8001652 Introduction to Pattern Recognition. Lecture 9: k-Nearest neighbors classification

Jussi Tohka

jussi.tohka@tut.fi

Institute of Signal Processing

Tampere University of Technology

KNN for density estimation

- During the previous lecture, we have seen that one of the biggest problems in the density estimation with Parzen windows is that we have to supply a parameter h_n specifying the window length.
- A potential remedy to the problem is to let the cell volume V_n to be function of training data. Note that the cell does not need to be a hypercube.
- For example, to estimate $p(\mathbf{x})$ from the training samples or prototypes, we can center a cell about $\mathbf{x}$ and let it grow until it captures k_n samples - called the k_n nearest neighbors (KNN) of $\mathbf{x}$.
- k_n is some specified function of n.

KNN for density estimation

- If the density is high near $\mathbf{x}$, the cell will be relatively small, which leads to good resolution.
- If the density is low, the cell will be larger.
- Hence, regions of high density are more accurately modeled than the regions of low density.
- If we take $p_n(\mathbf{x}) = \frac{k_n}{nV_n}$, we want k_n to go infinity as n goes to infinity.
- We also want to k_n to grow sufficiently slowly so that V_n will shrink to zero.
- These are the two conditions required for the convergence of KNN density estimates.

KNN for a-posteriori estimation

- The KNN and Parzen techniques can also be used for estimation of a-posteriori probabilities $P(\omega_i|\mathbf{x})$ from a set of n labeled samples.
- Suppose that we place a cell of volume V around $\mathbf{x}$ and capture k samples, k_i of which turn out to be labeled ω_i.
- An estimate for the joint probability is $p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV}$.
- Hence, we can estimate $P(\omega_i|\mathbf{x})$ by

$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{k}.$$

The nearest-neighbor rule

- Denote by $\mathcal{D}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ a set of labeled *prototypes* or training samples.
- Let $\mathbf{x}^*$ be the prototype nearest to $\mathbf{x}$.
- Then *the nearest-neighbor (NN) rule* for classifying $\mathbf{x}$ is to assign it the label associated with $\mathbf{x}^*$.

The NN rule - properties

- The NN-rule is suboptimal. It will usually lead to greater classification error than the lowest possible, the Bayes risk. (BTW: Note that we now on consider only zero-one loss functions.)
- However, provided that we have an infinite amount of training data, the error rate of the NN rule is never worse than twice the Bayes risk.

The NN rule - why it works

- Despite of suboptimality, the asymptotic performance of the NN rule is good and the rule is simple. Let us explain (heuristically) why it is so.
- Note that the label θ^* associated with the nearest neighbor of $\mathbf{x}$ is a random variable. The probability that $\theta^* = \omega_i$ is just a-posteriori probability $P(\omega_i|\mathbf{x}^*)$.
- Now if $\mathbf{x}$ is very close to $\mathbf{x}^*$, then we can assume $P(\omega_i|\mathbf{x}^*) \approx P(\omega_i|\mathbf{x})$. Assuming unlimited training data $\mathbf{x}$ will be very close to $\mathbf{x}^*$.
- If the labeling of training samples is made correctly then the value of θ^* is the label that maximizes a-posteriori probability $P(\omega|\mathbf{x}^*)$ and in most cases it also maximizes $\mathbf{x}$.

The NN rule - geometry

- Recall that the effect of the decision rule α related to a classifier is to divide the feature space into the decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_c$ corresponding to the c classes:

$$\mathcal{R}_i = \{\mathbf{x} : \alpha(\mathbf{x}) = \omega_i\}.$$

- The NN decision rule divides the feature space into polyhedral cells, so-called *Voronoi tessellation* or *Voronoi diagram*.

The NN rule - geometry

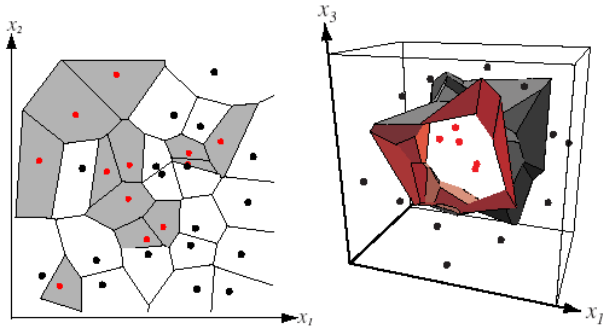

FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The KNN rule

- Generalization of the NN rule.
- The k_n nearest neighbors rule: Given a set of training samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a test point $\mathbf{x}$, find k training points closest to $\mathbf{x}$, $\mathbf{x}_1^*, \ldots, \mathbf{x}_k^*$. Collect the labels associated $\theta_1^*, \ldots, \theta_k^*$ and classify $\mathbf{x}$ to the class which has the greatest number of representatives in $\theta_1^*, \ldots, \theta_k^*$.
- In other words, the classification is performed by taking a majority vote among k nearest neighbor of $\mathbf{x}$.

The KNN rule

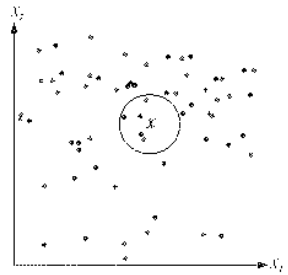

FIGURE 4.15. The k-nearest-neighbor query starts at the test point $\mathbf{x}$ and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point $\mathbf{x}$ would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The KNN rule - performance

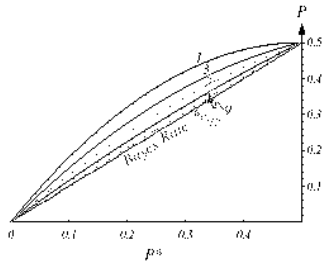

FIGURE 4.16. The error rate for the k-nearest-neighbor rule for a two-category problem is bounded by $C_k(P^*)$ in Eq. 54. Each curve is labeled by k; when $k = \infty$, the estimated probabilities match the true probabilities and thus the error rate is equal to the Bayes rate, that is, $P = P^*$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The KNN rule - computational complexity

- The computational load of the NN classification of a single test point is $O(nd)$, where d is the dimension of feature vectors and n is the number of training samples.
- For the KNN rule, if $k \ll n$, the complexity remains the same.
- $O(nd)$ is a lot of time, particularly if n is large.
- The difference to most other techniques for classification is that with KNN training points are needed during the classification when usually they are needed only during the training.
- Since training is performed only once and the classification many times, we are more concerned about the time consumption of classification.

The KNN rule - computational complexity

- However, there are many techniques currently for speeding up KNN classifiers:
- Partial distances
- Prestructuring prototypes
- Editing prototypes

Metrics

- With the KNN rule, we need to be able to compute the distance between two points $\mathbf{a}$ and $\mathbf{b}$ - the test point and a training point.
- The distance can be Euclidean:

$$d(\mathbf{a},\mathbf{b}) = ||\mathbf{a}-\mathbf{b}|| = \sqrt{\sum_{i=1}^{d}(a_i-b_i)^2}$$

- Other distance functions or metrics can be useful.

Metric - the definition

- Formally, a metric is a function $d(\cdot,\cdot)$ from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$.
- For all vectors $\mathbf{a},\mathbf{b},\mathbf{c}$ a metric d must satisfy
  1. $d(\mathbf{a},\mathbf{b}) \geq 0$
  2. $d(\mathbf{a},\mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$.
  3. $d(\mathbf{a},\mathbf{b}) = d(\mathbf{b},\mathbf{a})$
  4. $d(\mathbf{a},\mathbf{b}) + d(\mathbf{b},\mathbf{c}) \geq d(\mathbf{a},\mathbf{c})$.
- The only one of these properties which we occasionally want to sacrifice is the property 2.

Metric - examples

- Minkowski metric:

$$L_m(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d}(|a_i - b_i|^m)^{1/m}$$

- $L_\infty(\mathbf{a}, \mathbf{b}) = \max_i |a_i - b_i|$
- Tanimoto-metric between two sets A, B :

$$d_{Tanimoto}(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$