Nonparametric Learning

CS156

Chris Pollett

Dec 8, 2014

Outline

- Nearest Neighbors
- Quiz
- SVM

Non-parametric Learning

- Neural Networks use the training data to estimate a fixed set of parameters $\vec{w}$.
- A learning model that summarizes data with a set of parameters of a fixed size is called a **parametric model**.
- No matter how much data is thrown at the model it doesn't changes its mind about how many parameters to learn from.
- A **nonparametric** is one that cannot be characterized by a bounded set of parameters.
- For example, suppose our method of learning was to make a big table of all the labeled examples given.
- When someone asks for a the label of some input x, we look it up in the table, if found, we return the label there, otherwise we output "don't know".
- This kind of learning is called **instance-based learning** or **memory-based learning** and obviously does not generalize well.
- It is nonparametric though in that as the training set changes or grows the number of things we are keeping track of in order to give a result also changes.

Nearest Neighbor Model

- Let's now consider a better nonparametric learning model.
- We assume we have some kind of binary classification problem.
- Suppose when given a query $\vec{x}_q$, we lookup the k nearest neighbors to $\vec{x}_q$ with respect to some kind of distance function.
- Denote these neighbors by $NN(k, \vec{x}_q)$.
- To classify $\vec{x}_q$, we take the majority vote of these k neighbors.
- This is nonparametric as that we need to keep all of the training data in order to run this algorithm -- we don't learn some fixed parameters and then forget the training data.
- If we choose k too small we can get overfitting issues (outliers can cause mis-classification) and too big we can get underfitting issues (for example, if we chose k the whole data set size than we are just doing majority vote on the training set).
- We can use different can and perform tests to see which works best using cross-validation techniques. Usually, the result will be somewhere between $k = 1$ and square root of the data set size.

Choice of Distance Function

- How do we say how far apart a query point, $\vec{x}_q$, is from an example point, $\vec{x}_e$?
- If the vectors are from $\mathbb{R}^n$, then we could use an L^p-norm:
$$L^p\left(\vec{x}_q, \vec{x}_e\right) = \left(\sum_i |x_{e,i} - x_{q,i}|^p\right)^{\frac{1}{p}}$$
- When $p = 2$ this is the usual Euclidean distance, for $p = 1$ it is Manhattan distance.
- If the coordinates each take on Boolean values then we can use the number of coordinates that differ in value. I.e., the Hamming distance.
- For documents, we can use edit distance to say how close they are, or we can make a vector using distinct columns for each word and frequency counts of that word as the size of that component.
- Notice the range of values of each coordinate can be quite different.
- Frequently, a normalization step is applied coordinate-wise so that each coordinate has roughly the same weight.
- For example a $x_{j,i}$ value might be mapped to $\frac{x_{j,i} - \mu_i}{\sigma_i}$ where μ_i is the mean over the training set of the ith component values, and σ_i is its standard deviation.

Making Lookups of Nearest Neighbors Efficient

- One way to look up $NN\left(k, \vec{x}_q\right)$ is to cycle overall the data set and find the k closest points.
- If the dataset has N elements this is an $O(N)$ algorithm.
- To speed this up, we can use a $k-d$ tree.
- This tree is constructed as follows, we cycle over each coordinate (repeating coordinates until we get done to single examples), we compute the median value for that coordinate, we put the examples less than the median in the left branch those bigger in the right branch, etc.
- Using $k-d$-trees the look up time can be brought down to $O(\log N)$.
- Another technique is to use a locality sensitive hash function: We choose m coordinates $(c_1, ..., c_m)$ from the set of all coordinates for our examples at random. For each of these coordinates we have a hash table h_i of bin's of values of examples with respect to this coordinate c_i.
- When we get a new item to classify we look up which examples in the set were close to our query with respect to each of the c_i. Then pick the k closest of these using linear search.

Quiz

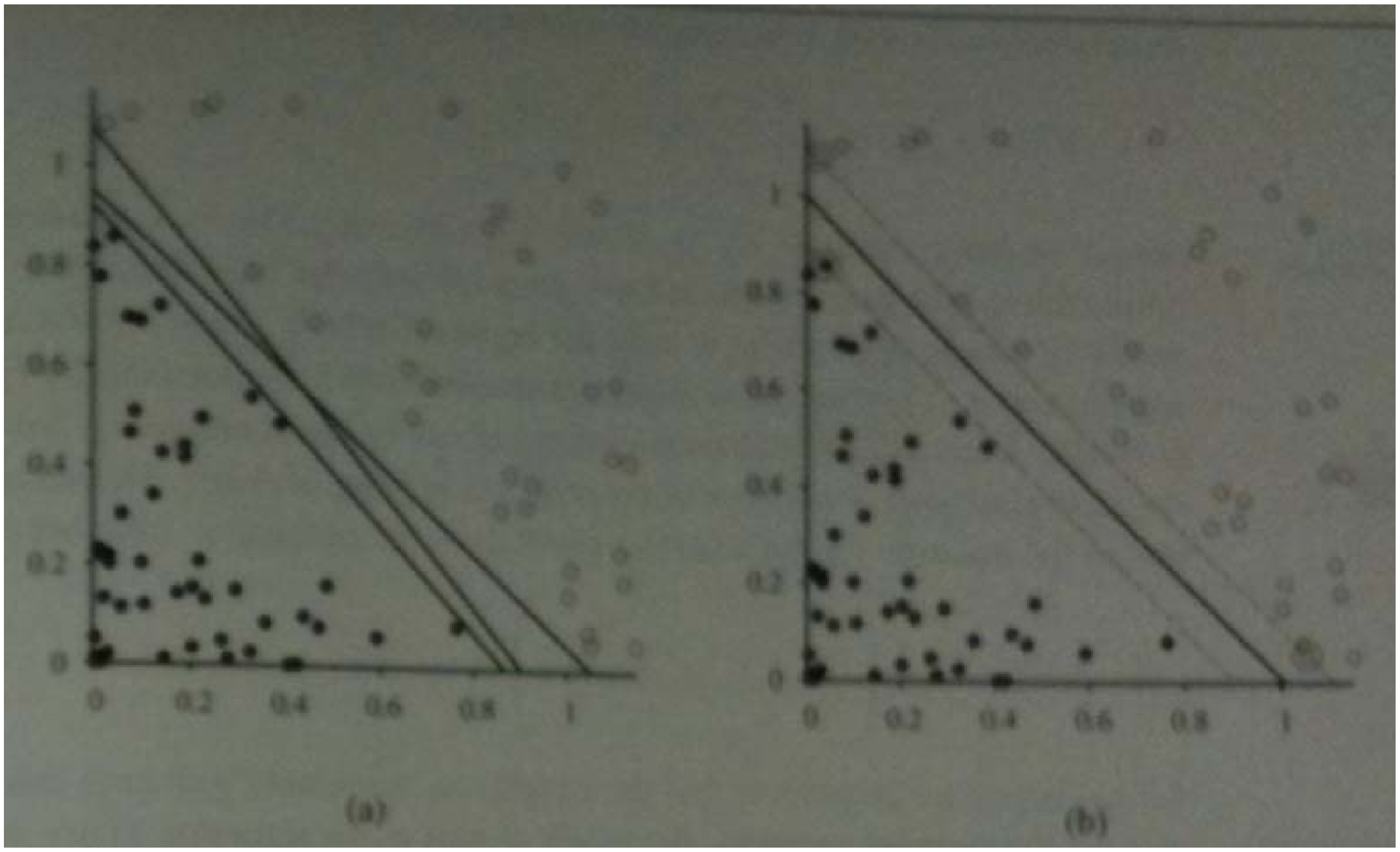
Which of the following is true?

1. Perceptron networks always have a hidden layer.
2. The information gain of an attribute A is the entropy of the data before testing on that attribute less the expected entropy left after testing on that attribute.
3. Our perceptron training algorithm assumed the function g in the perceptron was a step function.

Support Vector Machines

- We next look at one way to make our perceptron model into a nonparametric learning scheme
- The model we get called a **support vector machine** or SVM.
- It turns out not only are SVMs more robust in terms of the classifier they build, they also can be used to solve problems not solvable by traditional perceptrons.

Building a Better Separator



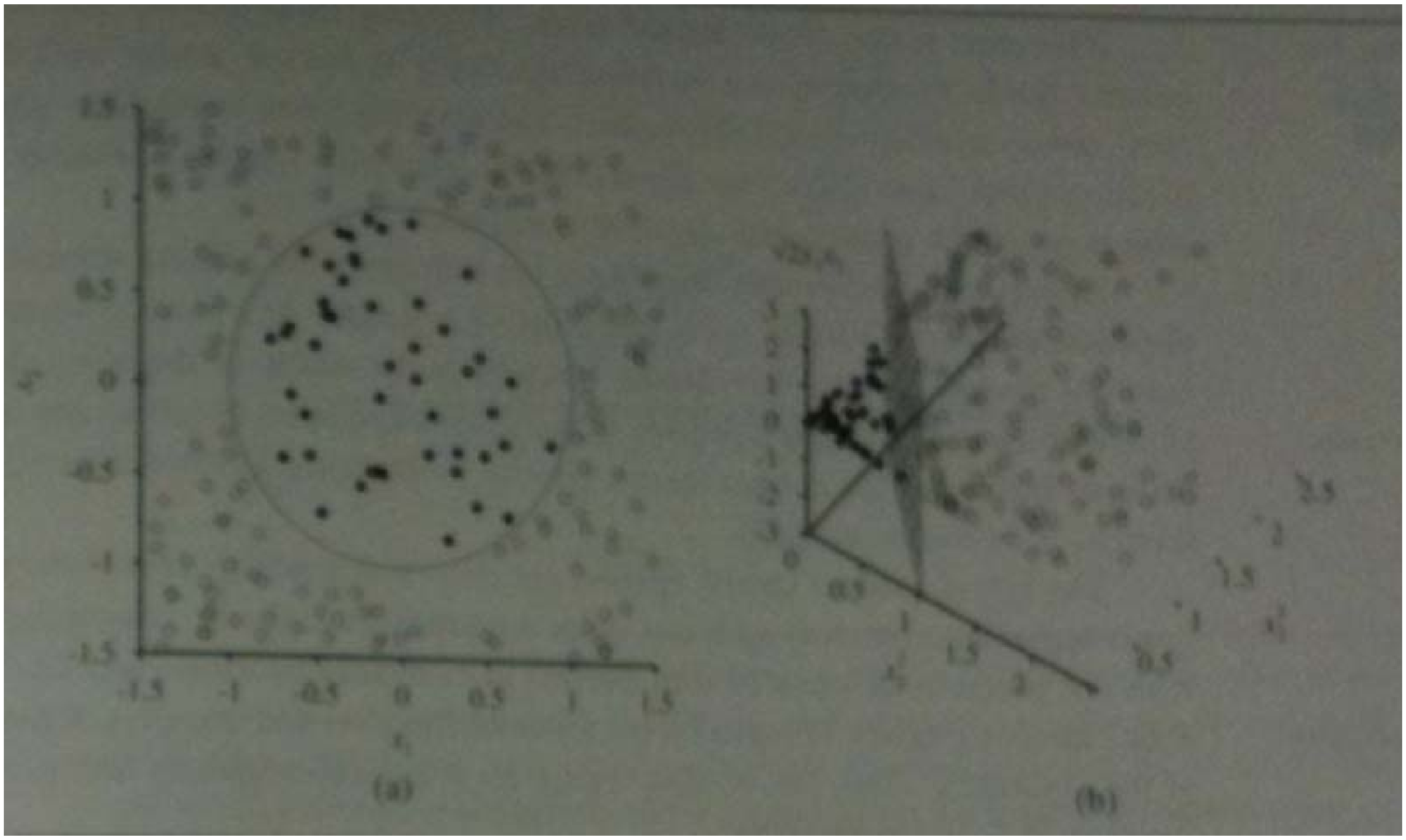
- If we look at the left hand image above, any one of the three lines separates all of the black points from the white points.
- So any one of the three lines minimizes the empirical loss, the number of training examples mis-classified, and depending on the data result from our algorithm for learning weights. The innermost one though is very close to a lot of black dots and one might expect that it would be less robust in classifying new queries.
- The diagram on the right shows a **maximum margin separator**. Here the **margin** is the width of the area bounded by dashed lines in the figure. It is defined as twice the distance from the separator to the nearest example point.
- Using a maximum margin separator will tend to minimize generalization loss (it should tend to make fewer errors on new data).
- SVM are essentially perceptrons where the weights are set so that a maximum margin separator is calculated. SVMs also unlike perceptrons allow for a mapping function which maps problems from a

lower dimensional space where the data might not be linearly separable to a higher dimensional space where it can be.

Computing Maximum Margin Separator

- The equation of a hyperplane is $\vec{w} \cdot \vec{x} + b = 0$ for some constant vector weight vector $\vec{w}$ and constant weight value b.
- Suppose our training examples are of the form $\left(\vec{x}_j, y_j\right)$ where y_j is 0 or 1.
- The book magically gives the equation for the maximal separator as a **quadratic programming** problem of maximizing the function
$$\text{argmax}_{\vec{\alpha}}\left(\sum_j \alpha_j - \frac{1}{2}\sum \alpha_j\alpha_k y_j y_k\left(\vec{x}_j \cdot \vec{x}_k\right)\right)$$
subject to $\alpha_j \geq 0$ and $\sum_j \alpha_j y_j = 0$.
- There are good software packages for this.
- Once $\vec{\alpha}$ has been computed, $\vec{w}$ can be computed via $\vec{w} = \sum_j \alpha_j \vec{x}_j$.
- The above function is convex and has a single global maximum. Also, the data only enters the expression in the form of a dot product of pairs of points.
- A classifier can be built out of $\vec{\alpha}$ directly using the equation:
$$h(\vec{x}) = sign\left(\sum_j \alpha_j y_j\left(\vec{x} \cdot \vec{x}_j\right) - b\right)$$
which also only uses the data in a dot product.
- An important property of the α_j's is that they are 0 usually except the **support vectors** -- the closest vectors to the separator (typically in each dimension).

SVMs in General



- A general SVM classifier replaces the data set $\vec{x}_j$ with the result of mapping it into a different space with respect to some vector-valued function $\vec{F}$.
- That is the classifier, we get is of the form
$$h(\vec{x}) = sign\left(\sum_j \alpha_j y_j \left(\vec{F}(\vec{x}) \cdot \vec{F}\left(\vec{x}_j\right)\right) - b\right)$$
- For example, in the above on the left hand side we have data which can't be separated by a linear equation in the plane.
- We map the data to 3D, via the function $\vec{F}(x_1, x_2) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$.
- We solve the quadratic programming problem in this new space and get a separator, giving the picture on the right.
- Most SVM packages come with a list of functions to try. Using F so dot products work out correctly so this works is called the **kernel trick**.