Machine Learning

9 – Softmax Regression, kNN Classification

Avisek Gupta
Postdoctoral Fellow, IAI, TCG CREST
avisek003@gmail.com

September 20, 2022

Recap: Logistic Regression Classifier

Decide class w_i where

$$P(w_i|x) > P(w_j|x)\ \forall j \neq i$$

Binary Classification:

Logistic Regression assumes the following parametric model to estimate the posterior probabilities of the two classes:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(w_0 + \sum_{t=1}^{d} w_t x_t)}$$

and,

$$P(y = 0|\mathbf{x}) = \frac{\exp(w_0 + \sum_{t=1}^{d} w_t x_t)}{1 + \exp(w_0 + \sum_{t=1}^{d} w_t x_t)}$$

The assumption of the above parametric model leads to a *linear* classifier, that classifies the data based on a hyperplane between the two classes.

Recap: Multi-class Classification

Given $c > 2$ number of classes, we consider building a single c-class discriminating classifier that is comprised of c linear functions of the form

$$y_j = \mathbf{w}_j^T \mathbf{x} + w_{j0} \, , \; j = 1, ..., c$$

Data **x** is assigned to class j if $y_j > y_k \, \forall k \neq j$.

The decision boundary between class j and class k is given by $y_j - y_k = 0$, and the equation of this (d-1) dimensional hyperplane is,

$$(\mathbf{w_j} - \mathbf{w_k})^T \mathbf{x} + (w_{j0} - w_{k0}) = 0$$

Each decision region is always a single connected and convex region.

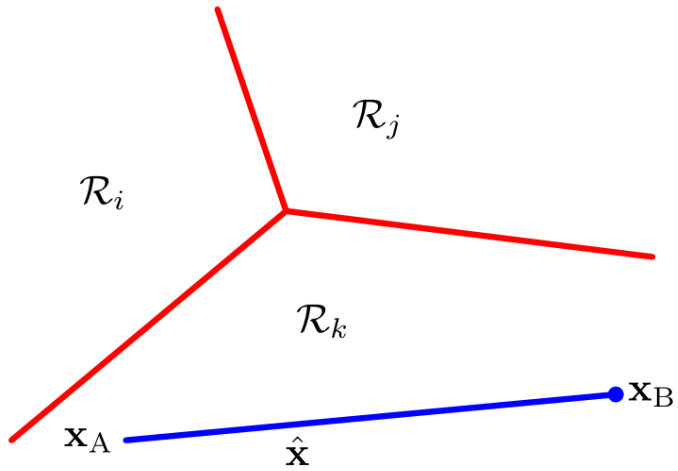


*Image Source:
Bishop - Pattern Recognition and Machine Learning

Multi-class Classification

Given $c > 2$ number of classes, we consider building a single c-class discriminating classifier that is comprised of c linear functions of the form,

$$y_j = \mathbf{w}_j^T \mathbf{x} + w_{j0} \, , \; j = 1, ..., c.$$

Data $\mathbf{x}$ is assigned to class j if $y_j > y_k \, \forall k \neq j$.

Multi-class Logistic Regression: Softmax Regression

We wish to estimate $P(y = j|\mathbf{x})$ in the following form,

$$P(y = j|\mathbf{x}) = \frac{\exp(a_j)}{\sum_{j'=1}^{c} \exp(a_{j'})}$$

$$= \frac{p(\mathbf{x}|y = j)P(y = j)}{\sum_{j'=1}^{c} p(\mathbf{x}|y = j')P(y = j')} \qquad \text{[By Bayes Rule]}$$

Hence we define $a_j = \ln\{p(\mathbf{x}|y = j)P(y = j)\}$.

Multi-class Logistic Regression: Softmax Regression

We assume $p(\mathbf{x}|y=j) \sim \mathcal{N}(\mu_j, \sigma^2 I)$:

$$p(\mathbf{x}|y=j) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left\{-\frac{||\mathbf{x}-\mu||^2}{2\sigma^2}\right\}$$

Then a_j, can be written as,

$$a_j = \ln\{p(\mathbf{x}|y=j)P(y=j)\} = -\frac{1}{2}\mathbf{x}^T\mathbf{x} + (\mathbf{w}_j^T\mathbf{x}) + b_j$$

where,

$$\mathbf{w}_j = \frac{\mu_j}{\sigma^2} \quad \text{, and,} \quad b_j = -\frac{1}{2}\mu_j^T\mu_j + \ln P(y=j) - \ln(2\pi\sigma^2)^{d/2}.$$

Multi-class Logistic Regression: Softmax Regression

Dropping $-\frac{1}{2}\mathbf{x}^T\mathbf{x}$ as a common term across all classes, we get

$$P(y=j|\mathbf{x}) = \frac{\exp(a_j)}{\sum_{j'=1}^{c}\exp(a_{j'})} \quad \text{, where,} \quad a_j = \mathbf{w}_j^T\mathbf{x} + b_j.$$

Therefore,

$$P(y=j|\mathbf{x}) = \frac{\exp(\mathbf{w}_j^T\mathbf{x} + b_j)}{\sum_{j'=1}^{c}\exp(\mathbf{w}_{j'}^T\mathbf{x} + b_{j'})},$$

where,

$$\mathbf{w}_j = \frac{\mu_j}{\sigma^2} \quad \text{, and,} \quad b_j = -\frac{1}{2}\mu_j^T\mu_j + \ln P(y=j) - \ln(2\pi\sigma^2)^{d/2}.$$

Softmax Regression

A **softmax** function is defined on the vector $\mathbf{a} = (a_1, ..., a_c)$:

$$\text{softmax}(\mathbf{a}) = \left\{ \frac{\exp(a_1)}{\sum_{j=1}^{c} \exp(a_j)}, ..., \frac{\exp(a_c)}{\sum_{j=1}^{c} \exp(a_j)} \right\}$$

The softmax function behaves similar to a max function:

$$\text{If } a_j >> a_k \ \forall k \neq j \ \text{, then, } \ P_j \to 1, P_k \to 0.$$

Softmax Regression

For multi-class classification, we have a single c-class discriminating classifier comprised of c linear functions:

$$P(y = j|\mathbf{x}) = \frac{\exp(\mathbf{w}_j^T \mathbf{x} + b_j)}{\sum_{j'=1}^{c} \exp(\mathbf{w}_{j'}^T \mathbf{x} + b_{j'})}, \; j = 1, ..., c.$$

Or, in terms of the softmax function,

$$\mathbf{y} = \text{softmax}(\mathbf{a}) \qquad \text{, where, } \mathbf{y} = \{P(y = 1|\mathbf{x}), ..., P(y = c|\mathbf{x})\}.$$

Softmax Regression

Then the system of c equations can be written as,

$$\mathbf{y} = \text{softmax}(\mathbf{a})$$
$$\mathbf{a} = W^T\mathbf{x} + \mathbf{b}$$

where the parameters we wish to estimate are:

$$W = \begin{bmatrix} w_{11} & \dots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{c1} & \dots & w_{cd} \end{bmatrix}, \text{ and, } \mathbf{b} = \begin{bmatrix} w_{10} \\ \vdots \\ w_{c0} \end{bmatrix}.$$

We get c values $\mathbf{y}^{(i)}$ for each data instance $\mathbf{x}^{(i)}$. For n data instances, we will get a total of $n \times c$ values $\mathbf{y}^{(1)}, ..., \mathbf{y}^{(n)}$.

Softmax Regression

We get c values $\mathbf{y}^{(i)}$ for each data instance $\mathbf{x}^{(i)}$. For n data instances, we will get a total of $n \times c$ values $\mathbf{y}^{(1)}, ..., \mathbf{y}^{(n)}$.

$$Y = \begin{bmatrix} y_1^{(1)} & \dots & y_c^{(1)} \\ \vdots & \ddots & \vdots \\ y_1^{(n)} & \dots & y_c^{(n)} \end{bmatrix}$$

Let the **ground-truth** target values we want the model to predict be given by,

$$T = \begin{bmatrix} t_1^{(1)} & \dots & t_c^{(1)} \\ \vdots & \ddots & \vdots \\ t_1^{(n)} & \dots & t_c^{(n)} \end{bmatrix}, \; t_j^{(i)} \in \{0, 1\}, \; \sum_{j=1}^{c} t_j^{(i)} = 1.$$

Softmax Regression

We define,

$$P(\mathbf{t}|\mathbf{w}, b) = \prod_{j=1}^{c} y_j^{t_j}$$

Ex.1.: If $c = 6$ and $\mathbf{t} = (0, 0, 0, 0, 1, 0)$, then $P(\mathbf{t}|\mathbf{w}, b) = \prod_{j=1}^{6} y_j^{t_j} = y_5$.

Ex.2.: If $c = 4$ and $\mathbf{t} = (0, 1, 0, 0)$, then $P(\mathbf{t}|\mathbf{w}, b) = \prod_{j=1}^{4} y_j^{t_j} = y_2$.

Softmax Regression

We define,

$$P(\mathbf{t}|\mathbf{w}, b) = \prod_{j=1}^{c} y_j^{t_j}$$

The negative log-likelihood (which we would like to minimize) can be defined as,

$$\ell(\mathbf{w}, b) = -\ln \prod_{i=1}^{n} \prod_{j=1}^{c} y_j^{(i) t_j^{(i)}} = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_j^{(i)} \ln y_j^{(i)}$$

This is called the **Cross-Entropy** (CE) loss: $CE = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_j^{(i)} \ln y_j^{(i)}$.

Recall: For binary classification, we defined Binary Cross-Entropy loss as,

$$BCE = -\sum_{i=1}^{n} t^{(i)} \ln y^{(i)} + (1 - t^{(i)}) \ln(1 - y^{(i)})$$

Softmax Regression

We define,

$$P(\mathbf{t}|\mathbf{w}, b) = \prod_{j=1}^{c} y_j^{t_j}$$

The negative log-likelihood (which we would like to minimize) can be defined as,

$$\ell(\mathbf{w}, b) = -\ln \prod_{i=1}^{n} \prod_{j=1}^{c} y_j^{(i) t_j^{(i)}} = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_j^{(i)} \ln y_j^{(i)}$$

What is the derivative of the CE loss wrt the model parameters?

This is called the **Cross-Entropy** (CE) loss: $CE = -\sum_{i=1}^{n} \sum_{j=1}^{c} t_j^{(i)} \ln y_j^{(i)}$.

Recall: For binary classification, we defined Binary Cross-Entropy loss as,

$$BCE = -\sum_{i=1}^{n} t^{(i)} \ln y^{(i)} + (1 - t^{(i)}) \ln(1 - y^{(i)})$$

Recap

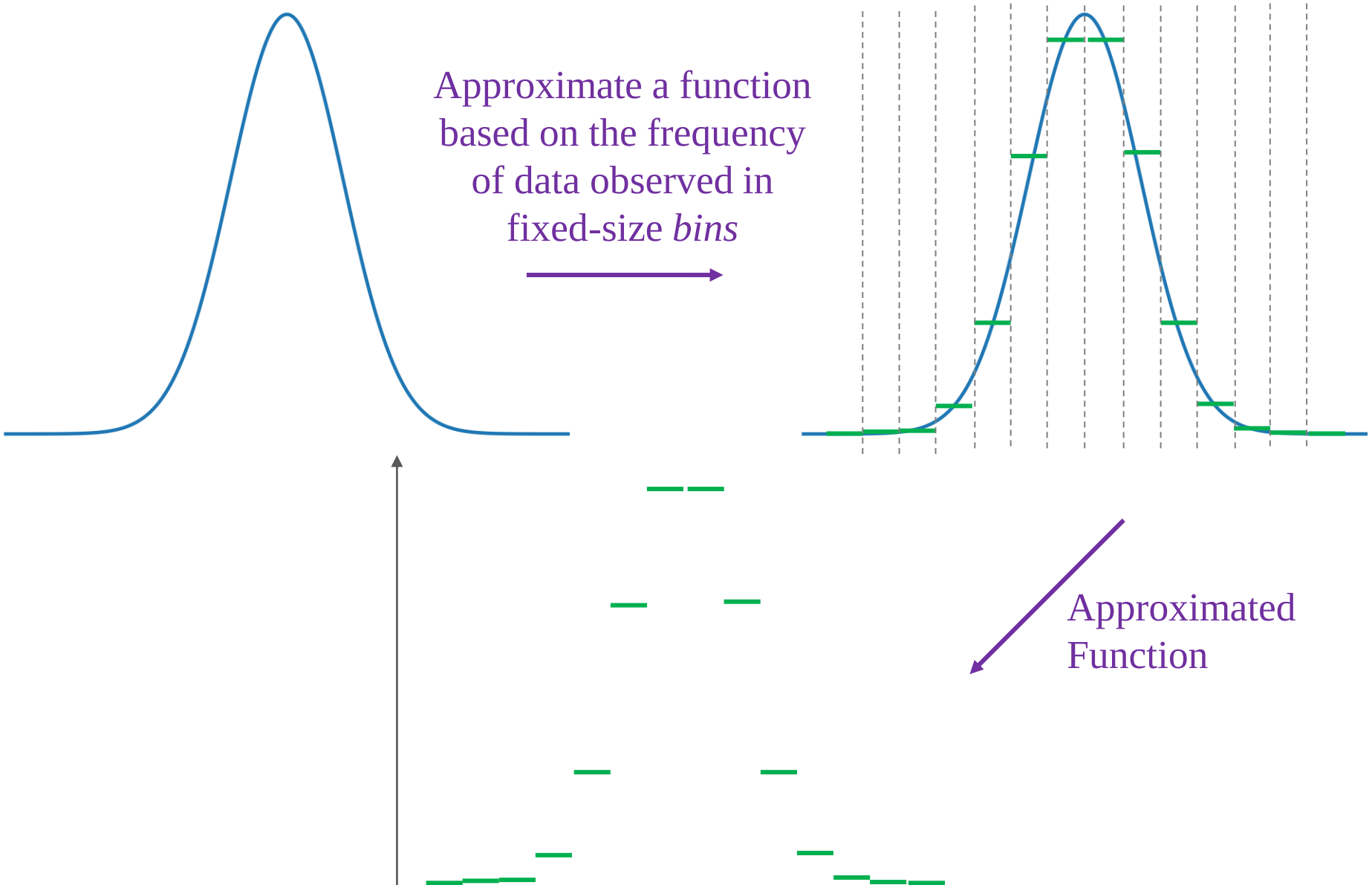
- Multi-Class Classification (Single system, one-vs-all, one-vs-one)
- Derivation of $P(y = j|\mathbf{x})$ for Softmax Regression
- Definition of the Softmax Function
- Derivation of Cross Entropy loss from the negative log-likelihood

k-NN Classification

k-NN Classification

- Parzen Window Density Estimation
- k-Nearest Neighbours Density Estimation
- k-NN Classification

Non-Parametric Estimation by Histograms



Similarly, histograms can be constructed for 2-dimensional functions, 3-dimensional functions, etc.

Non-Parametric Estimation by Histograms

Let there be a total of n samples, which fall into bins of width h.

Let the number of samples that fall in bin i be n_i.

Then, the *approximate probability density* of any point x that falls in bin i can be defined as:

$$\hat{p}_H(x) = \frac{n_i}{nh}$$

-> Can we find an approximate continuous function?

Parzen Window Density Estimation: Motivation

The probability that a vector $\mathbf{x}$ will lie in a region $\mathcal{R}$ is given by,

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x} \ .$$

If we draw n samples $\mathbf{x}_1, ..., \mathbf{x}_n$, then the probability that k of these samples will lie in $\mathcal{R}$ will follow a binomial distribution with parameters (n, P).

The expected value of the binomial distribution is $k = nP, \implies P = k/n$.

This is a decent estimate for P which follows a binomial distribution.

Parzen Window Density Estimation: Motivation

If we assume the region $\mathcal{R}$ is small enough so that $p(\mathbf{x})$ is uniform within the region, then using the volume V of the region we can write,

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x} \approx p(\mathbf{x}) V \ .$$

Using $P = k/n$, we obtain the following estimate:

$$\hat{p}(\mathbf{x}) \approx \frac{k/n}{V} \ .$$

This is still an averaged version of $p(\mathbf{x})$.

Parzen Window Density Estimation: Motivation

$$\hat{p}(\mathbf{x}) \approx \frac{k/n}{V} \ .$$

This is still an averaged version of $p(\mathbf{x})$. In order to closely approximate $p(\mathbf{x})$, we need to let $V \to 0$.

Challenges:

- If n is kept fixed as $V \to 0$, then $\hat{p}(\mathbf{x}) \to 0$.
- If a sample does fall in $\mathcal{R}$, then $\hat{p}(\mathbf{x}) \to \infty$.

Parzen Window Density Estimation: Motivation

To estimate $\hat{p}(\mathbf{x})$, a sequence of regions $\mathcal{R}_1, ..., \mathcal{R}_n$ are considered, where the first region is to be used with one sample, the second region is to be used with two samples, and so on.

Let V_n be the volume of $\mathcal{R}_n$, k_n be the number of samples that fall in $\mathcal{R}_n$. Then the n-th estimate for $p(x)$ is given by,

$$\hat{p}_n(\mathbf{x}) = \frac{k_n/n}{V_n}.$$

It can be proved that $\hat{p}_n(\mathbf{x})$ converges to $p(\mathbf{x})$ if the following three conditions hold:

- $\lim_{n\to\infty} V_n = 0$
- $\lim_{n\to\infty} k_n = \infty$
- $\lim_{n\to\infty} k_n/n = 0$

Parzen Window Density Estimation

We assume region $\mathcal{R}_n$ is a d-dimensional hypercube with length h_n. The volume of the hypercube is then $V_n = h_n^d$.

A window function is defined as,

$$\phi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq 1/2, \ j = 1, ..., d \\ 0 & o/w \end{cases}$$

Here $\phi(\mathbf{u})$ defines a unit hypercube centered at origin.

Therefore, $\phi((\mathbf{x} - \mathbf{x}_i)/h_n)$ will be equal to one if $\mathbf{x}$ falls within the hypercube of volume V_n centered at $\mathbf{x}_i$; and is zero otherwise. Hence the number of samples that fall in this hypercube is given by,

$$k_n = \sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right)$$

Parzen Window Density Estimation

The Parzen Window estimate is given by,

$$\hat{p}_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\phi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

In general, any choice of the above can be considered that follow these constraints:

- For h_n: $h_n > 0\ \forall n$, $\lim_{n\to\infty} h_n = 0$, $\lim_{n\to\infty} nh_n = \infty$.
  (Ex: $h_n = 1/\log n, h_n = 1/\sqrt{n}$), etc.

- For ϕ: Any density function, i.e., $\phi(u) \geq 0$, $\int_{-\infty}^{+\infty}\phi(u)du = 1$.

Along with the above two conditions, if $V_n = h_n^d$, then $\hat{p}_n(\mathbf{x})$ will be an asymptotically unbiased estimate of $p(x)$.

Parzen Window Density Estimation

In general, any choice following these constraints can be considered:

- For h_n: $h_n > 0 \; \forall n, \; \lim_{n\to\infty} h_n = 0, \; \lim_{n\to\infty} nh_n = \infty$.
  (Ex: $h_n = 1/\log n, h_n = 1/\sqrt{n}$), etc.

- For ϕ: Any density function, i.e., $\phi(u) \geq 0, \; \int_{-\infty}^{+\infty} \phi(u)du = 1$.

Along with the above two conditions, if $V_n = h_n^d$, then $\hat{p}_n(\mathbf{x})$ will be an asymptotically unbiased estimate of $p(x)$.

- Proved for $x \in \mathbb{R}$ by: Parzen, Emanuel. (1962). "On Estimation of a Probability Density Function and Mode". The Annals of Mathematical Statistics. 33 (3): 1065-1076.
- Proved for $x \in \mathbb{R}^d$ by: Cacoullos, Theophilos. (1964). "Estimation of a multivariate density". University of Minnesota.

Parzen Window Density Estimation

Two common approaches to obtain the sequence of regions satisfying the conditions:

1. Parzen Window approach: Shrink the initial volume by specifying V_n to be a function of n. Ex.: $V_n = 1/\sqrt{n}$
2. k-NN approach: Specify k_n as a function of n, so that V_n grows until it encloses the k_n neighbours of $\mathbf{x}$. Ex.: $k_n = \sqrt{n}$

Parzen Window Density Estimation

k-NN Density Estimation

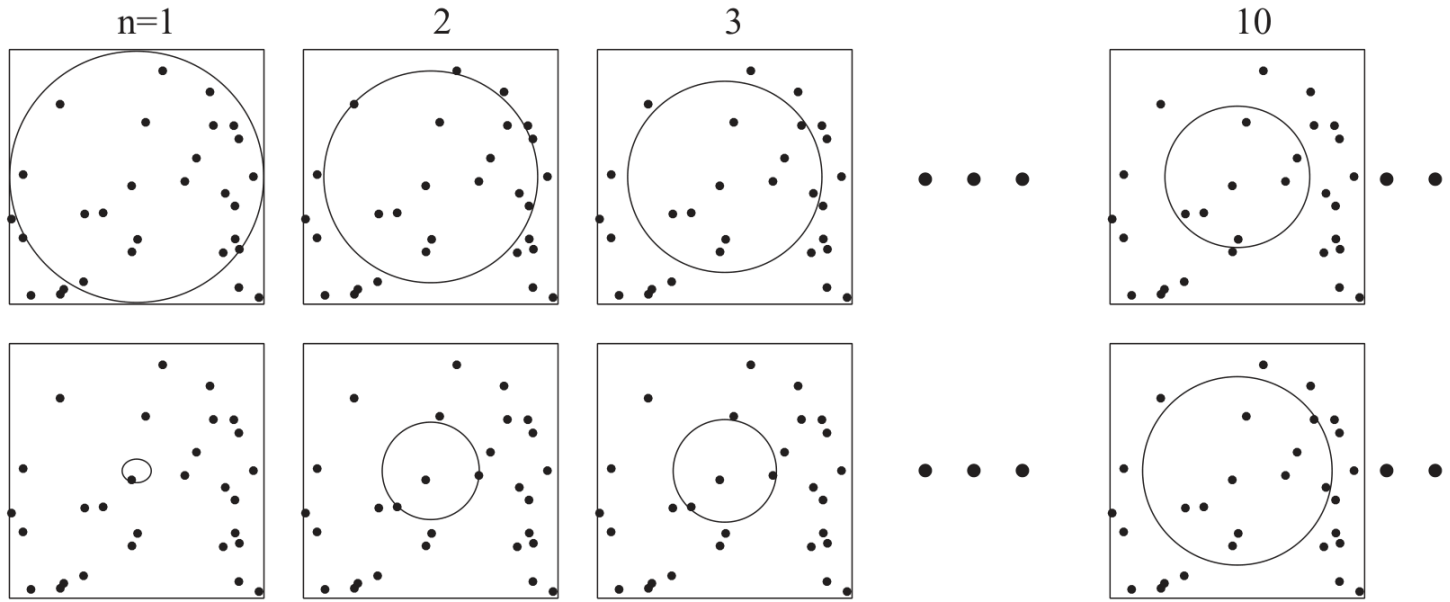


*Image Source:
Duda, Hart, Stork - Pattern Classification

k-NN Classification

- Parzen Window Density Estimation
- **k-Nearest Neighbours Density Estimation**
- k-NN Classification

k-Nearest Neighbours Density Estimation

Let $\mathbf{x}_1, ..., \mathbf{x}_n$ be n random iid vectors with $x_i \in \mathbb{R}^d$, from an uknown distribution $p(\mathbf{x})$ that we wish to estimate.

Let K_n be a positive integer.

The k-NN density estimation procedure to estimate $\hat{p}(\mathbf{x})$ for any $\mathbf{x}$ is described as:

1. Find r, the distance between $\mathbf{x}$ and its K_n-th nearest neighbour. Let $V_n(r, d)$ denote the volume of a disk of radius r in d dimensions.
2. The estimated density at $\mathbf{x}$ is given by,

$$\hat{p}(\mathbf{x}) = \frac{K_n}{n\, V_n(r, d)}.$$

k-Nearest Neighbours Density Estimation

The estimated k-NN density at $\mathbf{x}$,

$$\hat{p}(\mathbf{x}) = \frac{K_n}{n\, V_n(r, d)}.$$

If $\lim_{n\to\infty} K_n \to \infty$ and $\lim_{n\to\infty} \frac{K_n}{n} \to 0$, then $\hat{p}_n(\mathbf{x})$ is an asymptotically unbiased and consistent estimate of $p(x)$.

Proved by Loftsgaarden, Don O., and Charles P. Quesenberry. (1965). "A nonparametric estimate of a multivariate density function." The Annals of Mathematical Statistics 36 (3): 1049-1051.

k-NN Classification

- Parzen Window Density Estimation
- k-Nearest Neighbours Density Estimation
- **k-NN Classification**

k-Nearest Neighbours Classification

Our general classification rule:

Decide class y_j if $P(y_j|\mathbf{x}) > P(y_t|\mathbf{x})\ \forall t \neq j$.

or, decide class y_j if $P(y_j)p(\mathbf{x}|y_j) > P(y_t)p(\mathbf{x}|y_t)\ \forall t \neq j$.

Let the Priors be estimated as $\hat{P}(y_j) = n_j/n$.

To estimate $p(\mathbf{x}|y_j)$, first the k-nearest neighbours of $\mathbf{x}$ are found. Let K_j be the number of neighbours of $\mathbf{x}$ that belong to class j, and hence $\sum_{j=1}^{c} K_j = k$.

Let r denote the distance between $\mathbf{x}$ and its k-nearest neighbour. Let V_r denote the volume of a d-dimensional disk with radius r. Then we can define,

$$\hat{p}(\mathbf{x}|y_j) = \frac{K_j}{n_j V_r}.$$

k-Nearest Neighbours Classification

Let the Priors be estimated as $\hat{P}(y_j) = n_j/n$.

To estimate $p(\mathbf{x}|y_j)$, first the k-nearest neighbours of $\mathbf{x}$ are found. Let K_j be the number of neighbours of $\mathbf{x}$ that belong to class j, and hence $\sum_{j=1}^{c} K_j = k$.

Let r denote the distance between $\mathbf{x}$ and its k-nearest neighbour. Let V_r denote the volume of a d-dimensional disk with radius r. Then we can define,

$$\hat{p}(\mathbf{x}|y_j) = \frac{K_j}{n_j V_r}.$$

We decide class y_k if $\hat{P}(y_j)\hat{p}(\mathbf{x}|y_j) > \hat{P}(y_t)\hat{p}(\mathbf{x}|y_t)\ \forall t \neq j$.

$$\implies \frac{n_j}{n}.\frac{K_j}{n_j V_r} > \frac{n_t}{n}.\frac{K_t}{n_t V_r}\ \forall t \neq j$$

$$\implies K_j > K_t\ \forall t \neq j$$

k-Nearest Neighbours Classification

We decide class y_k if $K_j > K_t \; \forall t \neq j$

- For $k = 1$, the classifier is called the Nearest Neighbour classifier.
- *Lazy Learner*: k-NN does not 'learn' anything, that is, it has no parameters to estimate. Given an instance $\mathbf{x}$ to classify, k-NN simply consults the training data to find the k nearest neighbours of $\mathbf{x}$, and decides on the majority class among the neighbours.
- K_j can vary with the choice of distance metric used to determine the neighbours.
- Larger k correspond to smoother decision boundaries.

k-Nearest Neighbours Classification

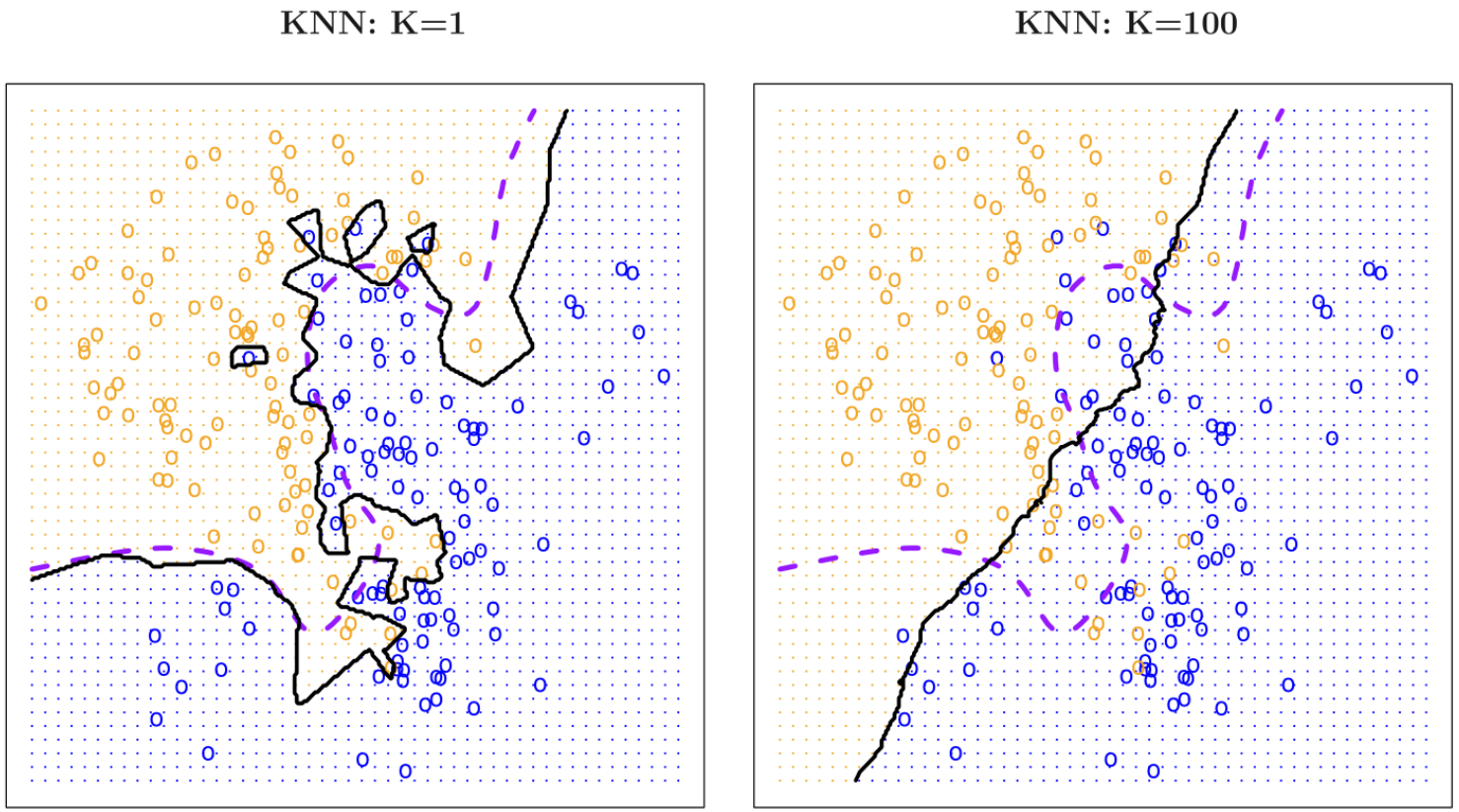


*Image Source:
James, Witten, Hastie, Tibshirani – An Introduction to Statistical Learning with R