Nearest Neighbor Classifiers

Oct 5 2009
When classifying an email to be spam or non-spam, one possible strategy is to look at existing emails that are similar to the email in question, and see if they are spams or not – this is the philosophy behind what we call nearest neighbor classifiers.
Nearest Neighbor Classifier

- Store all training examples $S$
- Given a new example $x$ to be classified, search for the training example $<x^i, y^i>$ whose $x^i$ is *most similar* (or closest) to $x$, and predict $y^i$
Similarity/distance

• How do we measure the similarity or distance between the two examples?

• The most commonly used measure is Euclidean distance

\[
D(x, x^i) = \| x - x^i \|^2 \\
= (x - x^i)^T (x - x^i) \\
= \sum_j (x_j - x^i_j)^2
\]
Decision Boundaries: The Voronoi Diagram

• Given a set of points, a Voronoi diagram describes the areas that are nearest to any given point.
• These areas can be viewed as zones of control.
Voronoi diagram

• Demo
  http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/index.html.en
Decision Boundaries: Subset of the Voronoi Diagram

- Each example controls its own neighborhood
- Create the voronoi diagram
- Decision boundary are formed by only retaining these line segments separating different classes.
- The more training examples we have stored, the more complex the decision boundaries can become
Decision Boundaries

With large number of examples and noise in the labels, the decision boundary can become nasty!

It can be bad some times – note the islands in this figure, they are formed because of noisy examples.

If the nearest neighbor happens to be a noisy point, the prediction will be incorrect!

How to deal with this issue?
K-Nearest Neighbor

Example:

$k = 3$

Find the $k$-nearest neighbors and have them vote.

It is typical to use an odd number for $k$ to avoid ties.
K-Nearest Neighbor

What if we set $k=8$ (or 7)?

We will always predict green because regardless of the location of the point, the k-nearest neighbors will contain more green points than red, simply because there are more greens than reds.
Effect of $k$

Different $k$ values give different results:

Larger $k$ produces smoother boundaries, why?
- The impact of class label noises canceled out by one another

But when $k$ is too large, what will happen?
- Oversimplified boundaries, say $k=N$, we always predict the majority class

Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)
Question: how to choose k?

• Can we choose k to minimize the mistakes that we make on training examples (*training error*)?

• Let’s first define training error
  – Given a training set $S$, learn a classifier $h$
  – $\varepsilon_{tr}=0$
  – for every example $(x^i, y^i)$ in $S$
    • If $h(x^i) \neq y^i$ $\varepsilon_{tr}++$

• What is training error of 1-nearest neighbor?
  – $\varepsilon_{tr}=0$
  – Because for any training example, its nearest neighbor in $S$ is always going to be itself
Question: how to choose k?

• Can we choose k to minimize the mistakes that we make on test examples (test error)?

• Let’s first define test error
  – Given a training set $S$ and a separate set $T$, learn a classifier $h$
    on $S$ (without looking at $T$)
  – for every example $(x^i, y^i)$ in $T$
    • If $h(x^i) \neq y^i$ then $\varepsilon_{te}++$

• How do $\varepsilon_{tr}$ and $\varepsilon_{te}$ change as we change the value of k?
For the previous example

Possibly a good choice for $k$
Model Selection

- Choosing k for k-nn is just one of the many *model selection* problems we face in machine learning.
- **Model selection** is about choosing among different models:
  - linear regression vs. quadratic regression
  - choosing k-nn over LTU
  - heavily studied in machine learning, crucial importance in practice
- If we use training error to select models, we will always choose more complex ones.

![Diagram showing test error and training error with underfitting and overfitting](image)
- **Test error**
- **Training error**
- **Increasing Model complexity** (e.g., as we decreases k for knn)
- **Underfitting**
- **Overfitting**
Use a **Validation** Set

- We can keep part of the labeled data apart as validation data
- Evaluate different k values based on the prediction accuracy on the validation data
- Choose k that minimize validation error

Validation can be viewed as another name for testing, but the name **testing** is typically reserved for final evaluation purpose, whereas **validation** is mostly used for model selection purpose.
The impact of validation set size

- If we only reserve one point in our validation set, should we trust the validation error as a reliable estimate of our classifier’s performance?
- No. The larger the validation set, the more reliable our model selection choices are
- When the total labeled set is small, we might not be able to get a big enough validation set – leading to unreliable model selection decisions
K-fold Cross Validation

- Split the training set into $K$ equal-sized subsets
- Perform learning/testing $K$ times
  - Each time reserve one subset for validation set, train on the rest

Example: A 5-fold cross validation

Train on $S2, S3, S4, S5$, test on $S1 \quad \varepsilon_1$

Train on $S1, S3, S4, S5$, test on $S2 \quad \rightarrow \quad \varepsilon_2$

\[ \vdots \]

Train on $S1, S2, S3, S4$, test on $S5 \quad \rightarrow \quad \varepsilon_5$

\[ \varepsilon = \frac{1}{5} \sum_{i=1}^{5} \varepsilon_i \]
Leave-one-out Cross Validation

• If we set $K = n = |S|$, we end up with what we call leave-out-out cross validation
• Each time we leave one example out for validation

What is the leave-one-out error of 1-nearest neighbor on this data set?
Practical issues with KNN

• Suppose we want to build a model to predict a person’s shoe size
• Use the person’s height and weight to make the prediction
• P1: (6’ , 175), P2: (5.7, 168), PQ: (6.1’, 170)

\[ D(PQ, P1) = \sqrt{0.1^2 + 5^2} \approx 5 \quad D(PQ, P2) = \sqrt{0.4^2 + 2^2} \approx 2.04 \]

• There is a problem with this what is it?

Because weight has a much larger range of values, the differences look bigger numerically.

Features should be normalized to have the same range of values (e.g., [0, +1]), otherwise features with larger ranges will be treated as more important.
Practical issues with KNN

- Our data may also contain the GPAs.
- Should we include this attribute into the calculation?
- When collecting data, people tend to collect as much information as possible regardless of whether they are useful for the question in hand.
- Recognize and remove such attributes when building your classification models.
Other issues

• It can be computationally expensive to find the nearest neighbors!
  – Speed up the computation by using smart data structures to quickly search for approximate solutions

• For large data set, it requires a lot of memory
  – Remove unimportant examples
Final words on KNN

• KNN is what we call lazy learning (vs. eager learning)
  – Lazy: learning only occur when you see the test example
  – Eager: learn a model before you see the test example, training examples can be thrown away after learning

• Advantage:
  – Conceptually simple, easy to understand and explain
  – Very flexible decision boundaries
  – Not much learning at all!

• Disadvantage
  – It can be hard to find a good distance measure
  – Irrelevant features and noise can be very detrimental
  – Typically can not handle more than 30 attributes
  – Computational cost: requires a lot computation and memory