Traditional Approaches to Machine Learning

A brief overview of different Machine Learning Algorithms

Tilman Roeder

January 26, 2020

Today

Naive Bayes Classifiers

Support Vector Machines

Decision Trees

Section 1

Naive Bayes Classifiers

(Super Brief) Primer in Bayesian Statistic

Define conditional probabilities

$$P(X|Y)P(Y) = P(X \cap Y) \tag{1}$$

(Super Brief) Primer in Bayesian Statistic

Define conditional probabilities

$$P(X|Y)P(Y) = P(X \cap Y) \tag{1}$$

Which immediately gives Bayes Theorem

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} = \frac{P(Y|X)P(X)}{\sum_X P(Y|X)P(X)} \tag{2}$$

(Super Brief) Primer in Bayesian Statistic

Define conditional probabilities

$$P(X|Y)P(Y) = P(X \cap Y) \tag{1}$$

Which immediately gives Bayes Theorem

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} = \frac{P(Y|X)P(X)}{\sum_X P(Y|X)P(X)} \tag{2}$$

In Bayesian Statistics probabilities quantify our 'degree of believe'.
Bayes Theorem allows us to 'update' our believes (prior) based on new evidence (data) to obtain a new believe (posterior).

Naive Bayes Classifiers

- Given a set of possible labels $\{X_i\}$ and an observation $Y \in \{Y_i\}$

Naive Bayes Classifiers

- Given a set of possible labels $\{X_i\}$ and an observation $Y \in \{Y_i\}$
- We seek $P(X|Y)$

Naive Bayes Classifiers

- Given a set of possible labels $\{X_i\}$ and an observation $Y \in \{Y_i\}$
- We seek $P(X|Y)$
- Apply Bayes Theorem to compute posterior probability

Naive Bayes Classifiers

- Given a set of possible labels $\{X_i\}$ and an observation $Y \in \{Y_i\}$
- We seek $P(X|Y)$
- Apply Bayes Theorem to compute posterior probability
- Conditional probabilities are based on simple (naive) heuristics

Example: Spam Filtering

Given a message M consisting of words $(W_1, W_2, \ldots, W_n)$, and assuming words occurrences are independent we have:

$$P(S|W) = \frac{P(W|S)P(S)}{P(W)} = \frac{P(S)\prod_i P(W_i|S)}{\sum_{s\in\{S,\neg S\}} P(s)\prod_i P(W_i|s)}, \quad (3)$$

where $P(S)$ is the probability of the message being spam.

Example: Spam Filtering

Note that the previous expression is prone to numeric instability. We can fix this by computing the probabilities in *log* space:

$$\ln\left(\frac{1}{P(S|W)} - 1\right) = \sum_i \ln\left(P(W_i|\neg S)\right) - \ln\left(P(W_i|S)\right) \\ + \ln(P(\neg S)) - \ln(P(S)) \tag{4}$$

Example: Spam Filtering

What are $P(W|S)$, $P(W)$, and $P(S)$?

Example: Spam Filtering

What are $P(W|S)$, $P(W)$, and $P(S)$? This is where we invent heuristics:

- $P(W|S) =$ fraction of spam words that are W
- $P(W|\neg S) =$ fraction of ham words that are W
- $P(S) =$ fraction of spam received in total (or unbiased: $\frac{1}{2}$)

Example: Spam Filtering

What are $P(W|S)$, $P(W)$, and $P(S)$? This is where we invent heuristics:

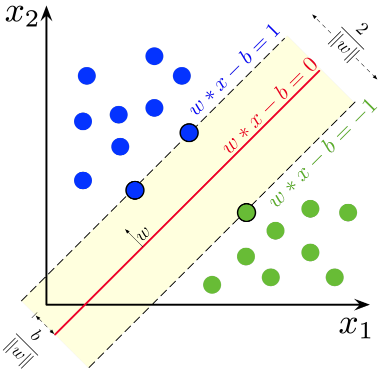
- $P(W|S) =$ fraction of spam words that are W
- $P(W|\neg S) =$ fraction of ham words that are W
- $P(S) =$ fraction of spam received in total (or unbiased: $\frac{1}{2}$)

These are very cheap to compute, and one of the oldest spam filters. Bayesian Classifiers are not as good as more advanced techniques. But for applications where speed and simplicity are important, they can be very competitive.

Section 2

Support Vector Machines

Linear Separable Data



(graphic by ZackWeinberg)

- SVMs classify data with binary labels
- Classification is achieved by drawing a hyperplane between the data

Support Vector Machine Classifier

A hyperplane P in $\mathbb{R}^n$ is defined by

$$\vec{\omega} \cdot \vec{x} - b = 0, \tag{5}$$

with $\vec{\omega}, \vec{x} \in \mathbb{R}^n$, $b \in \mathbb{R}$ and where $\cdot$ denotes an inner product.

Support Vector Machine Classifier

A hyperplane P in $\mathbb{R}^n$ is defined by

$$\vec{\omega} \cdot \vec{x} - b = 0, \tag{5}$$

with $\vec{\omega}, \vec{x} \in \mathbb{R}^n$, $b \in \mathbb{R}$ and where $\cdot$ denotes an inner product. We can now define the Support Vector Machine classifier:

$$\tilde{y}(\vec{x}) = \operatorname{sign}(\vec{\omega} \cdot \vec{x} - b). \tag{6}$$

Support Vector Machine Classifier

A hyperplane P in $\mathbb{R}^n$ is defined by

$$\vec{\omega} \cdot \vec{x} - b = 0, \tag{5}$$

with $\vec{\omega}, \vec{x} \in \mathbb{R}^n$, $b \in \mathbb{R}$ and where $\cdot$ denotes an inner product. We can now define the Support Vector Machine classifier:

$$\tilde{y}(\vec{x}) = \mathrm{sign}(\vec{\omega} \cdot \vec{x} - b). \tag{6}$$

Note that the classifier has $n + 1$ free parameters.

Hard Margin SVM

Consider a set of labeled points $\{(\vec{x}_i, y_i)\}$, where $y_i \in \{-1, +1\}$.

Hard Margin SVM

Consider a set of labeled points $\{(\vec{x}_i, y_i)\}$, where $y_i \in \{-1, +1\}$. Define two hyperplanes equidistant from P: P_1, P_{-1} as

$$\vec{\omega} \cdot \vec{x} - b = \pm 1. \tag{7}$$

Note that $P_{\pm 1}$ have a distance of $\frac{2}{|\vec{\omega}|}$.

Hard Margin SVM

Consider a set of labeled points $\{(\vec{x}_i, y_i)\}$, where $y_i \in \{-1, +1\}$. Define two hyperplanes equidistant from P: P_1, P_{-1} as

$$\vec{\omega} \cdot \vec{x} - b = \pm 1. \tag{7}$$

Note that $P_{\pm 1}$ have a distance of $\frac{2}{|\vec{\omega}|}$.
To find the SVM classifier we minimize $|\vec{\omega}|$, subject to the condition that P correctly classifies the data:

$$y_i(\vec{\omega} \cdot \vec{x}_i - b) \geq 1 \tag{8}$$

Hard Margin SVM

Consider a set of labeled points $\{(\vec{x}_i, y_i)\}$, where $y_i \in \{-1, +1\}$. Define two hyperplanes equidistant from P: P_1, P_{-1} as

$$\vec{\omega} \cdot \vec{x} - b = \pm 1. \tag{7}$$

Note that $P_{\pm 1}$ have a distance of $\frac{2}{|\vec{\omega}|}$.
To find the SVM classifier we minimize $|\vec{\omega}|$, subject to the condition that P correctly classifies the data:

$$y_i(\vec{\omega} \cdot \vec{x}_i - b) \geq 1 \tag{8}$$

Notice that the parameters $\vec{\omega}, b$ are completely determined by those $\vec{x}_i$ which are closest to P. These are called the 'support vectors'.

Soft Margin SVM

What happens when the data is not linearly separable?

Soft Margin SVM

What happens when the data is not linearly separable?
We can use a soft-margin classifier instead. This is done by modifying the loss function:

$$L = \frac{1}{n}\sum_{i} \max(0, 1 - y_i(\vec{\omega} \cdot \vec{x}_i - b)) + \lambda|\vec{\omega}| \tag{9}$$

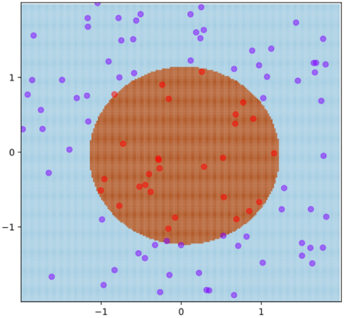
Soft Margin SVM

What happens when the data is not linearly separable?
We can use a soft-margin classifier instead. This is done by modifying the loss function:

$$L = \frac{1}{n} \sum_i \max(0, 1 - y_i(\vec{\omega} \cdot \vec{x}_i - b)) + \lambda|\vec{\omega}| \tag{9}$$

We now find $\vec{\omega}, b$ by minimizing L.

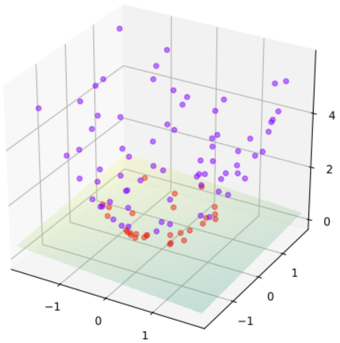
The Kernel Trick



(graphic by Shiyu Ji)

If the data is not linearly separable, a soft-margin SVM performs poorly.

The Kernel Trick



(graphic by Shiyu Ji)

However, there may be a map $\phi : \mathbb{R}^n \to \mathbb{R}^m$ to a higher dimensional space $\mathbb{R}^{m>n}$, in which the data is linearly separable.

The Kernel Trick

Consider the case where (like in the picture) class 1 lies inside the unit circle in $\mathbb{R}^2$.

The Kernel Trick

Consider the case where (like in the picture) class 1 lies inside the unit circle in $\mathbb{R}^2$.
Take the map

$$\phi(\vec{x}) = (x_1, x_2, x_1^2 + x_2^2). \tag{10}$$

The Kernel Trick

Consider the case where (like in the picture) class 1 lies inside the unit circle in $\mathbb{R}^2$.
Take the map

$$\phi(\vec{x}) = (x_1, x_2, x_1^2 + x_2^2). \tag{10}$$

In this new space, the data is separated by the plane

$$\hat{z} \cdot \vec{x} - 1 = 0. \tag{11}$$

The Kernel Trick

Consider the case where (like in the picture) class 1 lies inside the unit circle in $\mathbb{R}^2$.
Take the map

$$\phi(\vec{x}) = (x_1, x_2, x_1^2 + x_2^2). \tag{10}$$

In this new space, the data is separated by the plane

$$\hat{z} \cdot \vec{x} - 1 = 0. \tag{11}$$

We further define $k : \mathbb{R}^2 \to \mathbb{R}$ such that

$$k(\vec{x}, \vec{y}) = \phi(\vec{x}) \cdot \phi(\vec{y}) = \vec{x} \cdot \vec{y} + |\vec{x}|^2 |\vec{y}|^2. \tag{12}$$

The Kernel Trick

Consider the case where (like in the picture) class 1 lies inside the unit circle in $\mathbb{R}^2$.
Take the map

$$\phi(\vec{x}) = (x_1, x_2, x_1^2 + x_2^2). \tag{10}$$

In this new space, the data is separated by the plane

$$\hat{z} \cdot \vec{x} - 1 = 0. \tag{11}$$

We further define $k : \mathbb{R}^2 \to \mathbb{R}$ such that

$$k(\vec{x}, \vec{y}) = \phi(\vec{x}) \cdot \phi(\vec{y}) = \vec{x} \cdot \vec{y} + |\vec{x}|^2 |\vec{y}|^2. \tag{12}$$

k is called the 'Kernel' and allows us to easily compute inner products, without having to explicitly map $\vec{x}_i$ to the higher dimensional space.

The Kernel Trick (Note)

Keen observers might have noticed the following problem: how do we compute inner products with $\vec{\omega}$, without using $\phi(\vec{x})$ directly?

The Kernel Trick (Note)

Keen observers might have noticed the following problem: how do we compute inner products with $\vec{\omega}$, without using $\phi(\vec{x})$ directly?
The answer is as follows: assume $\vec{\omega}$ lies in the span of $\phi(\vec{x}_i)$, then we may write

$$\vec{\omega} \cdot \phi(\vec{x}) = \sum_i c_i y_i \phi(\vec{x}_i) \cdot \phi(\vec{x}), \tag{13}$$

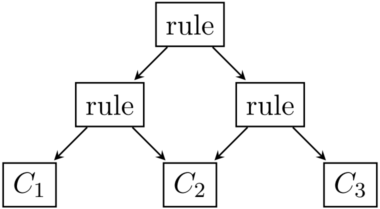
with $c_i \in \mathbb{R}$ and where y_i is included for mathematical convenience.

Section 3

Decision Trees

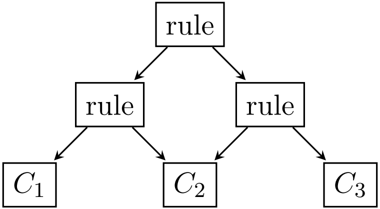
What is a Decision Tree

A decision tree is a **directed**, **acyclic graph**. The nodes in the graph represent decision points and all paths lead to a classification $C_i \in C$.



What is a Decision Tree

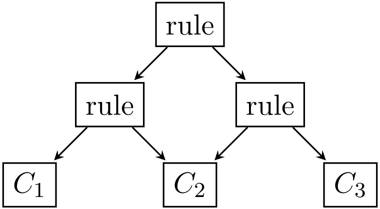
A decision tree is a **directed**, **acyclic graph**. The nodes in the graph represent decision points and all paths lead to a classification $C_i \in C$.



The decision nodes use (typically simple) rules to direct the data flow. The end nodes represent a classification.

What is a Decision Tree

A decision tree is a **directed**, **acyclic graph**. The nodes in the graph represent decision points and all paths lead to a classification $C_i \in C$.



The decision nodes use (typically simple) rules to direct the data flow. The end nodes represent a classification.
Note that the decision nodes can use simple rules (like $\vec{x}_i > \vec{y}_i$ for some i), or more complex rules (e.g. a SVM or Bayes Filter).

Building a Decision Tree

How do we construct such a tree?

Building a Decision Tree

How do we construct such a tree?
There are multiple options:

Building a Decision Tree

How do we construct such a tree?
There are multiple options:

- Write the rules by hand

Building a Decision Tree

How do we construct such a tree?
There are multiple options:

- Write the rules by hand
- Select a set of parameterized rules, and construct tree by choosing best separation achieved at each node

Random Forrest

We can build models with better performance by construction multiple trees:

Random Forrest

We can build models with better performance by construction multiple trees:

- Split the dataset into N disjoint subsets

Random Forrest

We can build models with better performance by construction multiple trees:

- Split the dataset into N disjoint subsets
- Build a decision tree from each subset

Random Forrest

We can build models with better performance by construction multiple trees:

- Split the dataset into N disjoint subsets
- Build a decision tree from each subset
- The trees can then be used to generate separate predictions for new data

Random Forrest

We can build models with better performance by construction multiple trees:

- Split the dataset into N disjoint subsets
- Build a decision tree from each subset
- The trees can then be used to generate separate predictions for new data
- These predictions are then combined (e.g. by majority vote)

Random Forrest

We can build models with better performance by construction multiple trees:

- Split the dataset into N disjoint subsets
- Build a decision tree from each subset
- The trees can then be used to generate separate predictions for new data
- These predictions are then combined (e.g. by majority vote)

In practice, random forests are good and fast predictors, that are competitive with neural networks (depending on specific task).