Lecture 26: Kernels and the SVM

Parametric methods
- e.g. fit polynomial to data; use training data to learn parameters (poly. coefficients)

Nonparametric methods
- e.g. $y = f(x)$ is "smooth"

$Y_{test} = \text{weighted sum } y_i$'s.
Assign bigger weight to $y_i$ if $x_i$ is close to $X_{test}$
Recall: Kernel function measures similarity or alignment of two feature vectors.

e.g. Gaussian kernel

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

Nadaraya-Watson Kernel Regression

\[ \hat{y}_{\text{test}} = \frac{\sum_{i=1}^{n} y_i K(x_i, x_{\text{test}})}{\sum_{j=1}^{n} K(x_j, x_{\text{test}})} \]
\[ \hat{y}_{test} = \sum_i \alpha_i K(x_i, x_{iw}) \]

\[ \alpha = (K + \lambda I)^{-1} y \]

equivalent to ridge regression in a high-dimensional feature space

where

\[ K(u, v) = \langle \phi(u), \phi(v) \rangle \]

\( \phi \) high dimensional feature vectors.
Kernels and Support Vector Machines for classification

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

First, consider regularized least squares:

$$
\hat{w} = \arg \min_w \sum_{i=1}^{n} \left(1 - y_i x_i^T w\right)^2 + \lambda \|w\|_2^2
$$

Last time, showed $\hat{w} = X^T \alpha$ for some $\alpha$

$$
= \sum_{i=1}^{n} \alpha_i x_i
$$

$\Rightarrow \hat{\alpha} = \arg \min_{\alpha} \sum_{i=1}^{n} \left(1 - y_i x_i^T \left(\sum_{j=1}^{n} \alpha_j x_j\right)\right)^2 + \lambda \|\sum_{j=1}^{n} \alpha_j x_j\|_2^2$

$$
= \arg \min_{\alpha} \sum_{i=1}^{n} \left(1 - y_i \sum_{j=1}^{n} \alpha_j <x_i, x_j>\right)^2 + \lambda \sum_{j=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j <x_i, x_j>
$$
"Kernel trick" — replace inner products \( \langle x_i, x_j \rangle \) with the kernel function \( K(x_i, x_j) \)

\[
\Rightarrow \hat{\alpha} = \arg\min_{\alpha} \sum_{i=1}^{n} (1 - y_i \sum_{j=1}^{n} \alpha_j K(x_i, x_j))^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j)
\]

\[
\hat{\alpha} = (K + \lambda \mathbf{I})^{-1} y.
\]

Now consider hinge loss:

\[
\hat{w} = \arg\min_{w} \sum_{i=1}^{n} (1 - y_i x_i^T w)^+ + \lambda \|w\|_2^2
\]

Show \( \hat{w} = X^T \hat{\alpha} = \sum_{j=1}^{n} \alpha_j x_j \) (like before, but different \( \alpha \)’s)
Imagine: \( \hat{w} = X^T \alpha + x^\perp \) (\( x^\perp \) some vector orthogonal to the \( x_i \)'s)

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

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

\[
\Rightarrow x^\perp = 0
\]

\[
\hat{\alpha} = \text{arg min}_{\alpha} \sum_{i=1}^{n} \left( 1 - y_i \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle \right)_+ + \lambda \left\| \sum_{j=1}^{n} \alpha_j x_j \right\|_2^2
\]

\[
= \sum_{j} \sum_{i} \alpha_i \alpha_j \langle x_i, x_j \rangle
\]
Apply Kernel trick:
\[
\hat{\alpha} = \arg\min_{\alpha} \sum_{i=1}^{n} (1 - y_i \sum_{j} \alpha_j K(x_i, x_j)) + \lambda \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j)
\]

no closed-form expression for \( \hat{\alpha} \)
find \( \hat{\alpha} \) using optimization (GD)

Why is this called a "support vector machine"?
\( \hat{\alpha} \) is sparse — most \( \hat{\alpha}_j = 0 \)

recall \( \hat{w} = \sum_{j=1}^{n} \hat{\alpha}_j x_j \)
\( \Rightarrow \hat{w} = \text{lin. comb} \text{ of only a few training } x_i \text{'s} \)
these \( x_i \text{'s} \) are called "support vectors"
far from decision boundary AND correctly classified
⇒ does not affect hinge loss
⇒ receive zero weight in opt solution (corresponding $\alpha = 0$)
At start of semester:
- no background in linear alg.
- no background in ML

Now
- classification (face emotions, integers)
- PCA (dimension reduction)
- matrix completion (recommender system)
- Page Rank
- optimization, neural networks
- kernels + SVM