Kernels Continued

Cornell CS 4/5780 Spring 2023

Instructor

April 11, 2023

1. Prove that the solution lies in the span of the training points (i.e. $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$ for some α_i)
2. Rewrite the algorithm and the classifier so that all training or testing inputs $\mathbf{x}_i$ are only accessed in inner-products with other inputs, e.g. $\mathbf{x}_i^\top \mathbf{x}_j$
3. Define a kernel function and substitute $k(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i^\top \mathbf{x}_j$

Well-defined Kernels

Wanna kernel? Gotta make sure it's well-defined!

Any positive semi-definite matrix is a well-defined kernel.

Definition: Positive Semi-Definite Matrices

A kernel matrix is positive-semidefinite is equivalent to any of the following statements:

1. All eigenvalues of K are non-negative.
2. There exists a real matrix P such that $K = P^\top P$.
3. For all real vectors $\mathbf{x}$, $\mathbf{x}^\top K \mathbf{x} \geq 0$.

Common kernels:

- *Linear*: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$
- *RBF*: $k(\mathbf{x}, \mathbf{z}) = e^{-\frac{(\mathbf{x}-\mathbf{z})^2}{\sigma^2}}$
- *Polynomial*: $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^d$

Positive Semidefinite Kernel Matrices

Who said positive is always a good thing? Just kidding, it is!

We can construct new kernels by recursively combining one or more rules from the following list:

1. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$
2. $k(\mathbf{x}, \mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$
4. $k(\mathbf{x}, \mathbf{z}) = g(k(\mathbf{x}, \mathbf{z}))$
5. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$
6. $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$
7. $k(\mathbf{x}, \mathbf{z}) = e^{k_1(\mathbf{x}, \mathbf{z})}$
8. $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{A} \mathbf{z}$

where $c \geq 0$ and $g()$ is a polynomial with positive coefficients.

Quiz Time!

Time to put your thinking caps on!

Quiz 1

Prove that the RBF kernel $k(\mathbf{x}, \mathbf{z}) = e^{\frac{-(\mathbf{x}-\mathbf{z})^2}{\sigma^2}}$ is a well-defined kernel matrix.

Quiz 2

Prove that the following kernel, defined on any two sets $S_1, S_2 \subseteq \Omega$, is well-defined: $k(S_1, S_2) = e^{|S_1 \cap S_2|}$.

Recap: Vanilla OLS regression minimizes the following squared loss regression loss function:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2,$$

to find the hyper-plane $\mathbf{w}$. The prediction at a test-point is simply $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$.
If we let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $\mathbf{y} = [y_1, \ldots, y_n]^\top$, the solution of OLS can be written in closed form:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}$$

To kernelize the algorithm, we express the solution $\mathbf{w}$ as a linear combination of the training inputs:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = \mathbf{X}\vec{\alpha}.$$

During testing, a test point is only accessed through inner-products with training inputs:

$$h(\mathbf{z}) = \mathbf{w}^\top \mathbf{z} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{z}.$$

We can now immediately kernelize the algorithm by substituting $k(\mathbf{x}_i, \mathbf{z})$ for any inner-product $\mathbf{x}_i^\top \mathbf{z}$.

Theorem

Kernelized ordinary least squares has the solution $\vec{\alpha} = \mathbf{K}^{-1}\mathbf{y}$.

Proof.

$$
\begin{aligned}
\mathbf{X}\vec{\alpha} &= \mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} \\
(\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top)\mathbf{X}\vec{\alpha} &= (\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top)(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} \\
(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})\vec{\alpha} &= \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} \\
\mathbf{K}^2\vec{\alpha} &= \mathbf{K}\mathbf{y} \\
\vec{\alpha} &= \mathbf{K}^{-1}\mathbf{y}
\end{aligned}
$$

□

Kernel SVM

Kernelize your SVMs for more power and fun!

(Original) SVM Primal Form

$$\min_{\xi_i \geq 0, \mathbf{w}, b} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } \forall i, \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$$

SVM Dual Form

$$\min_{\alpha_1, \cdots, \alpha_n} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} - \sum_{i=1}^{n} \alpha_i$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Where $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)$ and the decision function is:

$$h(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right).$$

Question: What is the dual form of the hard-margin SVM?

- Support vectors: only support vectors satisfy the constraint with equality: $y_i(\mathbf{w}^\top \phi(x_i) + b) = 1$. In the dual, these are the training inputs with $\alpha_i > 0$.
- Recovering b: we can solve for b from the support vectors using:

$$
\begin{aligned}
y_i(\mathbf{w}^\top \phi(x_i) + b) &= 1 \\
y_i \left(\sum_j y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) + b \right) &= 1 \\
\sum_j y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) + b &= y_i \\
y_i - \sum_j y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) &= b
\end{aligned}
$$

Kernel SVM - The Smart Nearest Neighbor

Because who wants a dumb nearest neighbor?

KNN for binary classification problems

$$h(\mathbf{z}) = \mathrm{sign}\left(\sum_{i=1}^{n} y_i \delta^{nn}(\mathbf{x}_i, \mathbf{z})\right),$$

where $\delta^{nn}(\mathbf{z}, \mathbf{x}_i) \in \{0, 1\}$ with $\delta^{nn}(\mathbf{z}, \mathbf{x}_i) = 1$ only if $\mathbf{x}_i$ is one of the k nearest neighbors of test point $\mathbf{z}$.

SVM decision function

$$h(\mathbf{z}) = \mathrm{sign}\left(\sum_{i=1}^{n} y_i \alpha_i k(\mathbf{x}_i, \mathbf{z}) + b\right)$$

Kernel SVM is like a smart nearest neighbor: it considers *all* training points but kernel function assigns more weight to closer points. It also learns a weight $\alpha_i > 0$ for each training point and a bias b, and sets many $\alpha_i = 0$ for useless training points.

Thank You!

I hope you had as much fun as I did!

That's all folks! Have a great day and keep kernelizing!