Techniques for Analyzing Stochastic Time-Series Data

Dennis Castleberry Brandon Oubre Haikou Yu

November 24, 2013

Project Overview

- Test a variety of different classifiers on different data sets.
- Can we improve classification accuracy of time-series data by adding attributes to current data from previous data?
- For example, does informing a classifier of the class of the previous n entries improve prediction accuracy for the current observation?
- How far back should we look and what information should we include?

The Naive Bayes Classifier

- Reduce classification to probability. What is $P(class|attribute1, attribute2, ..., attributeN)$.
- Assumes that each attribute is independent of the others. (Hence the "Naive" nickname.)
- For example, let's consider if a car is stolen using $P(stolen|Color, Type)$. Naive Bayes will assume $color = red$ and $type = sportscar$ to be independent.
- Naive Bayes is not sensitive to irrelevant attributes, since the probabilities of such attributes will be similar for all classes.
- Naive Bayes is quick to train, as it requires only one pass-though of the training data.

Naive Bayes in Action

Training Data

Over 170cm	**Eye Color**	**Hair Length**	**Sex**
No	Blue	Short	Male
Yes	Brown	Long	Female
No	Blue	Long	Female
Yes	Brown	Short	Male
Yes	Brown	Short	Female

Only discrete values shown, but we can still interpret real data using normal distributions!

Suppose we are given an unseen data point $\langle No, Blue, Short \rangle$. What should we classify it as?

Naive Bayes in Action

$$P(Male|No, Blue, Short)$$
$$= \frac{P(No, Blue, Short|Male)P(Male)}{P(No, Blue, Short)}$$
$$= \alpha P(Male)\boldsymbol{P(No|Male)P(Blue|Male)P(Short|Male)}$$
$$= \alpha \times \frac{2}{5} \times \frac{1}{2} \times \frac{1}{2} \times \frac{2}{2} = \boxed{0.1\alpha}$$

$$P(Female|No, Blue, Short)$$
$$= \alpha P(Female)P(No|Female)P(Blue|Female)P(Short|Female)$$
$$= \alpha \times \frac{3}{5} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} = \boxed{0.0\overline{2}\alpha}$$

Since $P(Male|Data) > P(Female|Data)$, we classify the unseen point as Male. For multiple classes, just select the class with the greatest probability!

Support Vector Machines (SVM)

- Idea is to draw a line (or hyperplane) between the data points of different classes. Classify unseen data by testing which side of the line it is on.
- Focus on support vectors, or the points that would change the line if removed from the training data.
- Find an optimal line to separate the data. Such a line will have the larger margin for data points and should mis-classify the least number of new points.
- If data is not linearly separable, then a transformation of the data to a new basis can be performed. The data may be linearly separable in the new basis.

SVM Example

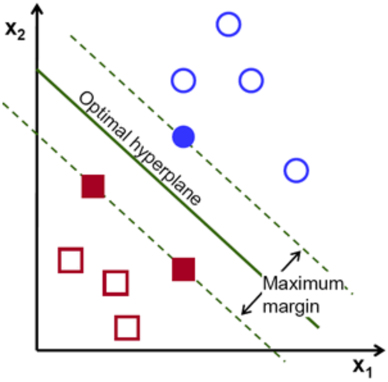

Image from http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

- Solid Figures are support vectors.
- Due to the maximized margin, unseen figures can be closer to the line than the support vectors and still be correctly classified.
- It is easy to see how new points are classified.

Neural Networks

- Inspired by biological neurons.
- Neurons maintain a weighted sum of their inputs. The result of this sum is passed into a function and output. (A step function produces on/off signals while a Sigmoid will produce continues levels of activation.)
- The network can be trained by adjusting the weights of the inputs to each neuron.
- In a feed-forward network, the backwards propagation algorithm accomplishes this.
- Networks with multiple layers can classify various types of non-linearly separable data.

An Artificial Neuron

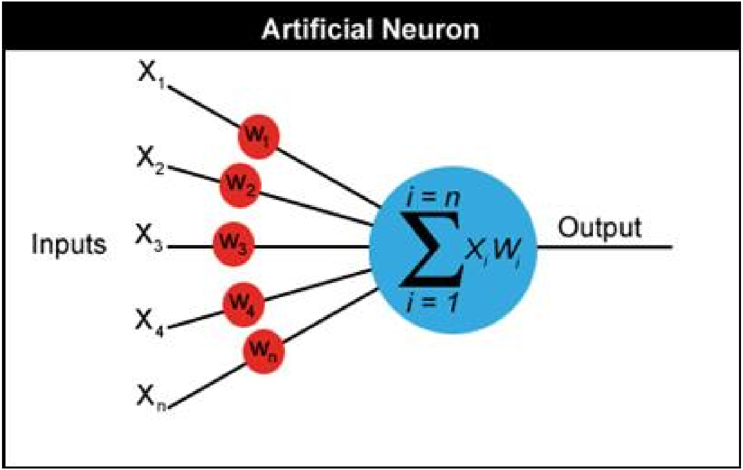

Image from http://www.ai-junkie.com/ann/evolved/nnt1.html

Neural Network Classification

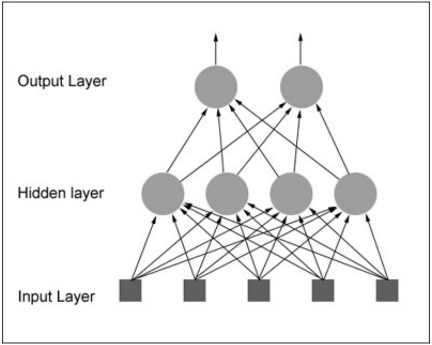

Image from http://www.ai-junkie.com/ann/evolved/nnt1.html

- Information is fed into the input layer.
- The outputs of the neurons in the output layer represent classifications.
- Hidden layers perform intermediary manipulations of signals. More hidden layers can be added as needed.

K-Nearest Neighbor

- Assumes that data vectors lie in a metric space.
- Training is simple. KNN stores all of the training data points with no computation. (KNN is lazy.)
- Classification is also simple. To classify point x, find the k points in the training data closest to x. Classify x as the majority vote its k-nearest neighbors.
- Can also weight votes based on the distance of the the neighbors.
- KNN suffers from the curse of dimensionality. (In high dimensions points start to become equidistant. This means metrics such as Euclidean distance become unhelpful.)

KNN Example

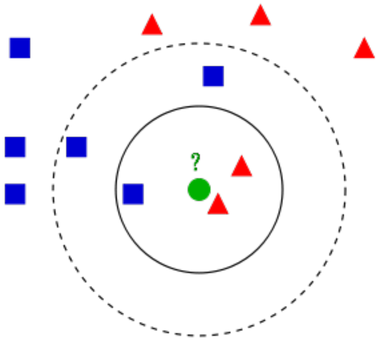

Image from
http://en.wikipedia.org/wiki/File:KnnClassification.svg

- Red and blue points represent training data.
- The green point is being classified.
- For $k = 3$ the point is classified as red.
- For $k = 5$ the point is classified as blue.
- Weighting votes by distance may shift favor back to red.

Classification and Regression Tree (CART)

- Create binary decision trees. Minimize the error in each leaf.
- Produces a classification tree for categorical data and a regression tree for numerical data.
- Data is recursively split according to rules until a set of stopping rules are met or when no further gain can be made.
- Can also split data as much as possible and the prune.
- Each internal node is a decision.
- Each leaf is a classification, which classifies according to majority vote of training data that follows that tree path.

CART Example

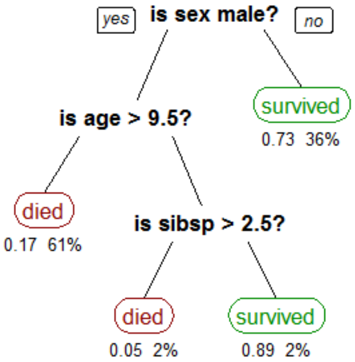

Image from http://en.wikipedia.org/wiki/File:CART_tree_titanic_survivors.png

- CART tree for classifying survival of passengers on the Titanic.
- Sibsp is number of spouses and siblings aboard the ship.
- Tree also shows probability of survival and percentage of observations.
- To classify unseen data point simply follow the tree until a leaf node is met.