Lecture #19: Support Vector Machines #2

CS 109A, STAT 121A, AC 209A: Data Science

Pavlos Protopapas Kevin Rader
Margo Levine Rahul Dave

Lecture Outline

Review

Extension to Non-linear Boundaries

Review

Classifiers and Decision Boundaries

Last time, we derived a linear classifier based on the intuition that a good classifier should

- maximize the distance between the points and the decision boundary (maximize margin)
- misclassify as few points as possible

SVC as Optimization

With the help of geometry, we translated our wish list into an optimization problem

$$\begin{cases} \min\limits_{\xi_n \in \mathbb{R}^+, w, b} \|w\|^2 + \lambda \sum_{n=1}^{N} \xi_n \\ \text{such that } y_n(w^\top x_n + b) \geq 1 - \xi_n,\ n = 1, \ldots, N \end{cases}$$

where ξ_n quantifies the error at x_n.

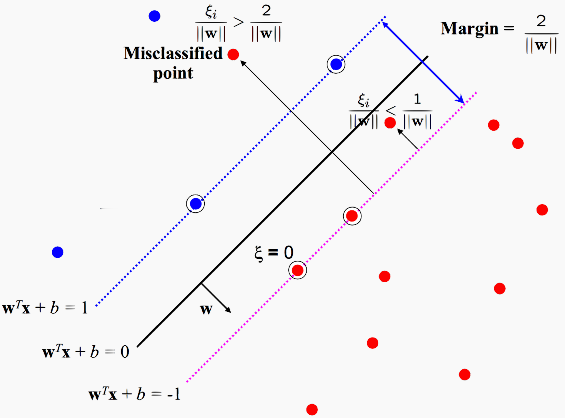
The SVC optimization problem is often solved in an alternate form (the dual form)

$$\max_{\alpha_n \geq 0, \sum_n \alpha_n y_n = 0} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^\top x_m$$

Later we'll see that this alternate form allows us to use SVC with non-linear boundaries.

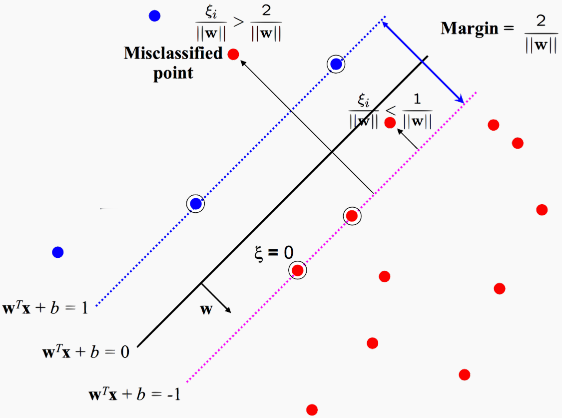
Decision Boundaries and Support Vectors

Recall how the error terms ξ_n's were defined: the points where $\xi_n = 0$ are precisely the support vectors



Decision Boundaries and Support Vectors

Thus to re-construct the decision boundary, ***only the support vectors are needed!***



Decision Boundaries and Support Vectors

- The decision boundary of an SVC is given by

$$\hat{w}^\top x + \hat{b} = \sum_{x_n \text{ is a support vector}} \hat{\alpha}_n y_n (x_n^\top x_n) + b$$

where $\hat{\alpha}_n$ and the set of support vectors are found by solving the optimization problem.

- To classify a test point x_{test}, we predict

$$\hat{y}_{test} = \text{sign}\left(\hat{w}^\top x + \hat{b}\right)$$

Extension to Non-linear Boundaries

Polynomial Regression: Two Perspectives

Given a training set

$$\{(x_1, y_1), \ldots, (x_N, y_N)\}$$

with a single real-valued predictor, we can view fitting a 2nd degree polynomial model

$$w_0 + w_1 x + w_2 x^2$$

on the data as the process of finding the best quadratic curve that fits the data. But in practice, we first expand the feature dimension of the training set

$$x_n \mapsto (x_n^0, x_n^1, x_n^2)$$

and train a ***linear model*** on the expanded data

$$\{(x_n^0, x_n^1, x_N^2, y_1), \ldots, (x_N^0, x_N^1, x_N^2, y_N)\}$$

Transforming the Data

The key observation is that training a polynomial model is just training a linear model on data with transformed predictors.

In our previous example, transforming the data to fit a 2nd degree polynomial model requires a map

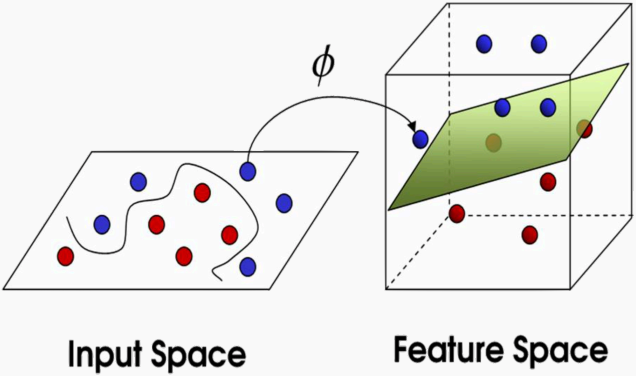
$$\phi : \mathbb{R} \to \mathbb{R}^3$$
$$\phi(x) = (x^0, x^1, x^2)$$

where $\mathbb{R}$ called the ***input space***, $\mathbb{R}^3$ is called the ***feature space***.

While the response may not have a linear correlation in the input space $\mathbb{R}$, it may have one in the feature space $\mathbb{R}^3$.

SVC with Non-Linear Decision Boundaries

The same insight applies to classification: while the response may not be linear separable in the input space, it may be in a feature space after a fancy transformation:



SVC with Non-Linear Decision Boundaries

The motto: instead of tweaking the definition of SVC to accommodate non-linear decision boundaries, we map the data into a feature space in which the classes are linearly separable (or nearly separable):

- Apply transform $\phi : \mathbb{R}^J \to \mathbb{R}^{J'}$ on training data

  $$x_n \mapsto \phi(x_n)$$

  where typically J' is much larger than J.

- Train an SVC on the transformed data

  $$\{(\phi(x_1), y_1), \ldots, (\phi(x_N), y_N)\}$$

The Kernel Trick

Since the feature space $\mathbb{R}^{J'}$ is extremely high dimensional, computing ϕ explicitly can be costly.

Instead, we note that computing ϕ is unnecessary.

Recall that training an SVC involves solving the optimization problem

$$\max_{\alpha_n \geq 0,\, \sum_n \alpha_n y_n = 0} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} y_n y_m \alpha_n \alpha_m \phi(x_n)^\top \phi(x_m)$$

In the above, ***we are only interested in computing inner products*** $\phi(x_n)^\top \phi(x_m)$ ***in the feature space*** and not the quantities $\phi(x_n)$.

The ***inner product*** between two vectors is a measure of the similarity of the two vectors.

Definition

Given a transformation $\phi : \mathbb{R}^J \to \mathbb{R}^{J'}$, from input space $\mathbb{R}^J$ to feature space $\mathbb{R}^{J'}$, the function $K : \mathbb{R}^J \times \mathbb{R}^J \to \mathbb{R}$ defined by

$$K(x_n, x_m) = \phi(x_n)^\top \phi(x_m), \quad x_n, x_m \in \mathbb{R}^J$$

is called the ***kernel function*** of ϕ.

Generally, ***kernel function*** may refer to any function $K : \mathbb{R}^J \times \mathbb{R}^J \to \mathbb{R}$ that measure the similarity of vectors in $\mathbb{R}^J$, without explicitly defining a transform ϕ.

The Kernel Trick

For a choice of kernel K,

$$K(x_n, x_m) = \phi(x_n)^\top \phi(x_m)$$

we train an SVC by solving

$$\max_{\alpha_n \geq 0,\ \sum_n \alpha_n y_n = 0} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} y_n y_m \alpha_n \alpha_m {\color{red} K(x_n, x_m)}$$

Computing $K(x_n, x_m)$ can be done without computing the mappings $\phi(x_n), \phi(x_m)$.

This way of training a SVC in feature space without explicitly working with the mapping ϕ is called ***the kernel trick***.

Transforming Data: An Example

Example

Let's define $\phi : \mathbb{R}^2 \to \mathbb{R}^6$ by

$$\phi\left([x_1, x_2]\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

The inner product in the feature space is

$$\phi\left([x_{11}, x_{12}]\right)^\top \phi\left([x_{21}, x_{22}]\right) = \left(1 + x_{11}x_{21} + x_{12}x_{22}\right)^2$$

Thus, we can directly define a kernel function $K : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ by

$$K(x_1, x_2) = \left(1 + x_{11}x_{21} + x_{12}x_{22}\right)^2.$$

Notice that we need not compute $\phi\left([x_{11}, x_{12}]\right), \phi\left([x_{21}, x_{22}]\right)$ to compute $K(x_1, x_2)$.

Kernel Functions

Common kernel functions include:

- **Polynomial Kernel** (`kernel='poly'`)

  $$K(x_1, x_2) = (x_1^\top x_2 + 1)^d$$

  where d is a hyperparameter

- **Radial Basis Function Kernel** (`kernel='rbf'`)

  $$K(x_1, x_2) = \exp\left\{-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right\}$$

  where σ is a hyperparameter

- **Sigmoid Kernel** (`kernel='sigmoid'`)

  $$K(x_1, x_2) = \tanh(\kappa x_1^\top x_2 + \theta)$$

  where κ and θ are hyperparameters.

Let’s go to the notebook

at min need 3 support vectors bc if only 2, then boundary just has to go thru midpoint between 2 points, so need 3rd one too.

when C is very large, few points in support vectors, so more variability, overfit to the data. when C is small and using more points, less variability.

if have lots of points really close to boundary or lots of overlap, logistic regression gets overwhelmed. SVM does better because directly estimating boundary