Lecture 23: Support vector machines

Reading: Chapter 9

STATS 202: Data mining and analysis

Sergio Bacallado
November 18, 2013

Announcements

- Today at 5pm is the change of grading basis deadline.
- Homework 7 is out (last one!). The installation of the R package stringkernels is bringing up problems. We will post a fix by the end of the day today.
- Today's lecture is the last lecture that counts towards the final. You can start reviewing every chapter of the book.
- We are preparing a list of practice problems for the final.
- You have 18 days left to make submissions in Kaggle. Credit for the competition will require:
  - At least 1 submission beating the null prediction.
  - A one-paragraph description of what you did.

Exercise 11 in Homework 6

- **Backfitting** is a method to estimate the coefficients of a GAM, for example:

$$y = f_1(x) + f_2(x) + \epsilon.$$

- Suppose f_1 and f_2 are smoothing splines.
- We know how to find the optimal $\hat{f}_1$ for a problem:

$$y = f_1(x) + \epsilon,$$

  and the optimal $\hat{f}_2$ for the problem:

$$y = f_2(x) + \epsilon.$$

- However, estimating $\hat{f}_1$ and $\hat{f}_2$, which simultaneously optimize the fit of the GAM is more difficult.

Exercise 11 in Homework 6

- **Idea of Backfitting:** Alternate the following
  1. Fix f_2 and set f_1 to the best fit for the model:
  $$y - f_2(x) = f_1(x) + \epsilon.$$
  2. Fix f_1 and set f_2 to the best fit for the model:
  $$y - f_1(x) = f_2(x) + \epsilon.$$
- We offered little reason to believe that this algorithm would converge to the right answer $\hat{f}_1, \hat{f}_2$.

Exercise 11 in Homework 6

- **Goal of the exercise:** Examine by simulation how backfitting works for a linear model.

$$y = \beta_1 x_1 + \beta_2 x_2.$$

  Or $f_1(x) = \beta_1 x_1$, and $f_2(x) = \beta_2 x_2$.

- Why linear regression? In this case, we know how to fit f_1 and f_2 simultaneously. The *right* answer is the estimate from Multiple Linear Regression.
- Conclusions of the simulation:
  - The backfitting algorithm converges to the right answer.
  - The convergence is extremely fast.
- Backfitting is a special case of the Gauss-Seidel algorithm, so we can *prove* that the convergence holds for several GAMs.

Review of support vector classifier

- The **support vector classifier** defines a hyperplane and two margins.
- Aims to maximize the width of the margins, with some budget C for "violations of the margins", i.e.

$$\sum_{\substack{x_i \text{ on the wrong} \\ \text{side of the margin}}} \text{distance from } x_i \text{ to the margin } \leq C.$$

- The only points that affect the orientation of the hyperplane are those on the wrong side of the margin.
- Low budget $C \iff$ Few samples used $\iff$ High variance $\iff$ Tendency to overfit.

Key fact about the support vector classifier

To find the hyperplane, all we need to know about the data matrix X is the dot product:

$$\langle x_i, x_k \rangle = \sum_{j=1}^{p} x_{ij} x_{kj}$$

between every pair of observations or samples.

Support vector machines

- A **support vector machine** is a support vector classifier applied on an expanded set of predictors:

$$\Phi : (X_1, X_2) \to (X_1, X_2, X_1X_2, X_1^2, X_2^2).$$

- We expand the vector of predictors for each sample x_i and then perform the algorithm.
- We only need to know the dot products:

$$\langle \Phi(x_i), \Phi(x_k) \rangle \equiv K(i, k)$$

for every pair of samples (x_i, x_k).

The kernel trick

- Often, the dot product:

$$\langle \Phi(x_i), \Phi(x_k) \rangle \equiv K(i,k)$$

  is a simple function $f(x_i, x_k)$ of the original vectors. Even if the mapping Φ significantly expands the space of predictors.

- **Example 1:** Polynomial kernel

$$K(i,k) = (1 + \langle x_i, x_k \rangle)^2.$$

- With two predictors, this corresponds to the mapping:

$$\Phi : (X_1, X_2) \to (\sqrt{2}X_1, \sqrt{2}X_2, \sqrt{2}X_1X_2, X_1^2, X_2^2).$$

The kernel trick

- Often, the dot product:

$$\langle \Phi(x_i), \Phi(x_k) \rangle \equiv K(i,k)$$

  is a simple function $f(x_i, x_k)$ of the original vectors. Even if the mapping Φ significantly expands the space of predictors.

- **Example 2:** RBF kernel

$$K(i,k) = \exp(\gamma d(x_i, x_k)^2),$$

  where d is the Euclidean distance between x_i and x_k.

- In this case, the mapping Φ is an expansion into an infinite number of transformations! We can apply the method even if we don't know what these transformations are.

The kernel trick

- **Because of math...** if the matrix K is positive semi-definite, then there exists *some* mapping Φ.
- It is not too hard to show that many kernels K are positive semi-definite.
- If we don't know which transformations we are using, why would we expect the SVM to work?
  - The kernel $K(i,k) = f(x_i, x_k)$ measures the similarity between samples x_i and x_k.
  - We can evaluate whether K is a good measure of similarity without understanding the feature expansion Φ.

Kernels for non-standard data types

- We can define families of kernels (with tuning parameters), which capture similarity between non-standard kinds of data:
  1. Text, strings
  2. Images
  3. Graphs
  4. Histograms
- Sometimes we know the mapping Φ, but there are algorithms that are fast for computing $K(i, k)$ without doing the expansion explicitly.
- Other times, the expansion Φ is infinite-dimensional or simply not known.

The kernel trick can be applied beyond SVMs

Kernel PCA:

- Suppose we want to do PCA with an expanded set of predictors, defined by the mapping Φ.
- If we want to make a biplot of the observations, all we need to know is the kernel or Gram matrix:

$$K(i,k) = \langle \Phi(x_i), \Phi(x_k) \rangle.$$

- Even if Φ expands the predictors to a very high dimensional space, we can do PCA!
- The cost only depends on the number of observations n.

Applying SVMs with more than 2 classes

- SVMs don't generalize nicely to the case of more than 2 classes.
- Two main approaches:
  1. **One vs. one:** Construct $\binom{n}{2}$ SVMs comparing every pair of classes. Assign a sample i to the class that wins the most one-on-one challenges.
  2. **One vs. all:** Construct a classifier $\beta^{(k)}$ for every class k against all others. Assign a sample i to the class k, such that the distance from x_i to the hyperplane defined by $\beta^{(k)}$ is largest (the distance is negative if the sample is misclassified).

Relationship to logistic regression

We can formulate the method for finding a support vector classifier

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

as a Loss + Penalty optimization:

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2.$$

For each sample (x_i, y_i) we incur a loss $\max[0, 1 - y_i f(x_i)]$ by using this classifier. In logistic regression, we minimize the loss

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} \log[1 + e^{-y_i f(x_i)}]$$

Comparing the losses