UNIVERSITY OF CALIFORNIA, BERKELEY

Lecture 14 - CS:189

Oscar Ortega

July 16, 2021

1 ISL: 9.3.2

1.1 THE SUPPORT VECTOR MACHINE

What separates the support vector classifier from the Support Vector Machine is what occurs from enlarging the feature using **kernels**
The linear support vector classifier can be represented as

$$f(x) = b_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

Where there are n parameters $\alpha_i, i = 1, ..., n$ one per training observation.
If training observation is not a support vector than $\alpha_i = 0$, That means we can write a classifier purely in terms of the support vectors of the classifier. Let $\mathscr{S}$ be defined as the set of support vectors for this note: We can generalize the notion of the inner product computation of the form

$$K(x_i, x_i')$$

where K is known as a **Kernel** function.
We define a kernel function as one that quantifies the similarity of two observations.
Furthermore, take note that the classifier would then take the following form:

$$f(x) = b_0 + \sum_{i \in \mathscr{S}} \alpha_i K(x, x_i)$$

Some important kernels:

- This is known as a **degree d polynomial Kernel**:

$$K(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j}^{'})^d$$

  Take note that if $k = 1$, this reduces to the linear kernel defined initially.

- the **Radial Kernel** is defined as follows:

$$K(x_i, x_i^{'} = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

  Radial kernel has very local behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

An advantage of Kernels as opposed to using and enlarging the feature space is that we save computation.

2 ESL: 12.3-12.3.1

Recall that the for a support vector classifier function can be written in the following form:

$$f(x) = h(x)^T \beta + \beta_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$$

Recall that we can deterimine the values of the scalars α_i, β_0 by solving $y_i f(x_i) = 1$ for any x_i where $0 < \alpha_i < C$, where C is the hyperparameter we define in the optimization.
We define the kernel function as an inner product on a transformed space $h(x)$, in other words, it is equal to the following:

$$K(x, x') = \langle h(x) h(x') \rangle$$

K should always be a positive semi-definite function.
We mentioned some popular kernel functions in the previous section, heres another one.

$$K(x, x') = \tanh(k_1 \langle x, x' \rangle + k_2)$$

This is known as the Neural Network Kernel.

3 Lecture: Kernels

Recall: We can apply these tricks to both regression and classification
If we have d input features, degree-p polynomials blow up to $O(d^p)$ When d is large this quickly becomes intractable. What else can we do?
We use 'magic' : we use the features without computing them! Observation: In many learning algorithms

- the weights can be written as a linear combination of sample pts.
- we can use inner products of $\Phi(x)$'s only. Recall $\Phi(x)$, is notation for a lifted feature vector
- we can use inner products of $\Phi(x)$'s only don't need to compute $\Phi(x)$!

let $w = X^T\alpha = \sum_{i=1}^{n} \alpha_i X_i$ for some $\alpha \in \mathbf{R}^n$ Once we know this is true, we can substitute into algorithms and optimize n dual weights α_i. These are known as the **dual weights** a. There are n primal weights, and d primal weights and are also known as dual parameters. Instead of d^P primal weights w.

4 Kernel Ridge Regression

We need to penalize the bias term to make this work. So in order to mitigate this penalization, we center X and y.

$$X_i X_i - \mu_x$$

$$y_i \rightarrow y_i - \mu_y$$

This lets us replace I' with I in normal equations.

$$(X^T X + \lambda I)w = X^T y$$

$$w = \frac{1}{\lambda}(X^T y - X^T X w) = X^T \alpha$$

where

$$\alpha = \frac{1}{\lambda}(y - Xw)$$

As we saw from a previous hw, this shows that w is a linear combination of sample pts. This means we can now solve for α instead of w

$$\lambda\alpha = y - Xw = y - XX^T\alpha \rightarrow \alpha = (XX^T + \lambda_i)^{-1}y$$

Here α is a **dual solution** to the ridge regression problem; solves the **dual form** of ridge regression. The dual form of ridge regression is as follows:

$$\min_\alpha \|XX^T\alpha - y\|_2^2 + \lambda\|X^T\alpha\|_2^2$$

Where the solution is the alpha value computed above.
This means that our regression function is:

$$h(z) = w^T z = a^T Xz = \sum_{i=1}^{n} a_i (X_i^T z)$$

Note that our inner product form gives us flexibility in choosing to compute $(a^T X)z$ or $a^T(Xz)$

$$\text{Let} k(x,z) = x^T z \text{ be known as the kernel fn}$$

We also define the **Kernel Matrix** below:

$$K = XX^T \in \mathbf{R}^{n,n}$$

Note: $K_{i,j} = K(X_i, X_j)$ and that K is singular if $n > d+1$.
In that case, no solution if $\lambda = 0$ With this in mind, we can now perform dual ridge regression alg as follows:'

$$\forall i, j : K_{i,j} = K(x_i, x_j)$$

$$\text{solve } (K + \lambda_I)\alpha = y \text{ for } \alpha$$

$$\text{for each test pt. z}$$

$$h(z) \rightarrow \sum_{i=1}^{n} \alpha_i K(X_i, z)$$

Where the first two lines correspond to training, and the next to involve to testing.
Note that we do not use X_i directly! only k.

- Dual: $O(n^3 + n^2 d)$ time
- Primal: $O(d^3 + d^2 n)$ time

Node that we prefer the dual if $d > n$,

5 The Kernel Trick - (Kernelization)

Consider a degree p polynomial kernel:

$$K(x, z) = (x^T z + 1)^p$$

Theorem: $(x^T z + 1)^P = \Phi(x)^T \Phi(z)$, where $\Phi(x)$ contains every monomial in x of degree 0,...,p.
As an example, for d = 2, p = 2:

$$(x^T z + 1)^2 = x_1^2 z_1^2 + x_2^T z_2^2 + 2x_1 z_1 x_2 z_2 + +2x_2 z_2 + 1$$

$$= \Phi(x)^T \Phi(z) = \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1 x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix} \begin{bmatrix} z_1^2 \\ x_2^2 \\ \sqrt{2}z_1 z_2 \\ \sqrt{2}z_1 \\ \sqrt{2}z_2 \\ 1 \end{bmatrix}$$

Key Gain: we can compute the inner products of these vectors in $O(d)$ time as opposed to $O(d^P)$ time. In kernel ridge regression, we replace X_i with $\Phi(x_i)$:

$$\text{Let} K(x, z) = \Phi(x)^T \Phi(z)$$

We don't compute $\Phi(x)$ or $\Phi(z)$, but we compute $(x^T z + 1)^P$

6 Kernel Perceptrons

Featurized perceptron algorithm:

$$w \leftarrow y_1 \Phi(x_i)$$

$$\text{while some } y_i \Phi(X_i) \cdot w < 0:$$

$$w \leftarrow w + \epsilon y_i \Phi(x_i)$$

$$\text{for each test pt z}$$

$$h(z) \rightarrow w \cdot \Phi(z)$$

Let $\Phi(X)$ be n by D matrix with rows $\Phi(X_i)^T$, D = length of $\Phi(.)$.

$$K = \Phi(X)\Phi(X)^T$$

Dualize with w = $\Phi(X)^T \alpha$ Then the code $\alpha_i \rightarrow \alpha_i + \epsilon y_i$ has the same effect as the update in the featurized perceptron algorithm.

$$\Phi(X_i) \cdot w = (\Phi(X)w)_i = (\Phi(X)\Phi(X)^T \alpha)_i = (K\alpha)_i$$

Therefore, we can define the dual perceptron algorithm as follows:

$$\alpha = \begin{bmatrix} y_1 & 0 & \dots & 0 \end{bmatrix}^T$$

$$\forall i, j:$$

$$K_{i,j} = k(X_i, X_j)$$

$$\text{while some } y_i (K\alpha)_i < 0$$

$$\alpha_i \leftarrow \alpha_i + \epsilon y_i$$

$$\text{for each test pt z}:$$

$$h(z) = \sum_{j=1}^{n} \alpha_j k(X_j, z)$$

Where the sign of the prediction corresponds to the label.

7 Kernel Logistic Regression

Stochastic Gradient Descent step:

$$\alpha_i = \alpha_i + \epsilon(y_i - s((K\alpha)_i))$$

Batch Gradient descent Step:

$$\alpha \leftarrow \alpha + \epsilon(y - s(K\alpha))$$

For each test pt z:

$$h(z) = s(\sum_{j=1}^{n} \alpha_j k(x_j, z)$$

8 THE GAUSSIAN KERNEL

The Gaussian Kernel, also known as the Radial Basis fn. Kernel: there exists a $\Phi(x)$ such that

$$K(x,z) = \exp(-\frac{(x-z)}{2\sigma^2})$$

Eg: for d = 1,

$$\Phi(x) = \exp\begin{bmatrix}1 & \frac{x^2}{\sigma\sqrt{1!}} & \dots\end{bmatrix}$$

Key observation: hypothesis h(z) is a linear combination of gaussians centered at the sample pts. Very popular in practice: Why?

- Gives very smooth function
- Behaves somewhat like k-nearest neighbors
- Oscillates Less than polynomial Kernel (depends on σ)
- k(x,z) interpreted as similarity measure
- maximum when z = x, and approaches 0 as distance increases.
- sample pts: "vote" for value at z, but closer points get weightier vote.

σ parameter trades off bias vs. variance: Larger σ: wider Gaussians, smoother h and more bias Smaller σ: