Techniques for Analyzing Stochastic Time-Series Data

Dennis Castleberry Brandon Oubre Haikou Yu

December 2, 2013

Project Overview

- Time series data is time-ordered, with each data entry at a later time step or stamp.
- Given a prediction task, unlike with regular data sets, the order of the data matters; e.g. using previous time-ordered values to predict the next value in a series.
- Given all time-dependent data up to a certain point, our task is to predict the class value for the next time step.
- Our project examines feature extraction on multivariate stochastic time series data to maximize classification accuracy for various classifiers (Naive Bayes, neural network, SVM, KNN, and CART).
- How can we do feature extraction on attribute values for previous time steps to achieve accurate predictions?

Project Overview

- For our base model, we use the raw attribute set. We use this model as a comparison for our other models.
- To generate other models, we improve add attributes to current data from previous data.
- For example, we hypothesize: does informing a classifier of the class of the previous n entries improve prediction accuracy for the current observation? We create a model wherein our derived attributes consist of the class values for the previous n entries.
- Our overarching goal is to find the relationship between classifier accuracy and how much previous data to include; what are the trade-offs of deriving large amounts of data; how should the data be incorporated.

The Naive Bayes Classifier

- Reduce classification to a computation of probabilities. 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 in Action

Training Data

Over 170cm	**Eye Color**	**Hair Length**	$\boxed{\textbf{Sex}}$
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 apply numeric data using standard deviations if the data is normally distributed.

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

Naive Bayes in Action

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

$$
\begin{aligned}
&P(Female|No, Blue, Short) \\
&= \alpha P(Female)P(No|Female)P(Blue|Female)P(Short|Female) \\
&= \alpha \times \tfrac{3}{5} \times \tfrac{1}{3} \times \tfrac{1}{3} \times \tfrac{1}{3} = \boxed{0.0\overline{2}\alpha}
\end{aligned}
$$

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

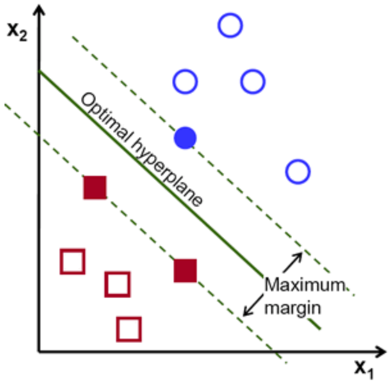

Image from `http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`

- Solid points are support vectors.
- With a maximized margin, unseen figures closer to the line than the support vectors are still 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

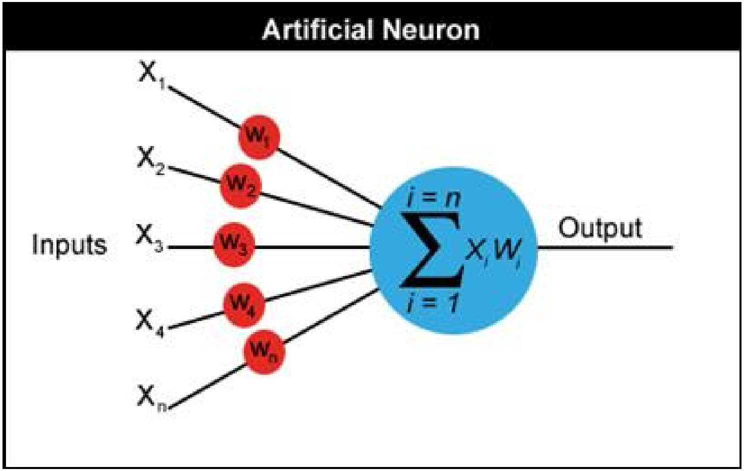

Image from `http://www.ai-junkie.com/ann/evolved/nnt1.html`

Neural Network Classification

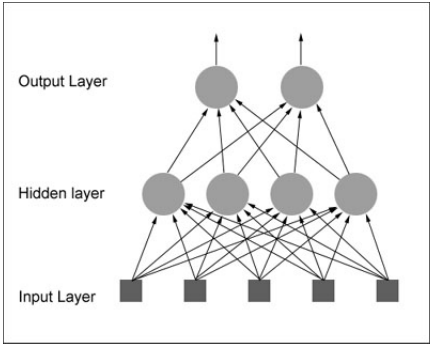

Image from `http://www.ai-junkie.com/ann/evolved/nnt1.html`

- Information is fed into the input layer.
- The outputs of the neurons in the output layer represent predicted class values.

K-Nearest Neighbor

- Assumes that data vectors lie in a metric space (a distance measure can be computed).
- Classification is simple. To classify point x, find the k points in the training data closest to x. Classify x as the majority vote of its k-nearest neighbors' class.
- Can also weight votes based on the distance of the neighbors.

KNN Example

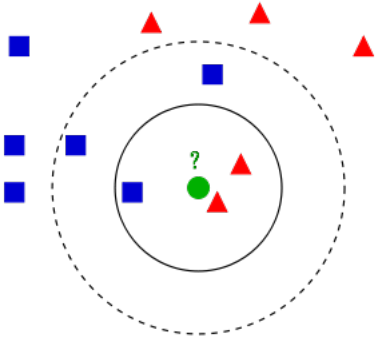

Image from
http://en.wikipedia.org/wiki/File:KnnClassification.svg

- 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

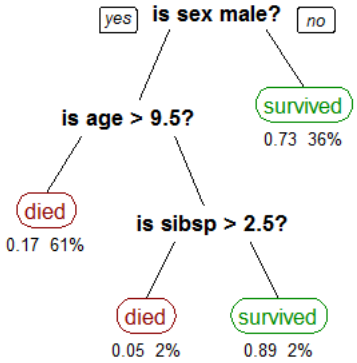

Image from `http://en.wikipedia.org/wiki/File:CART_tree_titanic_survivors.png`

- 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.

Experiment Setup

- We are given a time series T where each entry has attributes $A_1, A_2, ..., A_N$ and classes value C.
- Cannot use k-fold cross-validation.
    - Data order not interchangeable.
    - Must predict future entry given past entries.
    - The more unseen data we use to make predictions, the less accurate future predictions become. (Think of weather predictions.)
- Solution: Split series up into partitions of size M. Train model on data entries $1, 2, ..., (M - 1)$ and predict unseen entry M. Tally up number of correct predictions from all partitions in the data set to determine accuracy.

Experimental Conjecture

- Since the time series is stochastic, the attributes and class values of $T[1:i-1]$ may influence the attribute and class values of $T[i]$.
- Our conjecture is that adding derived attributes to each element's native attributes can improve classification accuracy. These derived attributes contain information about previous elements in the time series, which gives them predictive power.
- For example, the average weather conditions over the past 10 minutes could help predict the weather condition one minute from now.

Experiment Design

- Establish model H_0, which is a classification task done on the raw data set (No derived attributes).
- Create hypotheses $H_1, H_2, ..., H_N$, which each add different sets of derived attributes to the raw data.
- Run each H_i through the same classifier and observe the difference in classifier accuracy.
- Repeat this process for new hypotheses and different classifiers.
- Analyze results to determine which Hypotheses perform better with which classifiers, if any.

Hypotheses

- In H_1 we use as a derived attribute the class value from the previous iteration. We simply copy the class as an attribute of the data set and shift the column up by one.
- In H_2 we use as derived attributes the previous values for the native attributes. To do so, we simply copy the attribute data set and shift the column up by one.
- In H_3 we use as derived attributes exponential moving averages (MAs) of the previous m values. The MA is defined as $\sum_{j=1}^{m} \alpha_j x_{i-j}$. For an exponential moving average, all α's have weights which are proportional to i.

Data: Glucose Levels of a Diabetic

- Time series which tells whether glucose level is relatively abnormal (outside of one standard deviation relative to mean glucose level throughout the time period).
- Task: Predict whether glucose level for next iteration is normal or abnormal.
- Attributes: month, day, hour, minute.
- Class: normal/abnormal.

Data: EEG Eye Closure

- All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The eye state was detected via a camera during the EEG measurement and added later manually to the file after analysing the video frames.
- Task: Tell whether eyes will be open or closed on next iteration.
- Attributes: EEG electrode potentials.
- Class: Eyes open/closed.

Data: Electricity Consumption

- Time series data on power consumption of a household over time.
- Task: predict normality/abnormality of power consumption for future.
- Attributes: day, month, year, hour, minute, second, reactive, voltage
- Class: normal/abnormal.

Data: Stock Exchange Data

- Time series data on the values of stock indices.
- Task: Predict whether the Emergent Markets Index (EMI) will go up or down on the next iteration.
- Attributes: day, month, year, ISE, SP, DAX, FTSE, NIKKEI, BOVESPA, EU
- Class: EMI up/down.

Implementation

- Our implementation was done in R. We used packages:
  - **neuralnet** for ANN
  - **e1071** for BAYES
  - **rpart** for CART
  - **e1071** for KNN
  - **kernlab** for SVM
- Much of our implementation work dealt with pre-processing the data and adding the derived attributes to our data set for our hypotheses.

Results

- Results were generated:
  - For hypotheses 0, 1, 2, and 3
  - Over multiple data sets (diabetes, eeg, electric, stock)
  - Over models (ANN, BAYES, CART, KNN, and SVM)
  - For each iteration in the time series
- Thus we have rich data on classifier accuracy that tells us how models and hypotheses perform on data sets as the training data accumulates data from the time series.

Results

- Results were generated:
  - For hypotheses 0, 1, 2, and 3
  - Over multiple data sets (diabetes, eeg, electric, stock)
  - Over models (ANN, BAYES, CART, KNN, and SVM)
  - For each iteration in the time series
- Thus we have rich data on classifier accuracy that tells us how models and hypotheses perform on data sets as the training data accumulates data from the time series.