good linear separating hyperplane exists

\[ \phi(X_i) = \begin{bmatrix} X_{i1} \\ X_{i2} \\ X_{i1}^2 + X_{i2}^2 \end{bmatrix} \]
SVM - Degree-4 Polynomial in Feature Space

Training Error: 0.180
Test Error: 0.245
Bayes Error: 0.210

SVM - Radial Kernel in Feature Space

Training Error: 0.160
Test Error: 0.218
Bayes Error: 0.210

\[ K(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]

--- Ideal ('Bayes') decision boundary — gives smallest test errors on average
--- points \( \hat{y} = x^T \hat{w}_{Bayes} \pm k \)
--- SVM decision boundary