Lecture 08
Machine Learning 2:
classification (part 1)

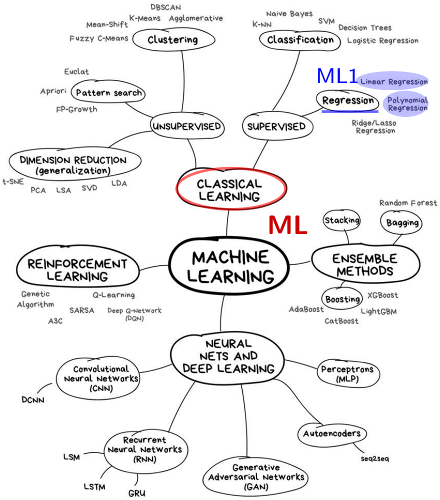
2024-10-09

Sébastien Valade

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Introduction

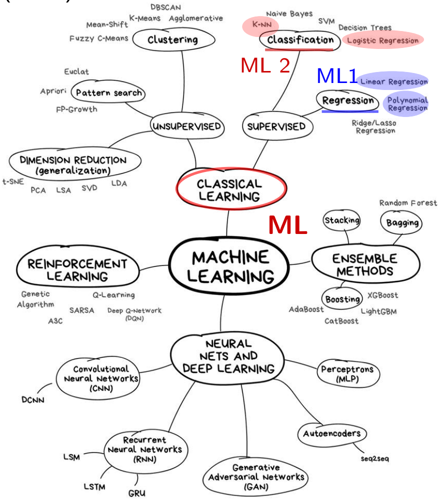
Last week: regression (ML1)



source

Introduction

Last week: regression (ML1)
This week: classification *part-1* (ML2)



source

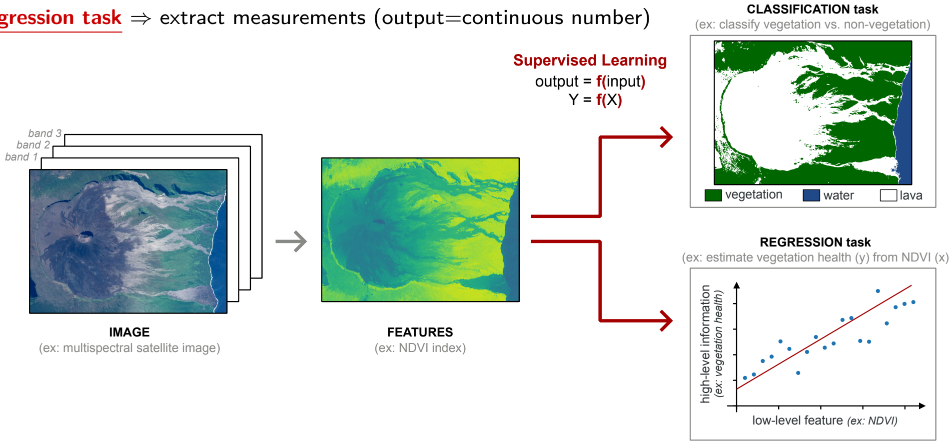
Introduction

Reminder:

⇒ the goal of **supervised learning** is to learn a function f which maps low-level image features (X) to high-level image information (Y), using training data (i.e. known pairs of (X_i, Y_i)):

→ **classification task** ⇒ extract semantic classes (output=discrete labels, categorical values)

→ **regression task** ⇒ extract measurements (output=continuous number)



*the term "feature" is here used in a broad sense, referring to any information extracted from the image

Introduction

What is classification?

⇒ the goal of **classification** is to assign **class labels** Y (i.e., discrete categorical values) to data points (e.g., pixels, images)

⇒ extracting **features** from the data is useful to find a space where samples from different classes are well separable (feature crafting can be either manual, or learned from the data using *unsupervised learning*, e.g. PCA)

⇒ the classification algorithm will have to learn the **decision boundary** in an N-dimensional **feature space**

What is classification?

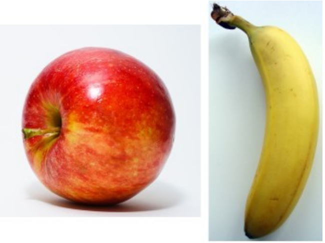
⇒ the goal of **classification** is to assign **class labels** Y (i.e., discrete categorical values) to data points (e.g., pixels, images)

⇒ extracting **features** from the data is useful to find a space where samples from different classes are well separable (feature crafting can be either manual, or learned from the data using *unsupervised learning*, e.g. PCA)

⇒ the classification algorithm will have to learn the **decision boundary** in an N-dimensional **feature space**

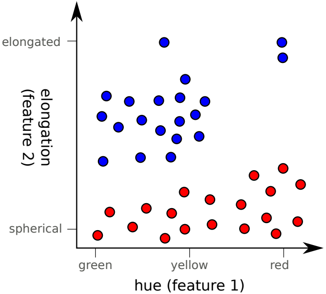
Toy example: *classify fruit images into either bananas or apples*

input image



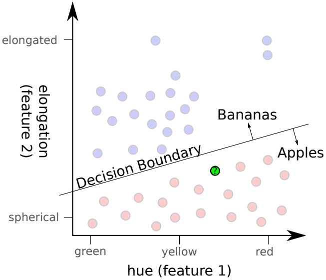
→

feature space



→

decision boundary



How is the decision boundary learned?

⇒ the **decision boundary** is the surface that separates the feature space into different regions corresponding to different classes

⇒ many algorithms exist to learn this boundary:

- *probabilistic approaches*:
  - **Logistic Regression** ⇒ *estimates the probability of a class using a logistic function, fitting a linear decision boundary (binary classification)*
  - **Softmax Regression** ⇒ *a multi-class extension of logistic regression that assigns probabilities to each class and fits linear boundaries between them*
  - **Naive Bayes** ⇒ *based on Bayes' theorem, uses probabilistic reasoning to calculate the likelihood of class membership*
- *deterministic approaches*:
  - **Perceptron** ⇒ *a linear classifier that finds a hyperplane to separate classes, adjusting weights based on misclassified points. Similar to logistic regression but non-probabilistic*
  - **k-Nearest Neighbors (kNN)** ⇒ *non-parametric method that classifies based on the majority class of the nearest neighbors, leading to non-linear boundaries*
  - **Support Vector Machines (SVM)** ⇒ *next lectures*
  - **Random Forest** ⇒ *next lectures*
  - **Convolutional Neural Networks (CNNs)** ⇒ *next lectures*

Introduction

How is the decision boundary learned?

⇒ the **decision boundary** is the surface that separates the feature space into different regions corresponding to different classes

⇒ many algorithms exist to learn this boundary:

- *probabilistic approaches*:
  - **Logistic Regression** ⇒ *estimates the probability of a class using a logistic function, fitting a linear decision boundary (binary classification)*
  - **Softmax Regression** ⇒ *a multi-class extension of logistic regression that assigns probabilities to each class and fits linear boundaries between them*
  - **Naive Bayes** ⇒ *based on Bayes' theorem, uses probabilistic reasoning to calculate the likelihood of class membership*
- *deterministic approaches*:
  - **Perceptron** ⇒ *a linear classifier that finds a hyperplane to separate classes, adjusting weights based on misclassified points. Similar to logistic regression but non-probabilistic*
  - **k-Nearest Neighbors (kNN)** ⇒ *non-parametric method that classifies based on the majority class of the nearest neighbors, leading to non-linear boundaries*
  - **Support Vector Machines (SVM)** ⇒ *next lectures*
  - **Random Forest** ⇒ *next lectures*
  - **Convolutional Neural Networks (CNNs)** ⇒ *next lectures*

Introduction

How is the decision boundary learned?

⇒ the **decision boundary** is the surface that separates the feature space into different regions corresponding to different classes

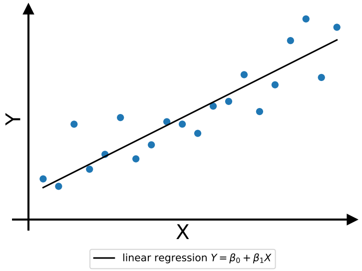
⇒ many algorithms exist to learn this boundary:

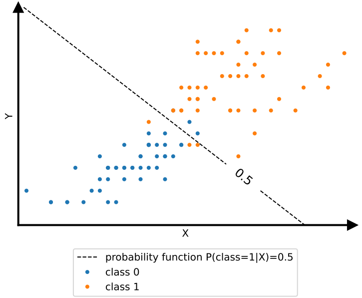
- *probabilistic approaches*:
  - **Logistic Regression** ⇒ *estimates the probability of a class using a logistic function, fitting a linear decision boundary (binary classification)*
  - **Softmax Regression** ⇒ *a multi-class extension of logistic regression that assigns probabilities to each class and fits linear boundaries between them*
  - **Naive Bayes** ⇒ *based on Bayes' theorem, uses probabilistic reasoning to calculate the likelihood of class membership*
- *deterministic approaches*:
  - **Perceptron** ⇒ *a linear classifier that finds a hyperplane to separate classes, adjusting weights based on misclassified points. Similar to logistic regression but non-probabilistic*
  - **k-Nearest Neighbors (kNN)** ⇒ *non-parametric method that classifies based on the majority class of the nearest neighbors, leading to non-linear boundaries*
  - **Support Vector Machines (SVM)** ⇒ *next lectures*
  - **Random Forest** ⇒ *next lectures*
  - **Convolutional Neural Networks (CNNs)** ⇒ *next lectures*

Logistic Regression

- Linear Regression (recap)
  - ⇒ used to predict continuous values of Y given X
  - ⇒ models the relationship between X and Y as a *linear equation*: $Y = \beta_0 + \beta_1 X$
  - ⇒ best model parameters (β_0, β_1) are found by *minimizing Mean Squared Error (MSE)*

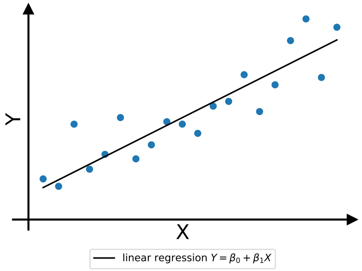
- Logistic Regression
  - ⇒ used to predict binary class values (discrete categorical values $y \in 0, 1$) of a data point, given features (X, Y)
  - ⇒ models a probability function using the *logistic function*: $p(y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$
  - ⇒ best model parameters (β_0, β_1) are found by *maximizing Likelihood*



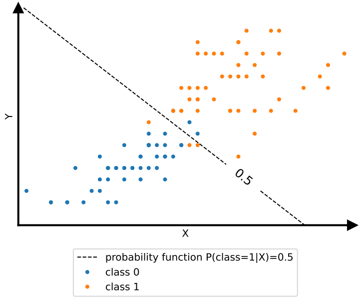


Logistic Regression

- Linear Regression (recap)
  - ⇒ used to predict continuous values of Y given X
  - ⇒ models the relationship between X and Y as a *linear equation*: $Y = \beta_0 + \beta_1 X$
  - ⇒ best model parameters (β_0, β_1) are found by *minimizing Mean Squared Error (MSE)*



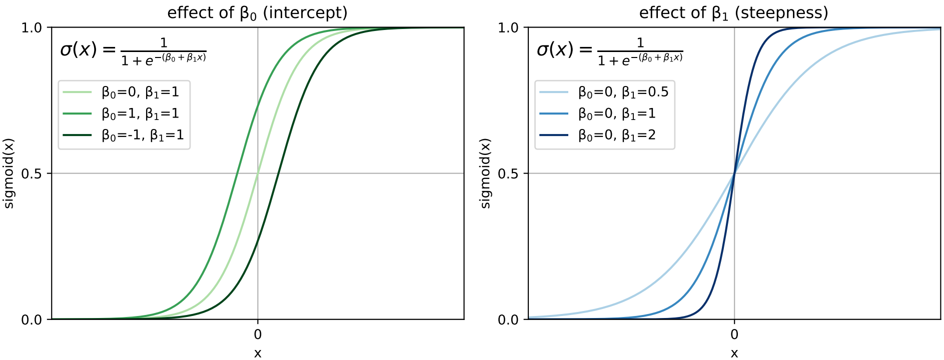
- Logistic Regression
  - ⇒ used to predict binary class values (discrete categorical values $y \in 0, 1$) of a data point, given features (X, Y)
  - ⇒ models a probability function using the *logistic function*: $p(y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$
  - ⇒ best model parameters (β_0, β_1) are found by *maximizing Likelihood*



Logistic Regression

⇒ in order to model the binary class probabilities, we use the **logistic function** (a.k.a. sigmoid function, S-shaped curve), which maps any real value of feature X to the range [0,1]:

$$\sigma(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$



Logistic Regression

⇒ in order to model the binary class probabilities, we use the **logistic function** (a.k.a. sigmoid function, S-shaped curve), which maps any real value of feature X to the range [0,1]:

$$\sigma(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-\theta^\top X}}$$

$$\text{with:} \quad \theta = [\beta_0, \beta_1] \quad \text{and} \quad X = [1, X]$$

⇒ the probability p is then calculated as:

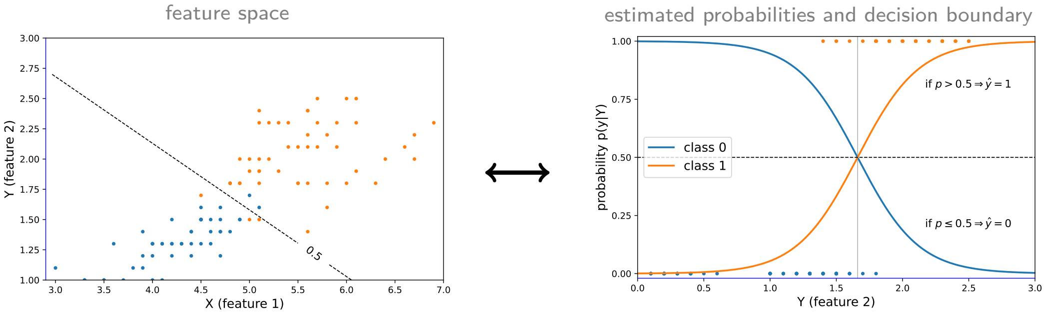
$$p(y = 1|X) = \sigma(X)$$
$$p(y = 0|X) = 1 - \sigma(X)$$

⇒ the prediction of the class $\hat{y} \in \{0, 1\}$ is made by comparing the probability $p(X)$ to a threshold (e.g. 0.5):

$$\hat{y} = \begin{cases} 1 & \text{if } p(X) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

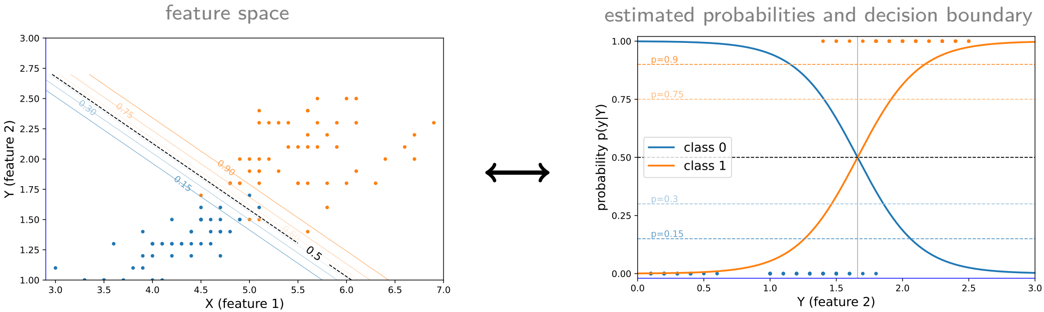
Logistic Regression

EX: estimate data point class $\hat{y} \in \{0|1\}$ from estimated probabilities $p(y|Y)$



Logistic Regression

EX: estimate data point class $\hat{y} = \{0|1\}$ from estimated probabilities $p(y|Y)$



Logistic Regression

⇒ the coefficients β_0 and β_1 are unknown, and must be estimated based on the available training data

⇒ the coefficients are estimated by using the **maximum likelihood function**

→ *find best estimates of β_0 and β_1, such that the predicted probability $\hat{p}(x_i)$ for each data point returns as closely as possible to the expected class*

→ *this can be formalized using a mathematical equation called a likelihood function:*

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

→ *the maximization of the likelihood function allows for the estimation of the best parameters β_0 and β_1:*

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\max_{\beta_0, \beta_1} L(\beta_0, \beta_1)$$

NB: we could also use least squares to fit the model, but the maximum likelihood is preferred because it has better statistical properties

Logistic Regression

⇒ the coefficients β_0 and β_1 are unknown, and must be estimated based on the available training data

⇒ the coefficients are estimated by using the **maximum likelihood function**

→ *find best estimates of β_0 and β_1, such that the predicted probability $\hat{p}(x_i)$ for each data point returns as closely as possible to the expected class*

→ *this can be formalized using a mathematical equation called a likelihood function:*

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

→ *the maximization of the likelihood function allows for the estimation of the best parameters β_0 and β_1:*

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\max_{\beta_0, \beta_1} L(\beta_0, \beta_1)$$

NB: we could also use least squares to fit the model, but the maximum likelihood is preferred because it has better statistical properties

Logistic Regression

⇒ the coefficients β_0 and β_1 are unknown, and must be estimated based on the available training data

⇒ the coefficients are estimated by using the **maximum likelihood function**

→ *find best estimates of β_0 and β_1, such that the predicted probability $\hat{p}(x_i)$ for each data point returns as closely as possible to the expected class*

→ *this can be formalized using a mathematical equation called a likelihood function:*

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

→ *the maximization of the likelihood function allows for the estimation of the best parameters β_0 and β_1:*

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\max_{\beta_0, \beta_1} L(\beta_0, \beta_1)$$

NB: we could also use least squares to fit the model, but the maximum likelihood is preferred because it has better statistical properties

Logistic Regression

⇒ the coefficients β_0 and β_1 are unknown, and must be estimated based on the available training data

⇒ the coefficients are estimated by using the **maximum likelihood function**

→ *find best estimates of β_0 and β_1, such that the predicted probability $\hat{p}(x_i)$ for each data point returns as closely as possible to the expected class*

→ *this can be formalized using a mathematical equation called a likelihood function:*

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

→ *the maximization of the likelihood function allows for the estimation of the best parameters β_0 and β_1:*

$$\hat{\beta}_0, \hat{\beta}_1 = \arg \max_{\beta_0, \beta_1} L(\beta_0, \beta_1)$$

NB: we could also use least squares to fit the model, but the maximum likelihood is preferred because it has better statistical properties

Logistic Regression

⇒ Logistic Regression can also be used to to predict a binary response but using multiple predictors
(in the previous slides, the binary class prediction was done using just the 1 feature Y)

⇒ if we consider k predictors, the model is then defined as:

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)}} = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k}}$$

NB: the classification results obtained using one predictor or several may be quite different, especially when there is correlation among the predictors!

⇒ nevertheless, **Logistic Regression** is limited as it can only model **binary classification**!
⇒ for **multi-class classification**, we need to use **Softmax Regression**

Logistic Regression

⇒ Logistic Regression can also be used to to predict a binary response but using multiple predictors
(in the previous slides, the binary class prediction was done using just the 1 feature Y)

⇒ if we consider k predictors, the model is then defined as:

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)}} = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k}}$$

NB: the classification results obtained using one predictor or several may be quite different, especially when there is correlation among the predictors!

⇒ nevertheless, **Logistic Regression** is limited as it can only model **binary classification**!
⇒ for **multi-class classification**, we need to use **Softmax Regression**

Softmax Regression

⇒ the **Softmax Regression** (a.k.a. *Multinomial Logistic Regression*) generalizes the logistic regression for multiple classes, by calculating probabilities $p(y)$ for each class $y \in \{1, ..., K\}$

$$p(y = k|X;\theta) = \frac{\exp(\theta^{(k)\top} X)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)}$$

where
- the numerator gives the exponentiated score for each class k
- the denominator normalizes the scores into a valid probability distribution

In other words:

$$\begin{bmatrix} p(y=1|X;\theta) \\ p(y=2|X;\theta) \\ \vdots \\ p(y=K|X;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)} \begin{bmatrix} exp(\theta^{(1)\top} X) \\ exp(\theta^{(2)\top} X) \\ \vdots \\ exp(\theta^{(K)\top} X) \end{bmatrix}$$

where $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$ are the model parameters

⇒ like with the *Logistic Regression classifier*, the *Softmax Regression classifier* predicts the class with the highest estimated probability:

$$\hat{y} = \arg\max_{k} p(y = k|X;\theta)$$

Softmax Regression

⇒ the **Softmax Regression** (a.k.a. *Multinomial Logistic Regression*) generalizes the logistic regression for multiple classes, by calculating probabilities $p(y)$ for each class $y \in \{1, ..., K\}$

$$p(y = k|X; \theta) = \frac{\exp(\theta^{(k)\top} X)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)}$$

where { the numerator gives the exponentiated score for each class k
the denominator normalizes the scores into a valid probability distribution

In other words:

$$\begin{bmatrix} p(y = 1|X; \theta) \\ p(y = 2|X; \theta) \\ \vdots \\ p(y = K|X; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)} \begin{bmatrix} exp(\theta^{(1)\top} X) \\ exp(\theta^{(2)\top} X) \\ \vdots \\ exp(\theta^{(K)\top} X) \end{bmatrix}$$

where $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$ are the model parameters

⇒ like with the *Logistic Regression classifier*, the *Softmax Regression classifier* predicts the class with the highest estimated probability:

$$\hat{y} = \arg\max_k p(y = k|X; \theta)$$

Softmax Regression

⇒ the **Softmax Regression** (a.k.a. *Multinomial Logistic Regression*) generalizes the logistic regression for multiple classes, by calculating probabilities $p(y)$ for each class $y \in \{1, ..., K\}$

$$p(y = k|X;\theta) = \frac{\exp(\theta^{(k)\top} X)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)}$$

where
- the numerator gives the exponentiated score for each class k
- the denominator normalizes the scores into a valid probability distribution

In other words:

$$\begin{bmatrix} p(y = 1|X;\theta) \\ p(y = 2|X;\theta) \\ \vdots \\ p(y = K|X;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} X)} \begin{bmatrix} exp(\theta^{(1)\top} X) \\ exp(\theta^{(2)\top} X) \\ \vdots \\ exp(\theta^{(K)\top} X) \end{bmatrix}$$

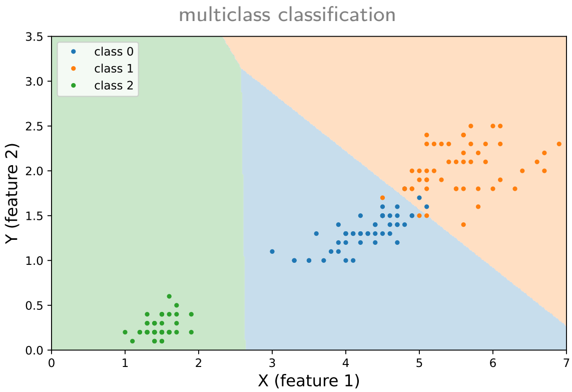
where $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$ are the model parameters

⇒ like with the *Logistic Regression classifier*, the *Softmax Regression classifier* predicts the class with the highest estimated probability:

$$\hat{y} = \arg\max_{k} p(y = k|X;\theta)$$

Softmax Regression

EX: estimate data point class $\hat{y} \in \{0, 1, 2\}$



$\Rightarrow$ **Softmax Regression** finds linear boundaries between multiple classes
$\rightarrow$ it is commonly used as the final layer in **neural networks** for classification tasks

kNN classification

⇒ Idea: **classify** a data point based on the majority class of its k **Nearest Neighbors**

⇒ Method:

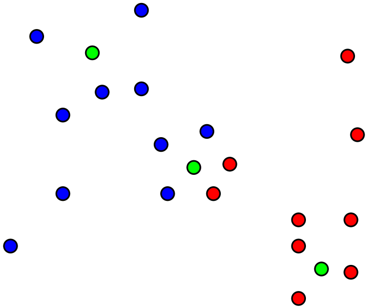
1. take random data points in the training dataset
2. for a sample find the k (e.g. 5) closest data points in the training dataset *(k is a hyperparameter)*
3. look at the neighbor labels, return/assign the mode
4. non-linear decision boundary can be recovered

kNN classification

⇒ Idea: **classify** a data point based on the majority class of its k **Nearest Neighbors**

⇒ Method:

1. take random data points in the training dataset
2. for a sample find the k (e.g. 5) closest data points in the training dataset *(k is a hyperparameter)*
3. look at the neighbor labels, return/assign the mode
4. non-linear decision boundary can be recovered

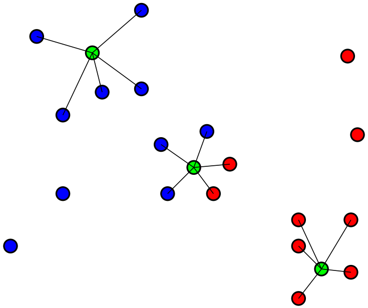


kNN classification

⇒ Idea: **classify** a data point based on the majority class of its k **Nearest Neighbors**

⇒ Method:

1. take random data points in the training dataset
2. for a sample find the k (e.g. 5) closest data points in the training dataset *(k is a hyperparameter)*
3. look at the neighbor labels, return/assign the mode
4. non-linear decision boundary can be recovered

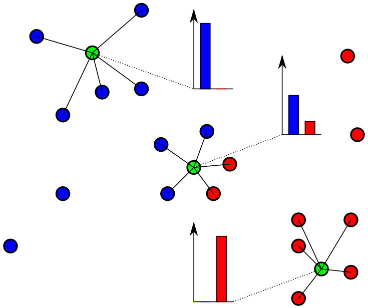


kNN classification

⇒ Idea: **classify** a data point based on the majority class of its k **Nearest Neighbors**

⇒ Method:

1. take random data points in the training dataset
2. for a sample find the k (e.g. 5) closest data points in the training dataset *(k is a hyperparameter)*
3. look at the neighbor labels, return/assign the mode
4. non-linear decision boundary can be recovered

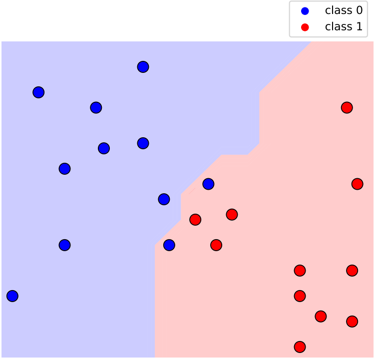


kNN classification

⇒ Idea: **classify** a data point based on the majority class of its k **Nearest Neighbors**

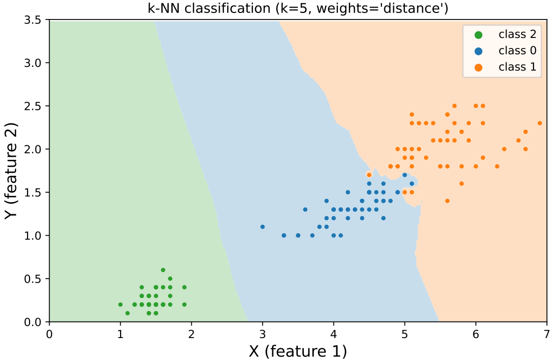
⇒ Method:

1. take random data points in the training dataset
2. for a sample find the k (e.g. 5) closest data points in the training dataset *(k is a hyperparameter)*
3. look at the neighbor labels, return/assign the mode
4. non-linear decision boundary can be recovered



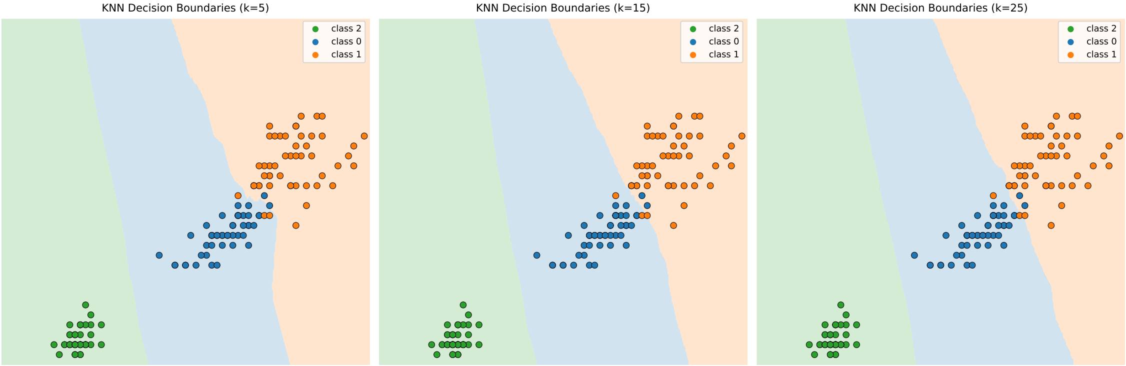
kNN classification

EX: estimate classification boundary using the kNN algorithm



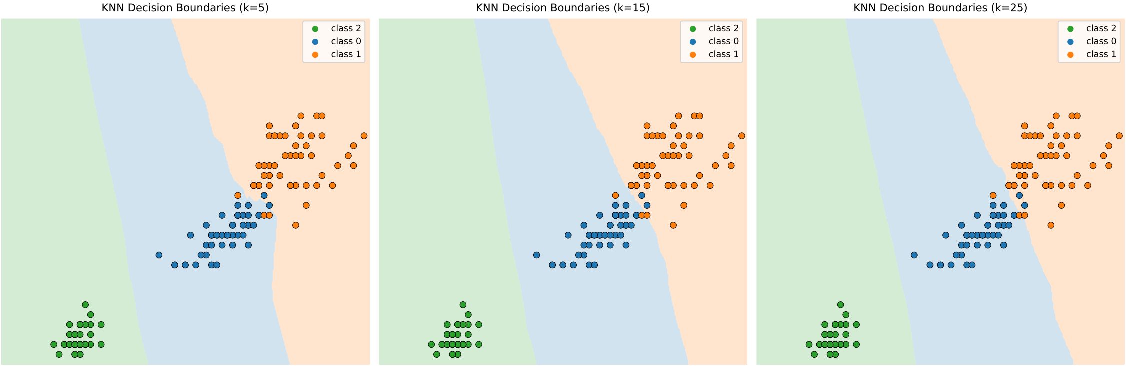
kNN classification

EX: estimate classification boundary using the kNN algorithm



kNN classification

EX: estimate classification boundary using the kNN algorithm



→ Pros: simple, easy to understand, works well with small datasets
→ Cons: slow for large datasets, sensitive to choice of distance metric and the value of k

more on classification next week !

(perceptron, SVM, PCA)