Statistical Machine Learning — 15 — Kernel Method — October 20, 2015

Lecturer: Prof. Homrighausen — Scribe: Zeke Wang

Kernel methods

Intuition: Many methods have linear decision boundaries. We know that sometimes this isn't sufficient to represent data. For example, sometimes we need to included a polynomial effect or a log transform in multiple regression.And sometimes, a linear boundary, but in a different space makes all the difference.So we should think about another method to deal with these kinds of problems.

Optimal separating hyperplane

Here is a reminder: The Wolfe dual, which gets maximized over α, produces the optimal separating hyperplane

$$\text{Wolf dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k Y_i Y_k X_i^\top X_k$$

$\alpha_i \geq 0$

A similar result holds after the introduction of slack variables

It's very important here that the features only enter via

$$X^\top X' = \langle X, X' \rangle$$

(Kernel) ridge regression

Suppose we want to predict at X, then

$$\hat{f}(X) = X^\top \lambda = X^{\top\top}(^\top + \lambda I)^{-1} Y$$

Also,

$$^\top = \begin{bmatrix} \langle X_1, X_1 \rangle & \langle X_1, X_2 \rangle & \cdots & \langle X_1, X_n \rangle \\ & \vdots & & \\ \langle X_n, X_1 \rangle & \langle X_n, X_2 \rangle & \cdots & \langle X_n, X_n \rangle \end{bmatrix}$$

and

$$X^{\top\top} = [\langle X, X_1 \rangle, \langle X, X_2 \rangle, \cdots, \langle X, X_n \rangle]$$

Again, we have the covariates enter only as

$$\langle X, X' \rangle = X^\top X'$$

Conclusion: A Linear rule in a transformed space can have a nonlinear boundary in the original features.

Logistic regression: transformations

Reminder: The logistic model: untransformed

$$\begin{aligned}\text{logit}(¶(Y=1|X)) &= \beta_0 + \beta^\top X \\ &= \beta_0 + \beta_1\text{balance} + \beta_2\text{income}\end{aligned}$$

The decision boundary is the hyperplane $\{X : \beta_0 + \beta^\top X = 0\}$. This is linear in the feature space.

Adding the polynomial transformation $\Phi(X) = (x_1, x_2, x_2^2)$:

$$\begin{aligned}\text{logit}(¶(Y=1|X)) &= \beta_0 + \beta^\top \Phi(X) \\ &= \beta_0 + \beta_1\text{balance} + \beta_2\text{income} + \beta_3\text{income}^2\end{aligned}$$

Decision boundary is still a hyperplane $\{X : \beta_0 + \beta^\top \Phi(X) = 0\}$. This is nonlinear in the feature space!

Of course, as we include more transformations, we need to choose the transformations manually, and also Computations can become difficult if we aren't careful. We need to regularize to prevent overfitting.

Kernel Methods

Nonnegative definite matrices

Let $A \in^{p\times p}$ be a symmetric, nonnegative definite matrix:

$$z^\top Az \geq 0 \text{ for all } z \text{ and } A^\top = A$$

Then, A has an eigenvalue expansion

$$A = UDU^\top = \sum_{j=1}^{p} d_j u_j u_j^\top$$

where $d_j \geq 0$

Each such A, generates a new inner product

$$\langle z, z' \rangle = z^\top z' = z^\top \underbrace{I}_{\text{Identity}} z'$$

$$\langle z, z' \rangle_A = z^\top A z'$$

If we enforce A to be positive definite, then $\langle z, z \rangle_A = ||z||_A^2$ is a norm.

Suppose A_i^j is the (i, j) entry in A, and A_i is the i^{th} row

$$Az = \begin{bmatrix} A_1^\top \\ \vdots \\ A_p^\top \end{bmatrix} z = \begin{bmatrix} A_1^\top z \\ \vdots \\ A_p^\top z \end{bmatrix}$$

Note: Multiplication by A is really taking inner products with its rows. Hence, A_i is called the (multiplication) kernel of matrix A

Kernel methods

$k : \mathcal{X} \times \mathcal{X} \to$ is a symmetric, nonnegative definite kernel.

Write the eigenvalue expansion of k as

$$k(X, X') = \sum_{j=1}^{\infty} \theta_j \phi_j(X) \phi_j(X')$$

with

- $\theta_j \geq 0$ nonnegative definite
- $(\theta_j)_{j=1}^{\infty}{}_2 = \sum_{j=1}^{\infty} \theta_j^2 < \infty$
- The ϕ_j are orthogonal eigenfunctions: $\int \phi_j \phi_{j'} = \delta_{j,j'}$

This is called Mercer's theorem, and such a k is called a Mercer kernel.

Kernel: Example

Back to polynomial terms/interactions:

Form

$$k_d(X, X') = (X^\top X' + 1)^d$$

k_d has $M = \binom{p+d}{d}$ eigenfunctions

These span the space of polynomials in ^{p} with degree d

There's another example. Let $d = p = 2 \Rightarrow M = 6$ and

$$\begin{aligned} k(u, v) &= 1 + 2u_1v_1 + 2u_2v_2 + u_1^2v_1^2 + u_2^2v_2^2 + 2u_1u_2v_1v_2 \\ &= \sum_{k=1}^{M} \Phi_k(u)\Phi_k(v) \\ &= \Phi(u)^\top \Phi(v) \\ &= \langle \Phi(u), \Phi(v) \rangle \end{aligned}$$

where

$$\Phi(v)^\top = (1, \sqrt{2}v_1, \sqrt{2}v_2, v_1^2, v_2^2, \sqrt{2}v_1v_2)$$

It's very important that these equalities are everything that makes kernelization work!

Kernel: Conclusion

We could recap that:

$$\begin{aligned} k(u, v) &= 1 + 2u_1v_1 + 2u_2v_2 + u_1^2v_1^2 + u_2^2v_2^2 + 2u_1u_2v_1v_2 \\ &= \langle \Phi(u), \Phi(v) \rangle \end{aligned}$$

Some methods only involve features via inner products $X^\top X' = \langle X, X' \rangle$.(We've explicitly seen two: ridge regression and support vector classifiers) If we make transformations of X to $\Phi(X)$, the procedure depends on $\Phi(X)^\top \Phi(X') = \langle \Phi(X), \Phi(X') \rangle$. It's crucial that we can compute this inner product via the kernel:

$$k(X, X') = \langle \Phi(X), \Phi(X') \rangle$$

Instead of creating a very high dimensional object via transformations, choose a kernel k. Now, the only thing left to do is form the outer product of kernel evaluations

$$\mathbb{K} = [k(X_i, X_{i'})]_{1 \leq i, i' \leq n}$$

```
x = c(1,2,3)# n = 3
k = function(x,y){ return(x + y + x*y)}
> outer(x,x,k)
     [,1] [,2] [,3]
[1,]    3    5    7
[2,]    5    8   11
[3,]    7   11   15
```

(Kernel) SVM

Kernel SVM

Recall that:

$$\frac{1}{2}\beta_2^2 - \sum_{i=1}^n \alpha_i [Y_i(X_i^\top \beta + \beta_0) - 1]$$

Derivatives with respect to β and β_0 imply:

- $\beta = \sum_{i=1}^n \alpha_i Y_i X_i$
- $0 = \sum_{i=1}^n \alpha_i Y_i$

Write the solution function

$$h(X) = \beta_0 + \beta^\top X = \beta_0 + \sum_{i=1}^n \alpha_i Y_i X_i^\top X$$

Kernelize the support vector classifier $\Rightarrow$ support vector machine (SVM):

$$h(X) = \beta_0 + \sum_{i=1}^n \alpha_i Y_i k(X_i, X)$$

General kernel machines

After specifying a kernel function, it can be shown that many procedures have a solution of the form

$$\hat{f}(X) = \sum_{i=1}^n \gamma_i k(X, X_i)$$

For some $\gamma_1, \ldots, \gamma_n$

Also, this is equivalent to performing the method in the space given by the eigenfunctions of k

$$k(u,v) = \sum_{j=1}^{\infty} \theta_j \phi_j(u) \phi_j(v)$$

Also, (the) feature map is

$$\Phi = [\phi_1, \ldots, \phi_p, \ldots]$$

Kernel SVMs

Hence specifying Φ itself unnecessary, we need only define the kernel that is symmetric, positive definite

Some common choices for SVMs:

- Polynomial: $k(x,y) = (1 + x^\top y)^d$
- Radial basis: $k(x,y) = e^{-\tau x - y_b^b}$
  For example, $b = 2$ and $\tau = 1/(2\sigma^2)$ is (proportional to) the Gaussian density

Kernel SVMs: Summary

the solution form for SVM is

$$\beta = \sum_{i=1}^{n} \alpha_i Y_i X_i$$

Kernelized, this is

$$\beta = \sum_{i=1}^{n} \alpha_i Y_i \Phi(X_i)$$

Therefore, the induced hyperplane is:

$$\begin{aligned} h(X) = \Phi(X)^\top \beta + \beta_0 &= \sum_{i=1}^{n} \alpha_i Y_i \langle \Phi(X), \Phi(X_i) \rangle + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i Y_i k(X, X_i) + \beta_0 \end{aligned}$$

The final classification is still $\hat{g}(X) = \operatorname{sgn}(\hat{h}(X))$

SVMs via penalization

Note that SVMs can be derived from penalized loss methods. The support vector classifier optimization problem:

$$\min_{\beta_0, \beta} \frac{1}{2} \beta_2^2 + \lambda \sum \xi_i \text{ subject to}$$

$$Y_i h(X_i) \geq 1 - \xi_i, \xi_i \geq 0, , \text{ for each } i$$

Writing $h(X) = \Phi(X)^\top \beta + \beta_0$, consider

$$\min_{\beta, \beta_0} \sum_{i=1}^{n} [1 - Y_i h(X_i)]_+ + \tau \beta_2^2$$

These optimization problems are the same! With the relation: $2\lambda = 1/\tau$. The loss part is the hinge loss function

$$\ell(X, Y) = [1 - Y h(X)]_+$$

The hinge loss approximates the zero-one loss function underlying classification. It has one major advantage, that is convexity.

Surrogate losses: convex relaxation

Looking at

$$\min_{\beta, \beta_0} \sum_{i=1}^{n} [1 - Y_i h(X_i)]_+ + \tau \beta_2^2$$

It is tempting to minimize (analogous to linear regression)

$$\sum_{i=1}^{n} \mathbf{1}(Y_i \neq \hat{g}(X_i)) + \tau \beta_2^2$$

However, this is nonconvex (in $u = h(X)Y$)

A common trick is to approximate the nonconvex objective with a convex one. This is known as convex relaxation with a surrogate loss function

Surrogate losses

Idea: We can use a surrogate loss that mimics this function while still being convex.

- Hinge: $[1 - Y h(X)]_+$
- Logistic: $\log(1 + e^{-Y h(X)})$