5. k-nearest neighbors and cross validation

Shabana K M
PhD Research Scholar
Computer Science and Engineering

IIT Palakkad

4 September 2021

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Recap

- Polynomial regression
- Overfitting and underfitting
- Regularization
  - Ridge regression
  - LASSO
  - Elastic Net
- Logistic regression

Logistic regression: Problem definition

Given: Training data set comprising N observations $(x_n, y_n)_{n=1}^{N}$, where $x_n = [x_{n1}, x_{n2}, ..., x_{nD}]$ is the input and $y_n \in \{0, 1\}$ is the corresponding output

Goal: Predict the y value for a new value of x

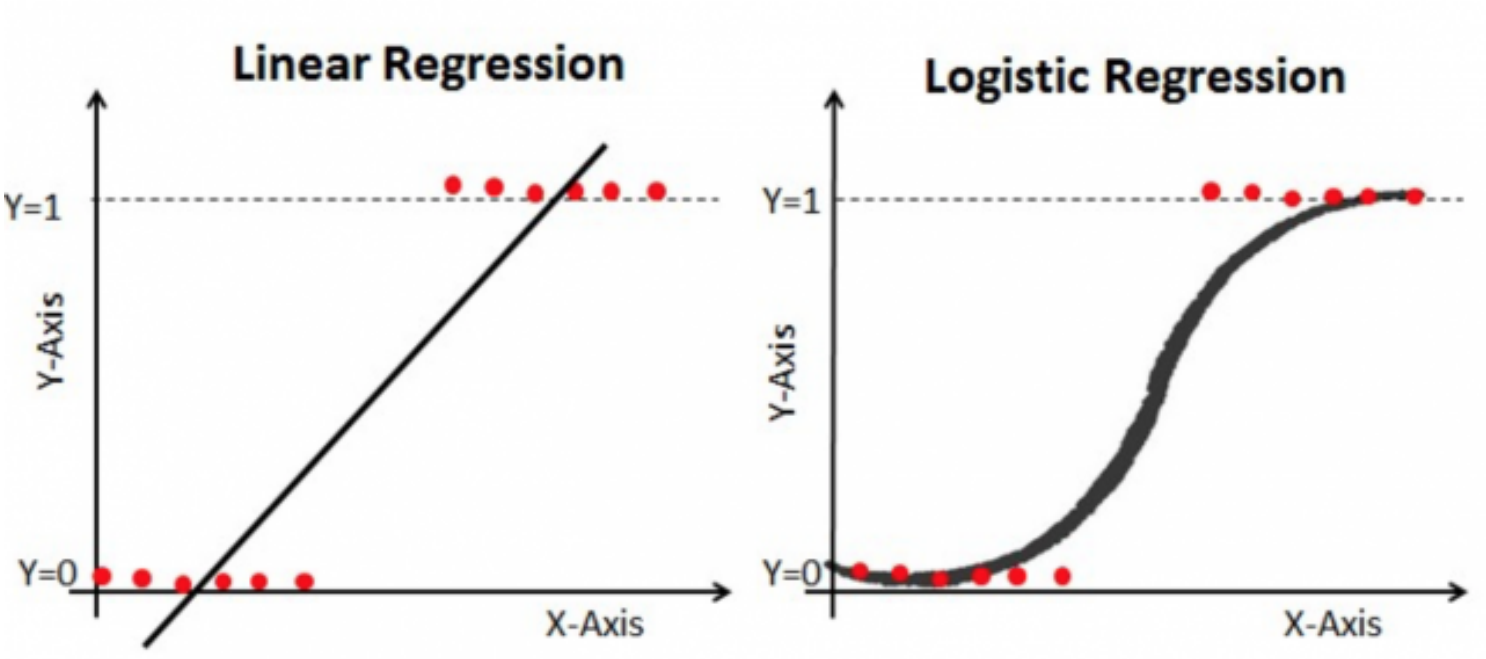
Estimate: The weights $w = [w_0, w_1, ..., w_D]$ such that:

Minimize: `cross-entropy`:

$$l(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(h(w, x_i)) + (1 - y_i) log(1 - h(w, x_i)))$$

$$\text{where } h(w, x_i) = \frac{1}{1+e^{-w^T x_i}}$$

Linear regression vs logistic regression



Gradient descent update

repeat until convergence $\{$

$$w_j = w_j - \alpha * \frac{1}{N} \sum_{i=1}^{N} (h(w, x) - y_i) * x_{ij} \qquad \text{for } j := 0 \ldots D \quad \}$$

Matrix notation

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1D} \\ 1 & x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

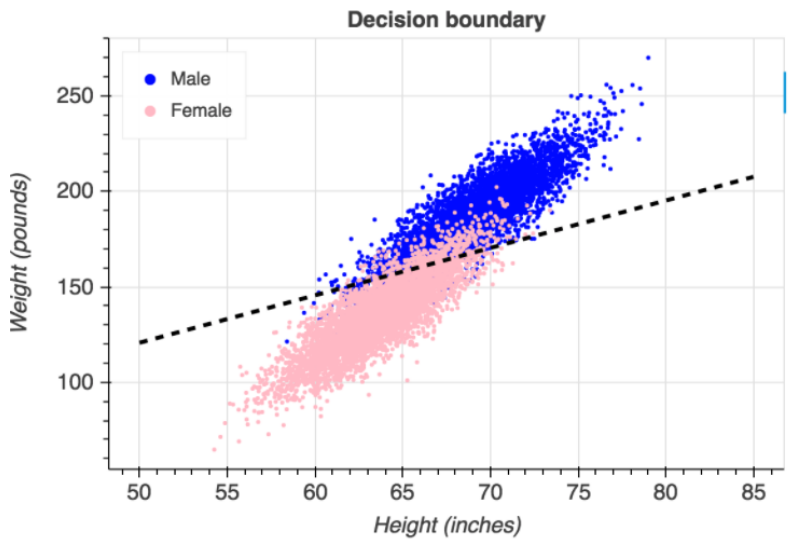
$$\boxed{w := w - \frac{\alpha}{N} X^T (g(Xw) - y)}$$

Making prediction

The **decision boundary** is the region that separates the area where $y = 0$ and $y = 1$

- determined by the model parameters
- is linear for logistic regression

- $w^T x \geq 0 \Rightarrow y = 1$
- $w^T x < 0 \Rightarrow y = 0$



Multi-class classification

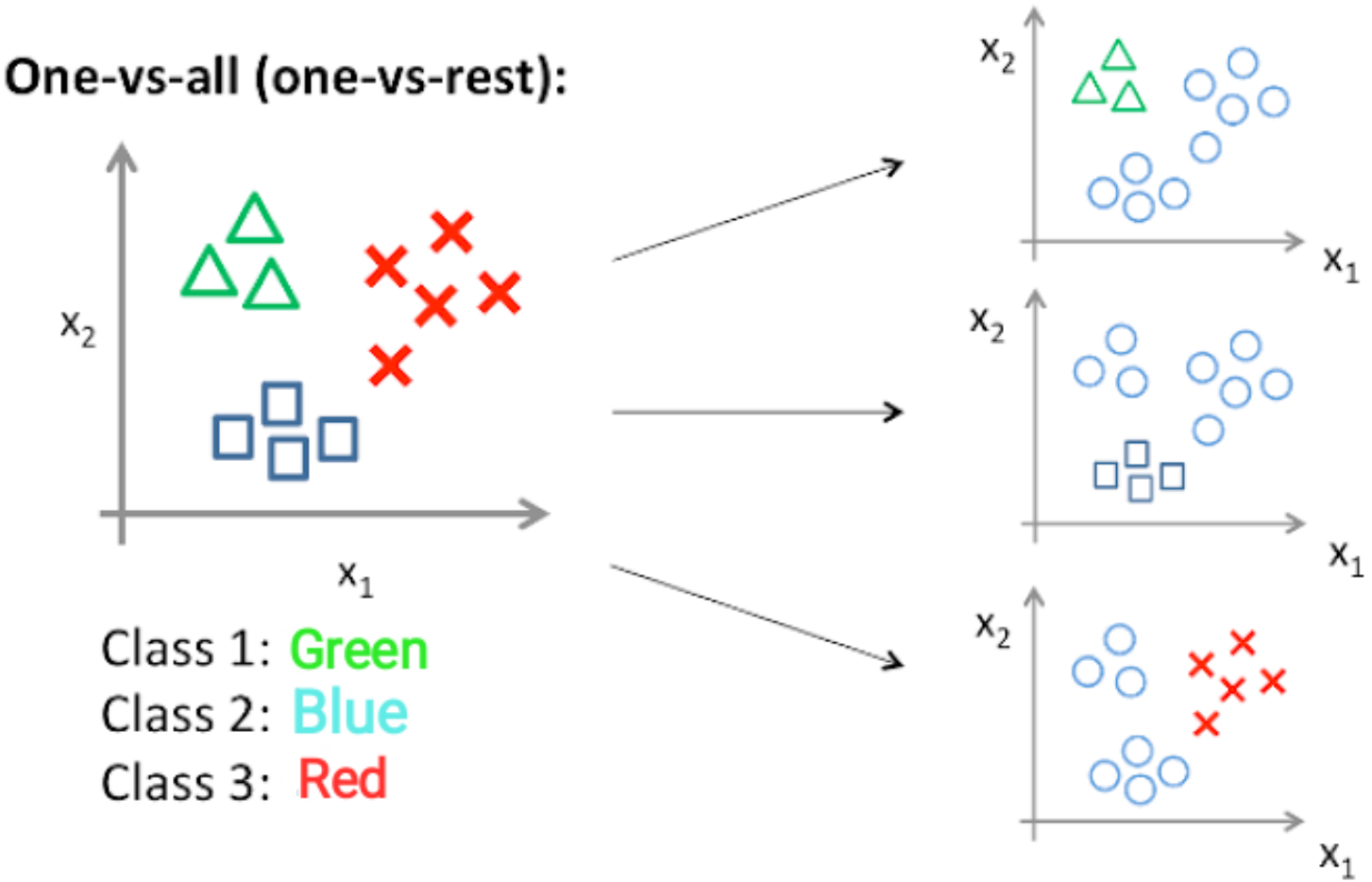
- logistic regression, by default, works for binary classification
- can be extended for multi-class classification (more than 2 classes)

One-vs-all approach

- Train a logistic regression classifier $h^{(i)}(w, x)$ for each class i to predict the probability that $y = i$
- On a new input x to make the prediction, pick the class i that maximizes $h^{(i)}(w, x)$

When training the classifier for class 1, training data points belonging to class 1 are treated as `positive` samples ($y = 1$) and all other classes as `negative` samples ($y = 0$)

Multi-class classification



Regularization for logistic regression

L2 regularization

$$l(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(h(w, x_i)) + (1 - y_i) log(1 - h(w, x_i))) + \lambda \sum_{j=1}^{D} w_j^2$$

`Gradient descent`

repeat until convergence $\{$

$$w_0 = w_0 - \alpha * \frac{1}{N}\sum_{i=1}^{N}(h(w, x) - y_i) * x_{i0}$$

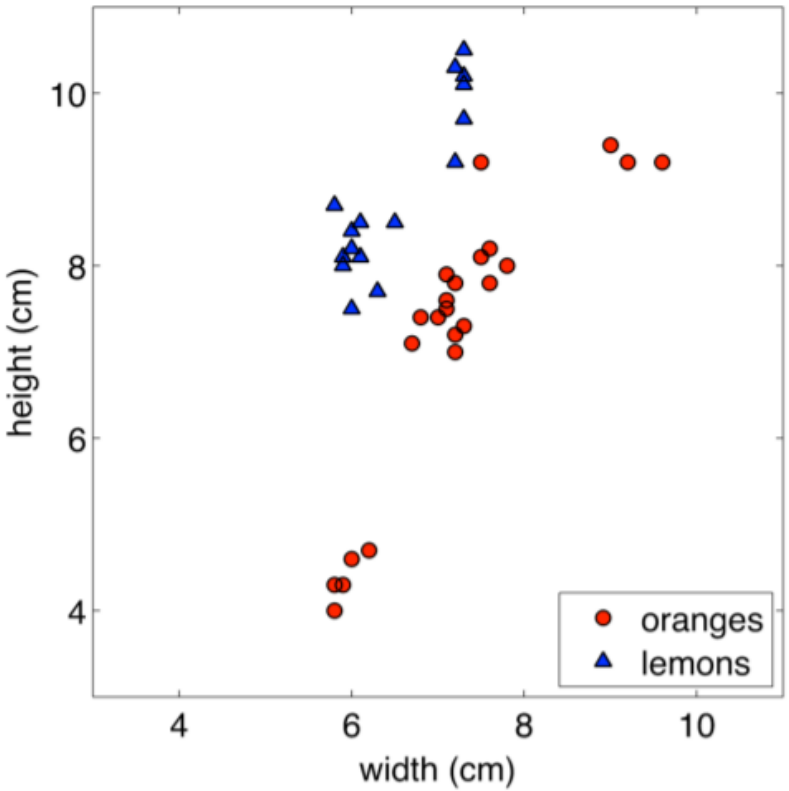
for $j \in \{1, .., D\}$

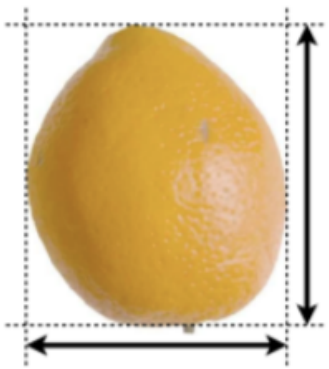
$$w_j = w_j - \left(\alpha * \frac{1}{N}\sum_{i=1}^{N}(h(w, x) - y_i) * x_{ij}\right) + \lambda w_j \quad \}$$

k-nearest neighbors algorithm

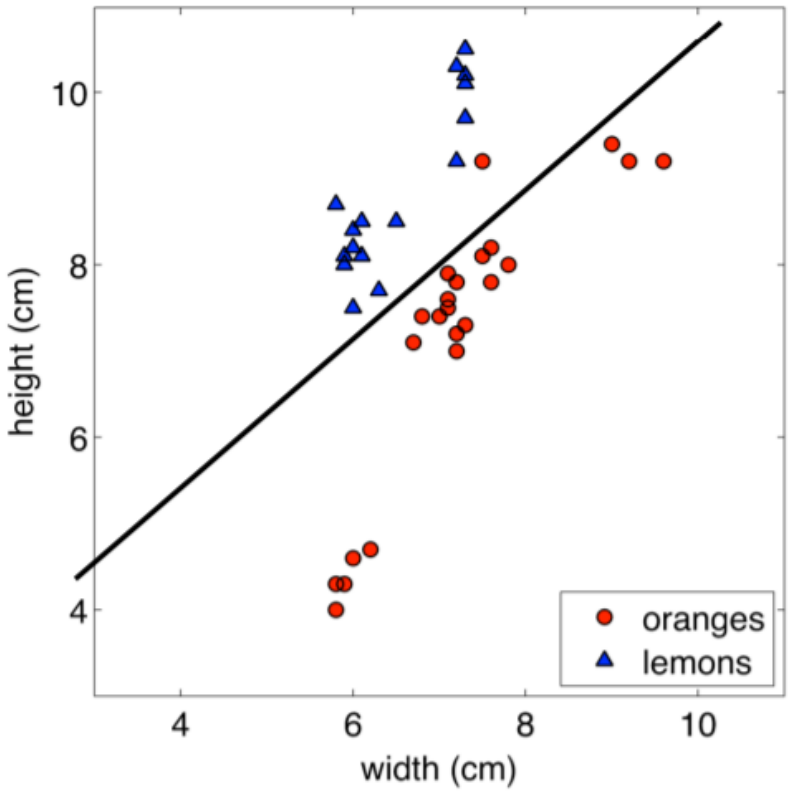
- supervised machine learning technique
- used for classification as well as regression
- `non-parametric`, `lazy` learning algorithm
  - non-parametric: does not make any assumptions on the underlying data distribution
  - lazy: does not use the training data points to do any generalization
- makes the assumption that `similar` points share similar labels
- instance based learning algorithm
  - no explicit training
  - learning amounts to simply storing training data

Classification: Example



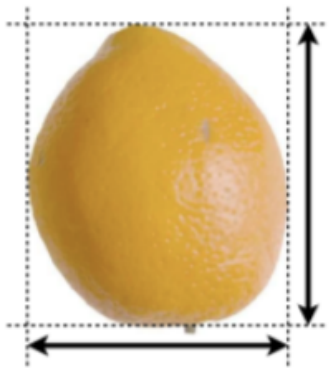


Classification: Example



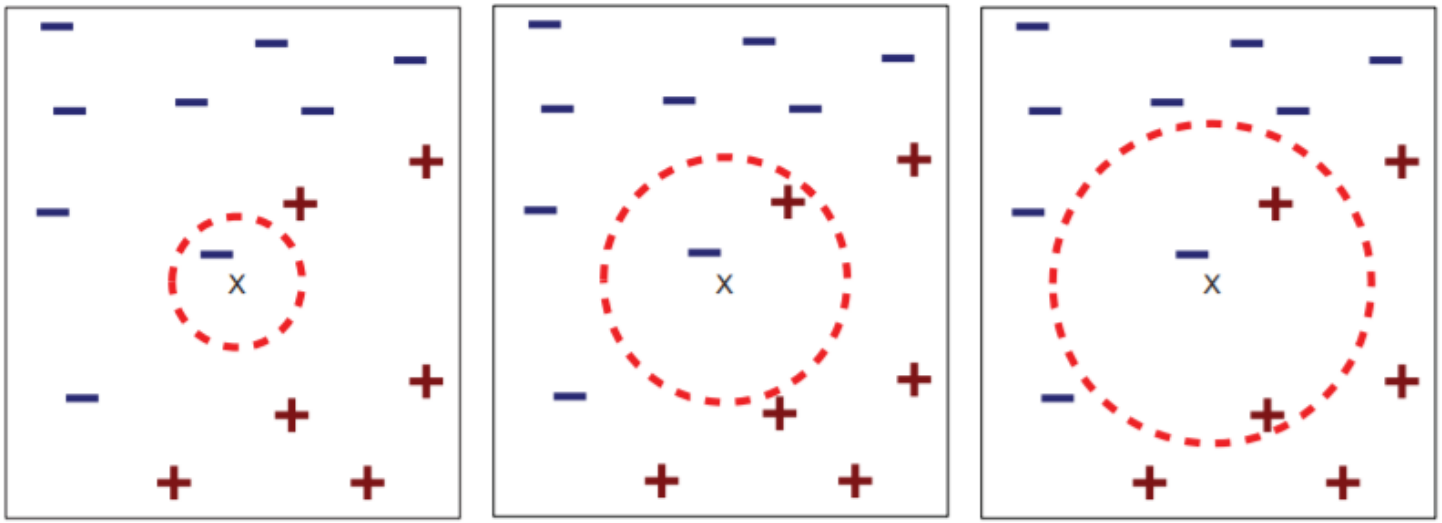
Can construct simple linear decision boundary:

$y = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$



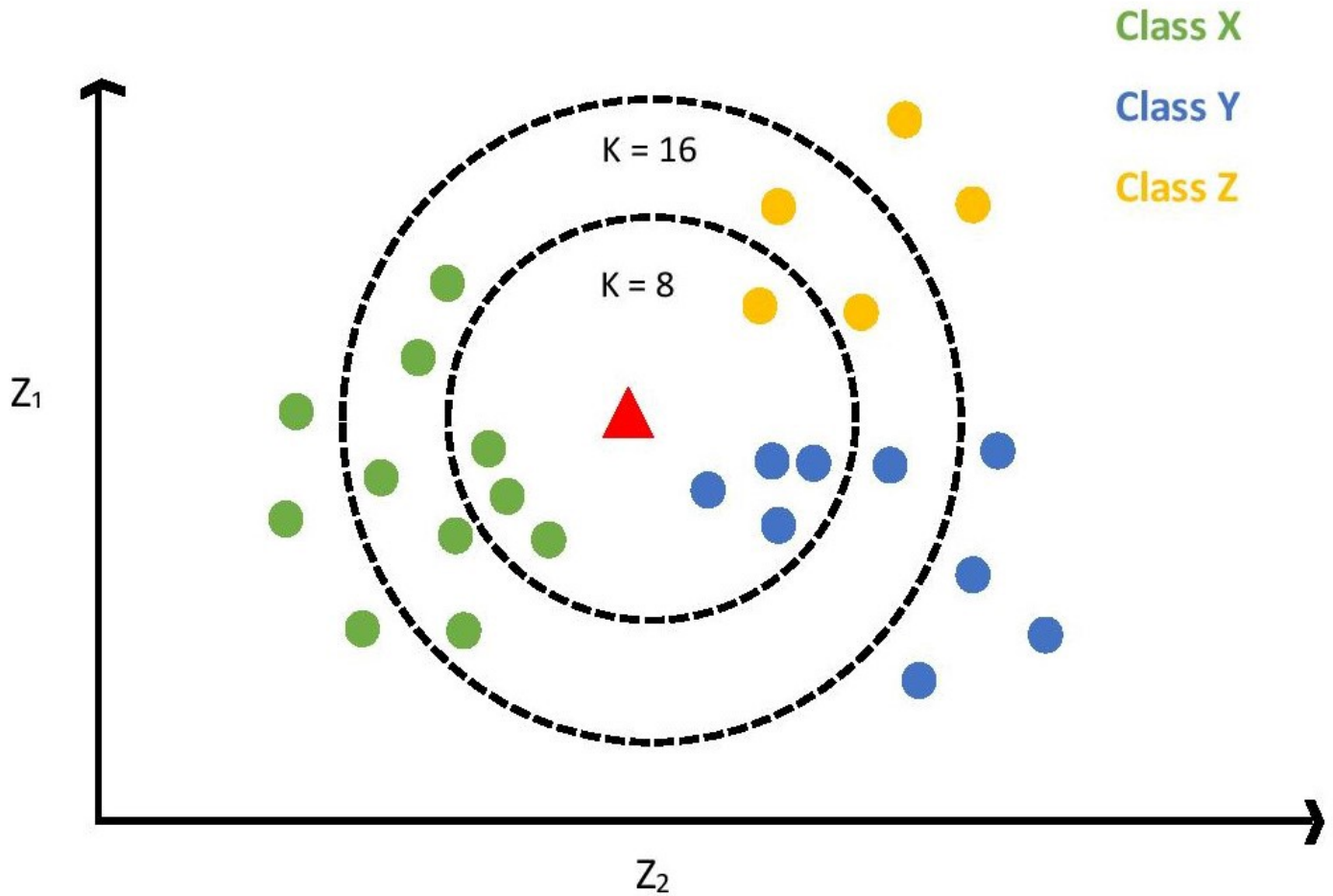
k-nearest neighbors classification

Idea: Classify using the majority vote of the `k` closest training points



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-nearest neighbors: Multi-class classification



k-nearest neighbors (kNN) classification

Given: Training data set comprising N observations $(x_n, y_n)_{n=1}^{N}$, where $x_n = [x_{n1}, x_{n2}, ..., x_{nD}]$ is the input and $y_n \in \{1, ..., L\}$ is the corresponding output

Goal: Predict the y value for a new value of x

kNN algorithm

1. Find k examples (x_i, y_i) `closest` to the test instance x
2. Classification output y is the majority class among all y_i

k-nearest neighbors algorithm

Each data point $x_i \in \mathcal{R}^d$

Distance computation: Commonly used distance measure is `Euclidean distance`

$$d_E(x_i, x_j) = \sqrt{\sum_{k=1}^{D}(x_{ik} - x_{jk})^2}$$

k-nearest neighbour regression

the target value for the test data point is predicted as the (weighted) average of the target values of the k neighbors

weighted knn

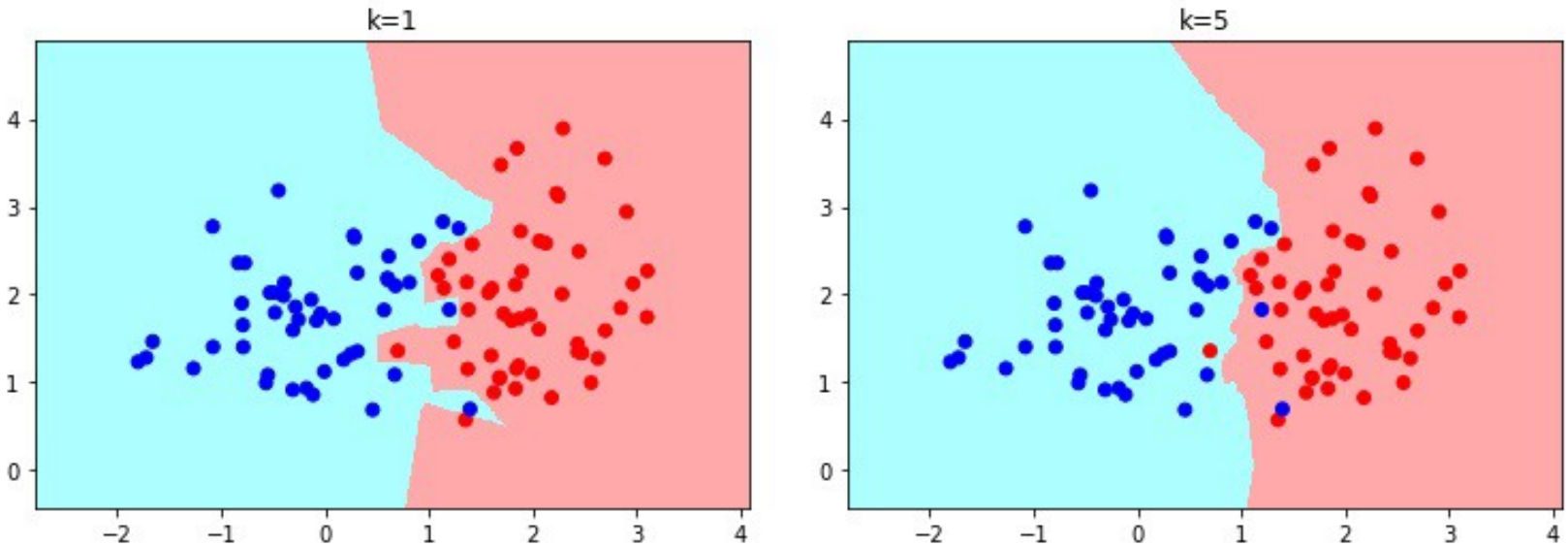
each point has a weight which is typically calculated based on its distance from the test data point

Choosing value of k

- best choice of k depends on data
- chosen as an odd number in case of binary classification
- small k values
  - models noise
  - causes overfitting
  - when k=1,(nearest neighbor classifier) training error is always zero
- large k values
  - accuracy "might" increase with increase in k but the computation cost also increases
  - end up looking at samples that are far away from query point
  - lead to underfitting
- a rule of thumb is $k < \sqrt{N}$, where N is the number of training examples

Decision boundary

- kNN doesn't explicitly compute decision boundaries
- implicitly learns complex non-linear decision boundary



kNN: Pros and Cons

Pros:

- simple and powerful - tuning complex parameters not needed
- works well with lots of data
- no training involved - new training examples can be added easily
- can learn complex target functions

Cons:

- expensive and slow
  - to determine the nearest neighbor of a new point x, distance to all N training examples must be computed
  - testing is expensive in terms of time and memory
- distances are less meaningful in high dimensions

Feature scaling

- technique to normalize the range of features in data
- attributes having larger magnitudes and ranges could gain dominance in distance computation
- could impact the performance of the learning algorithm

Normalization (Min Max scaling)

- values are shifted and rescaled so that they end up ranging between 0 and 1

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

Standardization

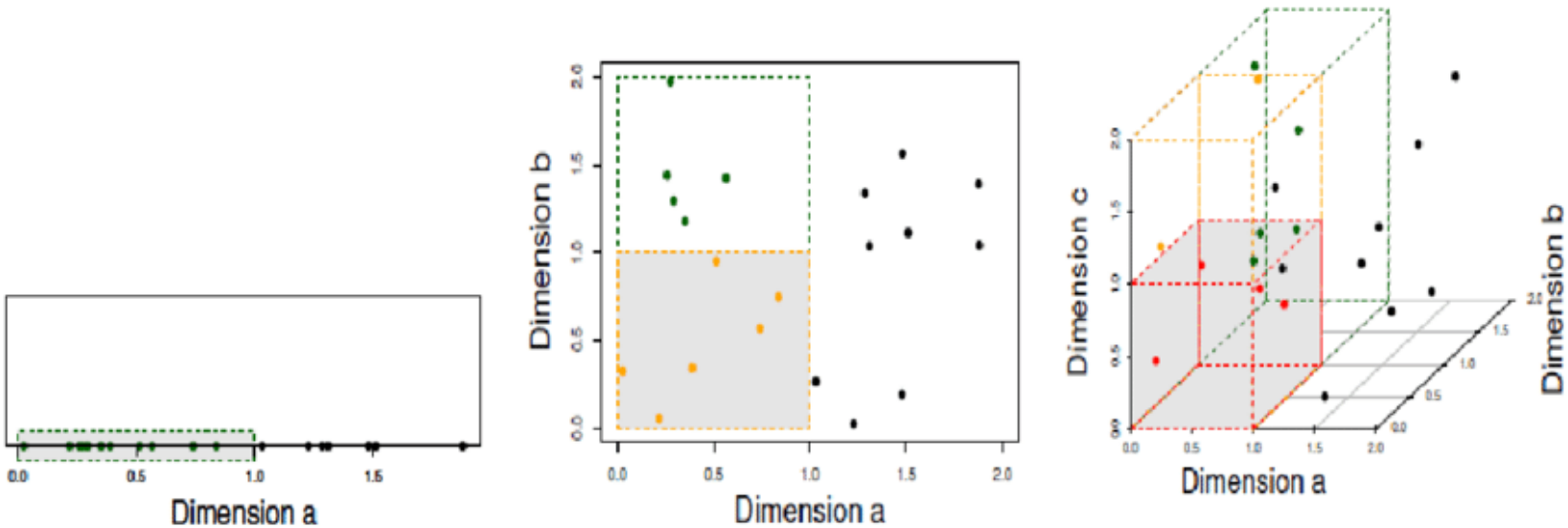
- values are centered around the mean with a unit standard deviation

$$x' = \frac{x - \mu(x)}{\sigma(x)}$$

- scaled attribute has zero mean and a unit standard deviation

Curse of dimensionality

- as the number of features increases, the dimension of the data point also increase
- in high dimensional spaces, points tend to never be close together
  - even the closest neighbors being too far away to give a good estimate
  - could lead to overfitting



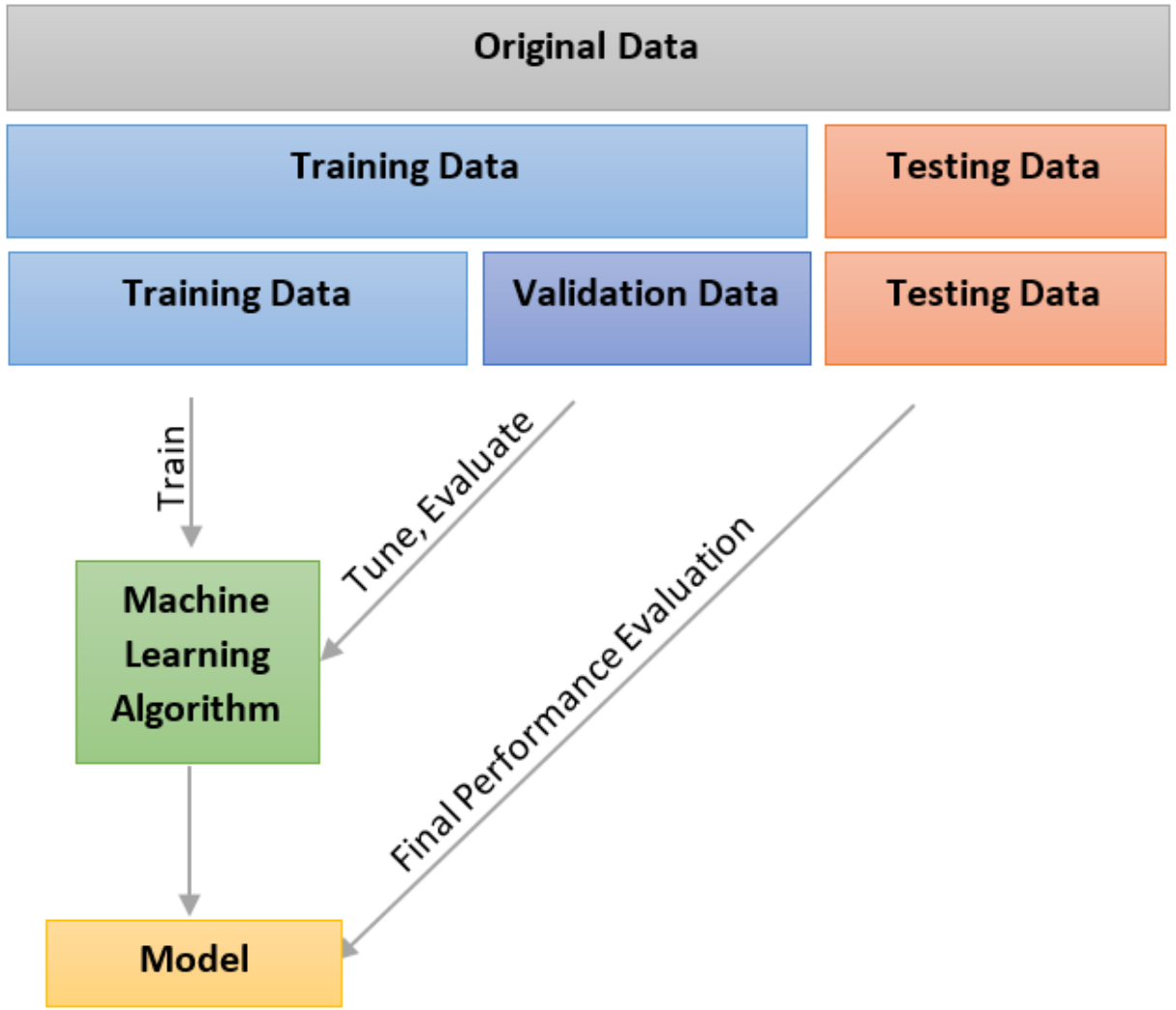
Curse of dimensionality

- kNN breaks down in high dimensional spaces because the "neighborhood" becomes very large
- need to perform dimension reduction by applying:
  - feature selection
  - feature extraction techniques such as prinipal component analysis (PCA)

Cross validation

- technique used to test the effectiveness of a machine learning model
- goal is to test the model's ability to predict new data that was not used in estimating it
  - avoids overfitting
- derives a more accurate estimate of model prediction performance and its ability to generalize to unseen data
- can be used to perform `hyperparameter` tuning for a model
  - hyperparameter: value used to control the learning process
  - user-defined
  - eg:- λ in ridge regression/lasso, k in knn, etc.
- different techniques - holdout method, k-fold cross validation, leave one out cross validation, etc.

Cross validation: The idea



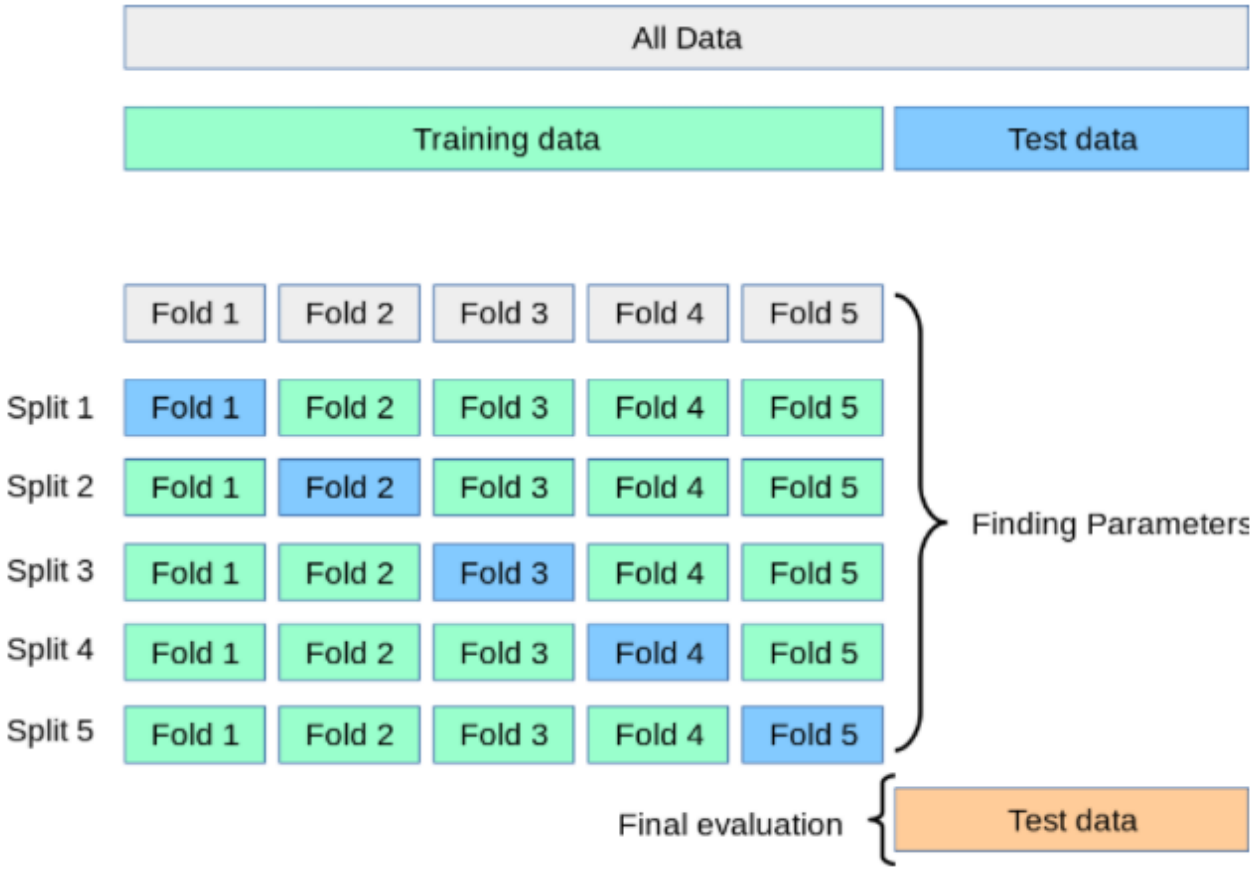
Holdout method

- another part of the dataset held out as a `validation set`
  - training performed on the training set
  - evaluation/hyperparameter tuning done on the validation set
  - final model evaluation performed on the test set
- drastically reduce the number of samples used for learning the model
- the results can depend on a particular random choice of (train, validation) sets

k-fold cross validation

- one of the popular validation techniques
- the training set is randomly split into k smaller sets (folds)
- for each of the k folds
  1. a model is trained using the $k-1$ folds as training data
  2. the resulting model is validated on the k^{th} fold (used as a test set to compute a performance measure such as accuracy)
- the performance measure reported by k-fold cross-validation is the average of the k values
- can be computationally expensive, but doesn't waste too much data
  - higher confidence on the cross-validation result with a larger k value
  - $k = 5$ or 10 is usually preferred

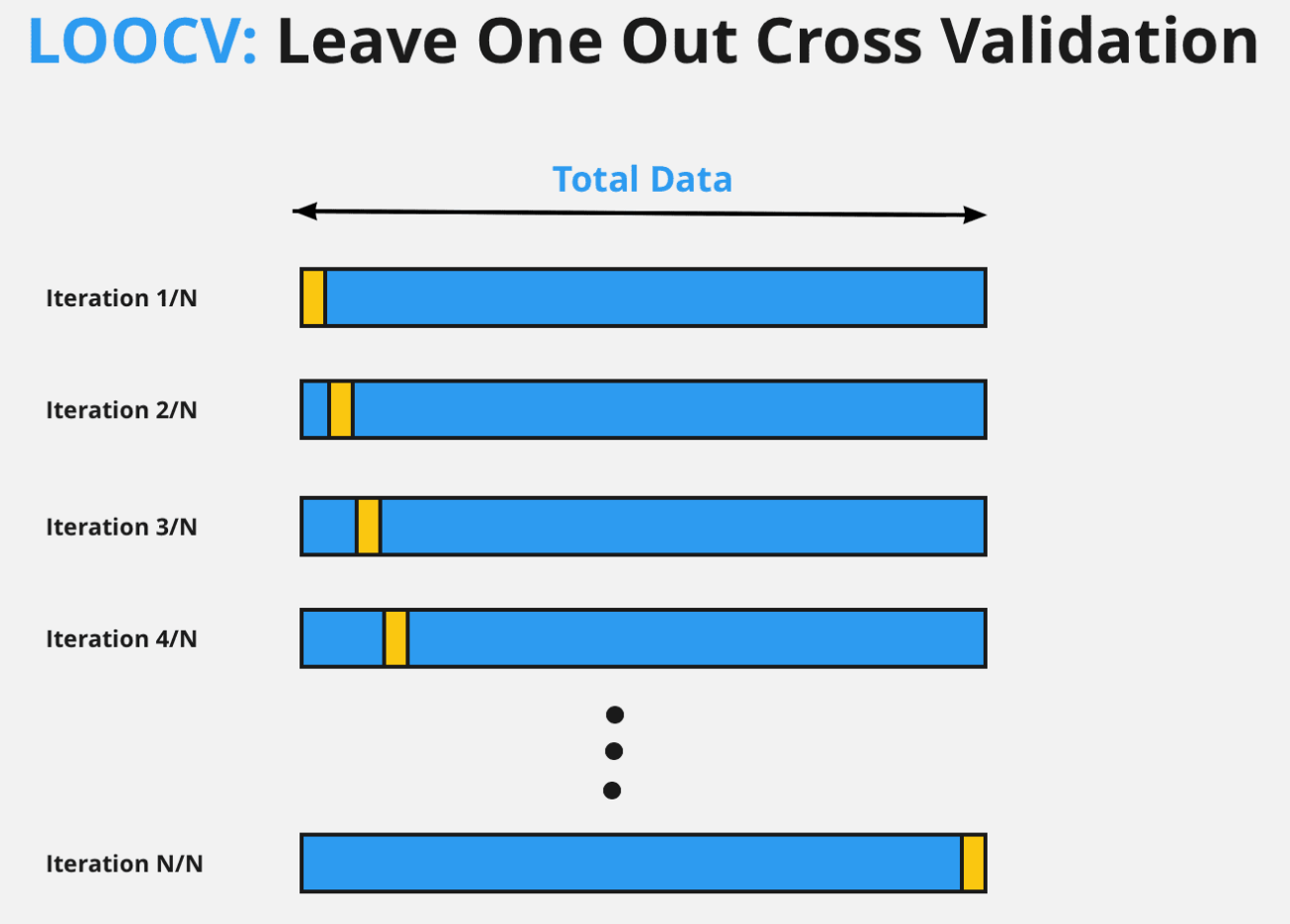
k-fold cross validation



Leave one out cross validation

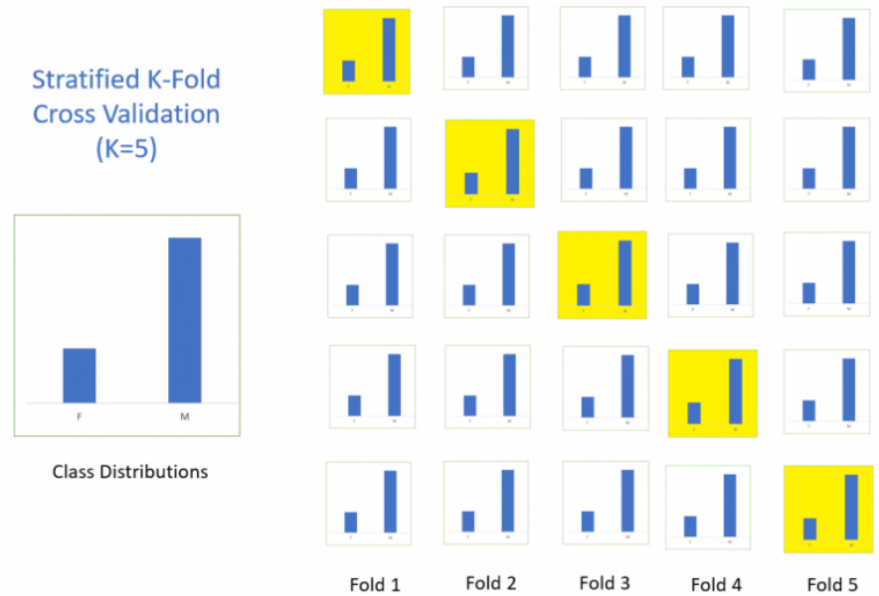
- each training set is created by taking all the samples except one
  - the test set is the left out sample
- for n samples we have n different training sets and test sets
- cross-validation procedure does not waste much data as only one sample is removed from the training set
- computationally expensive as n models have to be trained
- since the test set consists of a single data point, estimation gets highly influenced by the data point
  - results in high variance as an estimator for the test error

Leave one out cross validation



Stratified k fold cross validation

- a variation of *k*-fold which returns `stratified folds`: each set contains approximately the same percentage of samples of each target class as the complete set
  - the mean response value is approximately equal in all the folds



References

1. https://www.coursera.org/learn/machine-learning/resources/Zi29t
2. https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec06.pdf
3. https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/
4. https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/05_nn.pdf
5. https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/
6. https://scikit-learn.org/stable/modules/cross_validation.html
7. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

Thanks Google for the pictures!