1 Nearest Neighbor Classification

1.1 Model

In classification, it is reasonable to conjecture that data points that are sufficiently close to one another should be of the same class. For example, in fruit classification, perturbing a few pixels in an image of a banana should still result in something that looks like a banana. The k-nearest-neighbors classifier is based on this observation. To train this classifier, we simply store our data somewhere.\footnote{Sometimes we store the data in a specialized structure called a \textit{k-d tree}. This data structure is out of scope for this course, but it usually allows for faster (average-case $O(\log n)$) nearest neighbors queries.} To predict on a test data point $x$, we compute the $k$ closest training data points to $x$, where “closeness” can be quantified in some distance function such as Euclidean distance - these are the $k$ nearest neighbors to $x$. We then find the most common class $y$ among these $k$ neighbors and classify $x$ as $y$ (that is, we perform a majority vote). For binary classification, $k$ is usually chosen to be odd so we can break ties cleanly. In general, in order to select $k$ we use cross-validation.

![1-NN Decision Region for 3 Exemplars](image)

Figure 1: The decision function induced by 1-NN with 3 classes. Points in a region shaded a certain color will be classified as that color.

Nearest neighbors can produce very complex decision functions - Figure 2 shows one possibility.
1.2 Properties

**Computational complexity:** We require $O(n)$ space to store a training set of size $n$. There is no runtime cost during training if we do not use specialized data structures to store the data. However, predictions take $O(n)$ time, which is costly. There has been research into approximate nearest neighbors (ANN) procedures that quickly find an approximation for the nearest neighbor - some common ANN methods are *Locality-Sensitive Hashing* and algorithms that perform dimensionality reduction via randomized (Johnson-Lindenstrauss) distance-preserving projections.\(^2\)

**Bias/Variance:** Using 1-NN allows for perfect separation of the training data (each point is its own nearest neighbor). It has low bias but high variance, as 1-NN depends only on one nearest neighbor and thus is sensitive to variation in the training and test data. As $k$ increases, bias increases but variance decreases. See Guerrilla Section 1, Problem 2 for a mathematical justification of this tradeoff. By using cross-validation, we can select a value of $k$ that we hope will balance the bias with the variance to result in low generalization error.

**Flexibility:** When $k > 1$, KNN can be modified to output predicted probabilities $P(Y|X)$ by defining $P(Y|X)$ as the proportion of nearest neighbors to $X$ in the training set that have class $Y$. KNN can also be adapted for regression - instead of taking the majority vote, take the average of the $y$ values for the nearest neighbors. KNN can learn very complicated, non-linear decision boundaries.

**Non-parametric:** KNN is a non-parametric method, which means that the number of parameters in the model grows with $n$, the number of training points. This is as opposed to parametric methods, for which the number of parameters is independent of $n$. Some parametric models are linear regression, LDA, and neural networks.

**Behavior in high dimensions:** KNN does not behave well in high dimensions. As dimension

---

\(^2\)ANN methods are beyond the scope of this course, but are useful in real applications.
increases, data points drift farther apart, so even the nearest neighbor to a point will tend to be very far away. More on this later in the note.

Theoretical properties: 1-NN has impressive theoretical guarantees for such a simple method. Cover and Hart, 1967 prove that as the number of training samples $n$ approaches infinity, the expected prediction error for 1-NN is upper bounded by $2\varepsilon^*$, where $\varepsilon^*$ is the Bayes (optimal) error. Fix and Hodges, 1951 prove that as $n$ and $k$ approach infinity and if $\frac{k}{n} \to 0$, then the $k$ nearest neighbor error approaches the Bayes error.

2 High-Dimensional Space

To understand why KNN does not perform well in high-dimensional space, we first need to get an intuition for what happens in high-dimensional space. Actually, in high-dimensional space, much of our low-dimensional intuition breaks down. Here is one classical example. Consider a ball in $\mathbb{R}^d$ centered at the origin with radius $r$, and suppose we have another ball of radius $r - \varepsilon$ centered at the origin. In low dimensions, we can visually see that much of the volume of the outer ball is also in the inner ball.

In general, the volume of the outer ball is proportional to $r^d$, while the volume of the inner ball is proportional to $(r - \varepsilon)^d$. Thus the ratio of the volume of the inner ball to that of the outer ball is

$$\frac{(r - \varepsilon)^d}{r^d} = \left(1 - \frac{\varepsilon}{r}\right)^d \approx e^{-\varepsilon d/r} \to 0$$

Hence as $d$ gets large, most of the volume of the outer ball is concentrated in the annular region \{ $x: r - \varepsilon < x < r$ \} instead of the inner ball.

High dimensions also make Gaussian distributions behave counter-intuitively. Suppose $X \sim \mathcal{N}(0, \sigma^2 I)$. If $X_i$ are the components of $X$ and $R$ is the distance from $X$ to the origin, then $R^2 = \sum_{i=1}^d X_i^2$. We have $E(R^2) = d\sigma^2$, so in expectation a random Gaussian will actually be reasonably far from the origin. If $\sigma = 1$, then $R^2$ is distributed chi-squared with $d$ degrees of freedom. One can show that with high probability, this multivariate Gaussian will lie within an annular region with squared radius $E(R^2) = d\sigma^2$ (one possible approach is to use a chi-squared Chernoff bound). This phenomenon is known as concentration of measure. Without resorting to more complicated inequalities, we can show a simple, weaker result:

**Theorem:** If $X_i \sim \mathcal{N}(0, \sigma^2)$, $i = 1, ..., d$ are independent and $R^2 = \sum_{i=1}^d X_i^2$, then for every $\varepsilon > 0$, the following holds:

$$\lim_{d \to \infty} P(|R^2 - E(R^2)| \geq d^{\frac{1}{2} + \varepsilon}) = 0$$

Thus in the limit, the squared radius is concentrated about its mean.

**Proof:** From the formula for the variance of a chi-squared distribution, we see that $Var(R^2) = 2d\sigma^4$. Applying a Chebyshev bound yields

$$P(|R^2 - E(R^2)| \geq d^{\frac{1}{2} + \varepsilon}) \leq \frac{2d\sigma^4}{d^{1+2\varepsilon}} \to 0$$

$\square$
Thus a random Gaussian will lie within a thin annular region away from the origin in high dimensions with high probability, even though the mode of the Gaussian bell curve is at the origin. This illustrates the phenomenon in high dimensions where random data is spread very far apart. The KNN classifier was conceived on the principle that nearby points should be of the same class - however, in high dimensions, even the nearest neighbors that we have to a random test point will tend to be far away, so this principle is no longer useful.

3 Improving KNN

There are two main ways to improve KNN and overcome the shortcomings we have discussed.

1. Get more training data.
2. Reduce the dimensionality of the features and/or pick better features. Consider other choices of distance function.

One example of reducing the dimensionality in image space is to lower the resolution of the image - while this is throwing some of the original pixel features away, we may still be able to get the same or better performance with a nearest neighbors method.

We can also modify the distance function. For example, we have a whole family of Minkowski distances that are induced by the $L^p$ norms:

$$D_p(x, z) = \left( \sum_{i=1}^{d} |x_i - z_i|^p \right)^{\frac{1}{p}}$$

Without preprocessing the data, 1-NN with the $L^3$ distance outperforms 1-NN with $L^2$ on MNIST.

We can also use kernels to compute distances in a different feature space. For example, if $k$ is a kernel with associated feature map $\Phi$ and we want to compute the Euclidean distance from $\Phi(x)$ to $\Phi(z)$, then we have

$$\|\Phi(x) - \Phi(z)\|_2^2 = \Phi(x)^T \Phi(x) - 2\Phi(x)^T \Phi(z) + \Phi(z)^T \Phi(z) = k(x, x) - 2k(x, z) + k(z, z)$$

Thus if we define $D(x, z) = \sqrt{k(x, x) - 2k(x, z) + k(z, z)}$, then we can perform Euclidean nearest neighbors in $\Phi$-space without explicitly representing $\Phi$ by using the kernelized distance function $D$. 