CS 340: Lec. 5 - K-Nearest Neighbors

AD

January 2011

How to Select K

- We want to select K so as to obtain a small classification error on the test data but, in real-world applications, we cannot evaluate this error on the test set!
- A simple idea to evaluate the error rate consists of splitting the training data into two blocks: a block used as training data and the other block known as validation set.
- **Example**: Assume you are given $\left\{\mathbf{x}^i, y^i\right\}_{i=1}^{N}$ training data, then only $N_{train} < N$ data, say $\left\{\mathbf{x}^i, y^i\right\}_{i=1}^{N_{train}}$ are used as training data whereas the remaining $N_{valid} = N - N_{train}$ data $\left\{\mathbf{x}^i, y^i\right\}_{i=N_{train+1}}^{N}$ are used to assess the performance of the classifier using

$$\underbrace{Err}_{\text{Error rate}} = \frac{1}{N_{valid}} \sum_{i=N_{train+1}}^{N} \mathbb{I}\left(\widehat{y}\left(\mathbf{x}^i\right) \neq y^i\right).$$

- We compute Err for various values of K and select the one which minimizes Err.
- This is a very common, general and useful procedure!

Cross-Validation

- If N is small, this technique is unreliable as the model won't have enough data to train on, and we won't have enough data to make a reliable estimate of the future performance.
- A simple and popular solution to this is M-fold **cross validation** (CV). We split the training data into M folds then, for each fold $k \in \{1, 2, ..., M\}$, we train on all the folds but the k'th, and test on the k'th, in a round-robin fashion to estimate $Err = \frac{1}{M}\sum_{k=1}^{M} Err_k$. N-fold CV is called **leave-one-out CV**.

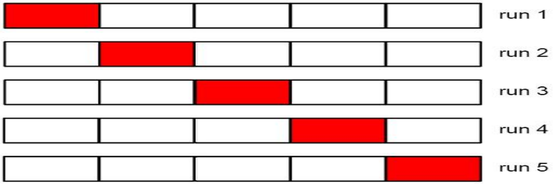

Figure: 5-fold cross validation

Cross-Validation for K-NN

- Illustration of the 10-fold CV for K-NN.

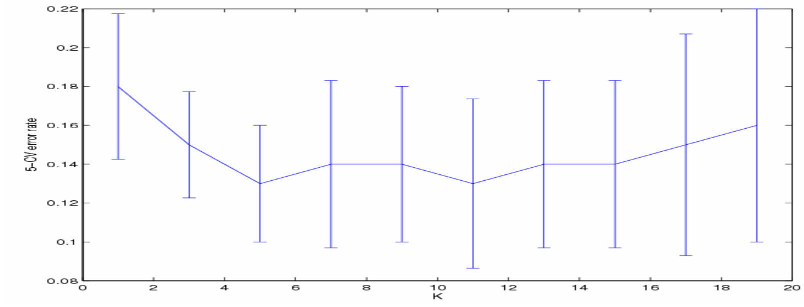

Figure: 10 fold-CV error rate as a function of K

- In this case, we would pick $K = 11$.

Problems with K-NN

- Can be slow to find nearest neighbor in high-dimensional space.
- Need to store all the training data, so takes a lot of memory.
- Need to specify the distance function.
- Does not give probabilistic output.
- Difficult to interpret.
- Curse of dimensionality...

Reducing Running Time of K-NN

- Takes $O(Nd)$ to find the exact nearest neighbor
- Use a branch and bound technique where we prune points based on their partial distances

$$D_r(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{r} (x_k - x_k')^2 .$$

- Structure the points hierarchically into a kd-tree (does offline computation to save online computation).
- Use locality sensitive hashing (a randomized algorithm).
- Various heuristic algorithms have been proposed to prune/edit/ condense "irrelevant" points that are far from the decision boundaries.
- Later we will study sparse kernel machines that give a more principled solution to this problem.

A Probabilistic Version of K-NN

- A classification function returns a single best guess $\widehat{y}(\mathbf{x})$ of y given an input $\mathbf{x}$.
- A probabilistic classifier returns a probability distribution over outputs given an input:

$$\Pr(\widehat{y}(\mathbf{x}) = i | \mathbf{x}) \geq 0$$
$$\sum_{i=1}^{C} \Pr(\widehat{y}(\mathbf{x}) = i | \mathbf{x}) = 1.$$

- For $C = 2$ if $\Pr(\widehat{y}(\mathbf{x}) = i | \mathbf{x}) \approx 0.5$ (very uncertain), the system may choose not to classify as $0/1$ and instead ask for human help.
- Useful to fuse different predictions $\widehat{y}(\mathbf{x})$ of y.

A Basic Probabilistic K-NN

- We can compute the empirical distribution over labels in the K-neighborhood; i.e. we set

$$\Pr\left(\widehat{y}\left(\mathbf{x}\right)=i\middle|\,\mathbf{x}\right)=\frac{1}{K}\sum_{\{j:\mathbf{x}^j \text{ is one of the } K\text{-NN of } \mathbf{x}\}}\mathbb{I}\left(y^j=i\right)$$

- **Example**: let $C=3$, $K=5$ and the 5 nearest neighbor of $\mathbf{x}$ have labels $\{2,3,3,3,2\}$ then

i	1	2	3
$\Pr\left(\widehat{y}\left(\mathbf{x}\right)=i\middle|\,\mathbf{x}\right)$	$\frac{0}{5}$	$\frac{2}{5}$	$\frac{3}{5}$

A Basic Probabilistic K-NN

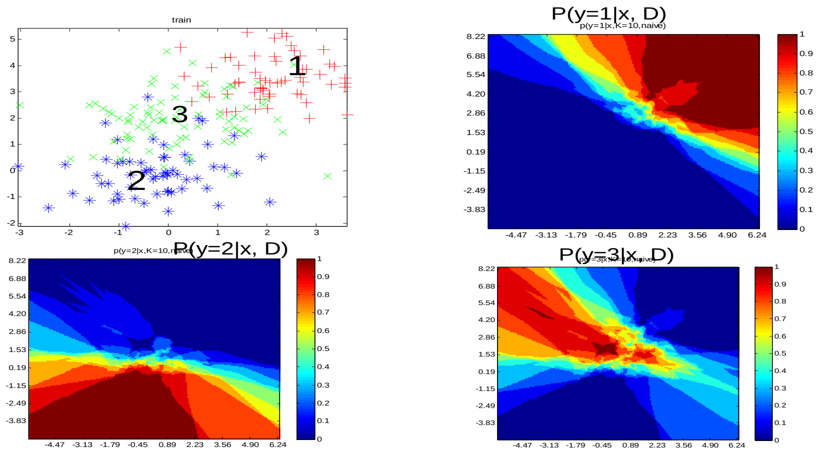

Figure: Illustration of the outpout of a probabilistic KNN classifier

Smoothing Empirical Frequencies

- The empirical distribution will often predict 0 probability due to sparse data.
- We can add pseudo counts to the data and then normalize.
- **Example**: let $C = 3$, $K = 5$ and the 5 nearest neighbor of $\mathbf{x}$ have labels $\{2, 3, 3, 3, 2\}$ then if we add pseudo-counts to the data and then normalize, we obtain

i	1	2	3
$\Pr\left(\widehat{y}\left(\mathbf{x}\right) = i \middle| \mathbf{x}\right)$	$\frac{0+1}{5+1+1+1} = \frac{1}{8}$	$\frac{2+1}{5+1+1+1} = \frac{3}{8}$	$\frac{3+1}{5+1+1+1} = \frac{4}{8}$

- This approach is related to Bayesian statistics.

Softmax (multinomial logit) Function

- We can also "soften" the empirical distribution so it spreads its probability mass over unseen classes.
- Define the softmax with inverse temperature $\beta > 0$

$$\Pr\left(\widehat{y}\left(\mathbf{x}\right) = i \middle| \mathbf{x}\right) = \frac{\exp\left(\beta \quad \pi_i\right)}{\sum_{k=1}^{C} \exp\left(\beta \quad \pi_k\right)}$$

where

$$\pi_i = \frac{1}{K} \sum_{\{j:\mathbf{x}^j \text{ is one of the } K\text{-NN of } \mathbf{x}\}} \mathbb{I}\left(y^j = i\right)$$

- Big β = cool temp = spiky distribution.
- Small β = high temp = uniform distribution.

Curse of Dimensionality for K-NN

- To explain the curse, consider using a KNN classifier where the inputs are uniformly distributed in the unit hypercube $[0,1]^d$.
- Suppose we want to take our decision for a test point $\mathbf{x}$ by "growing" a hypercube around $\mathbf{x}$ until it contains a desired fraction s of the training data points.
- The expected edge length of this cube will be $e_d(s) = s^{1/d}$ as $e(s)^d = s$.
- If $d = 10$ and we want to base our estimate on 1% of the data, we have $e_{10}(0.01) = 0.01^{1/10} \approx 0.63$ so we need to extend the cube 63% along each dimension around $\mathbf{x}$. Since the entire range of the data is only 1 along each dimension, the method is no longer very local, despite the name "nearest neighbor".

Curse of Dimensionality for K-NN

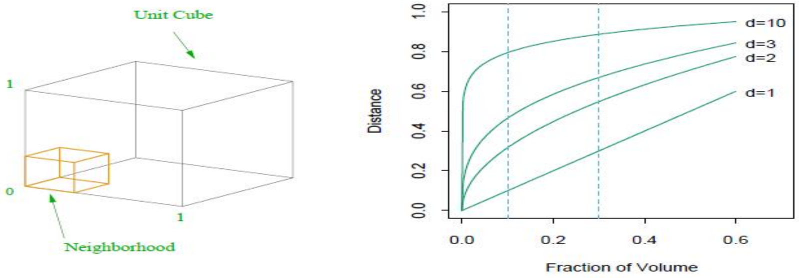

Figure: Illustration of the curse of dimensionality

Limiting the Curse of Dimensionality

- **Feature selection**: eliminate some of the “irrelevant” features x_i; e.g. the car you drive might not be a good indicator whether you have blue eyes or not.
- **Dimension reduction**: find a low-dimensional manifold on which the data lies, and measure distance in that subspace.

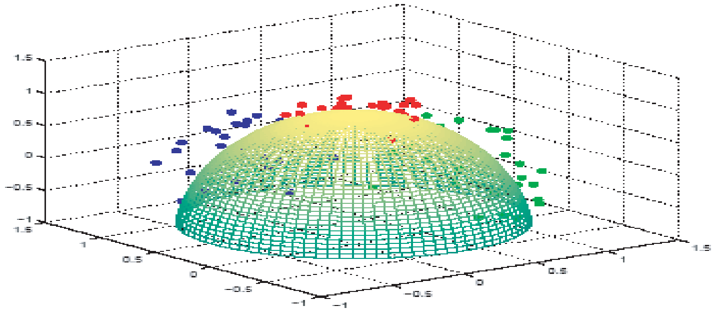

Figure: Simulated data in three classes, near the surface of a half-sphere