Introduction to Machine Learning

Ron Schwessinger & Edward Sanders

Hughes' Lab

MRC Molecular Haematology Unit

MRC WIMM Centre for Computational Biology

MRC Weatherall Institute of Molecular Medicine

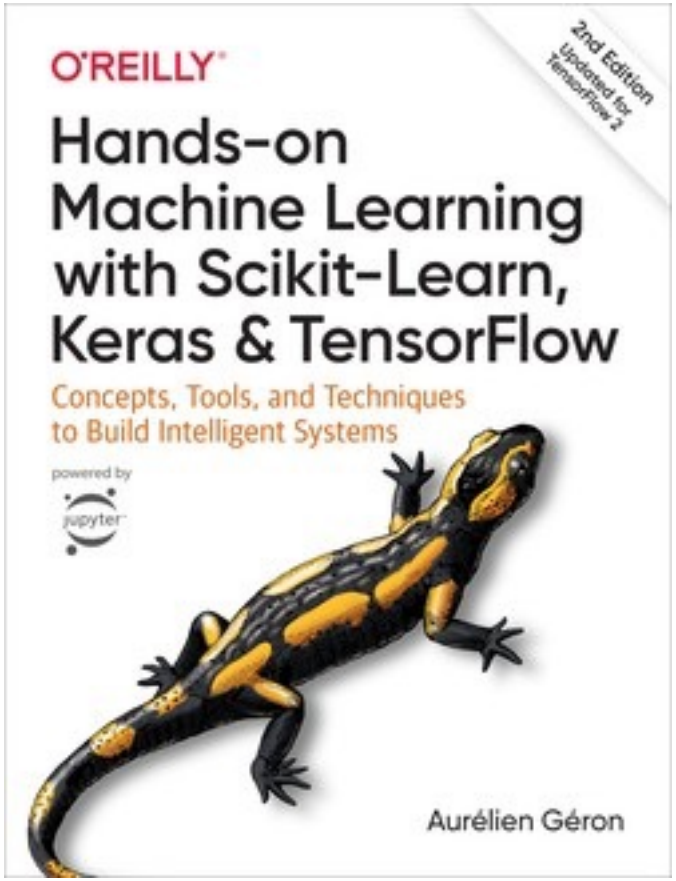
University of Oxford

GMS teaching – 06.12.2021

Outline

1) Introduction to ML
    1) Why bother?
    2) (Un- / Semi-) Supervised Learning
    3) Instance- vs. Model-based
    4) Linear Models
2) Training a ML model
    1) Linear regression as ML problem
    2) Gradient descent
    3) Regularisation
3) Classification
    1) Logistic Regression
    2) Decision trees
    3) Ensembles
    4) Neural Networks

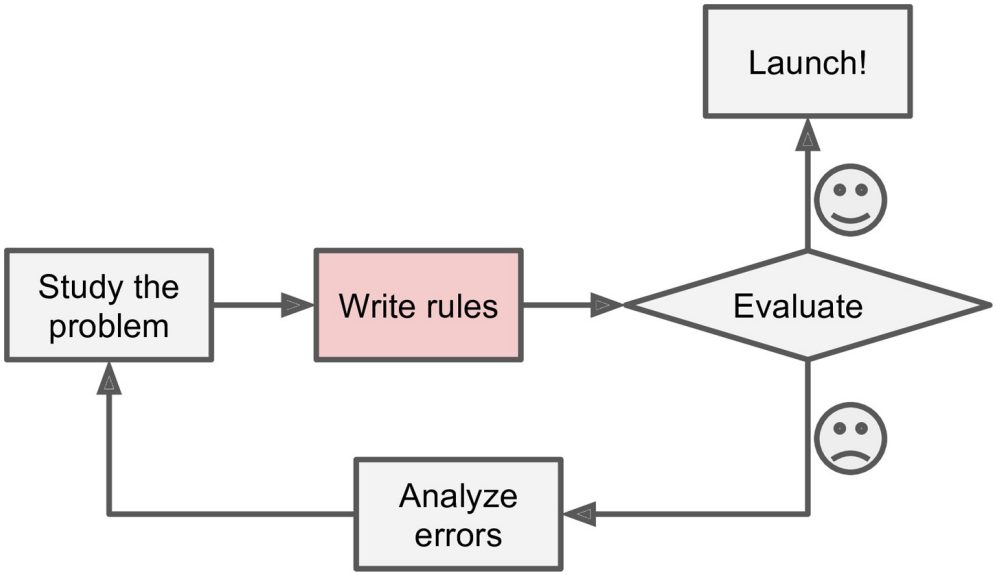
Introduction to Machine Learning



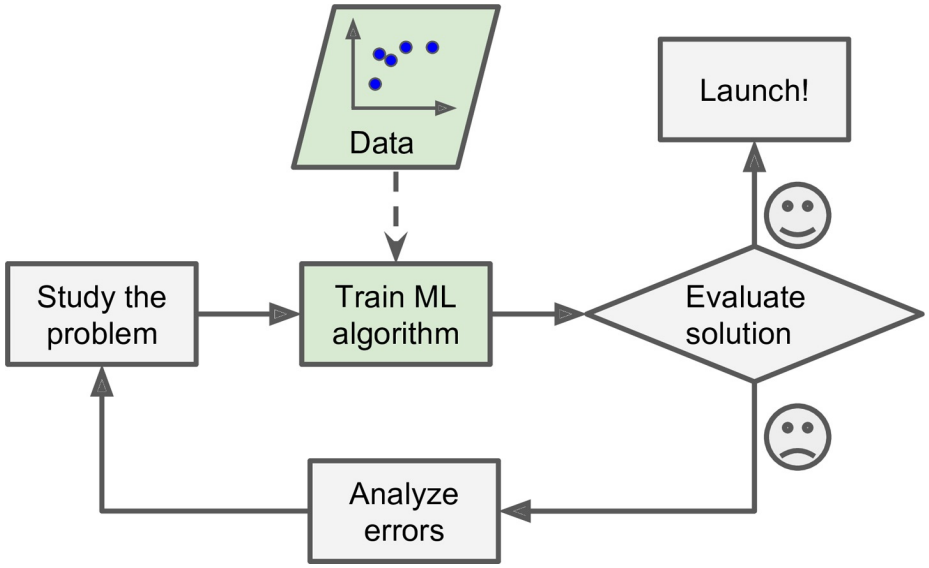
https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

Introduction to Machine Learning

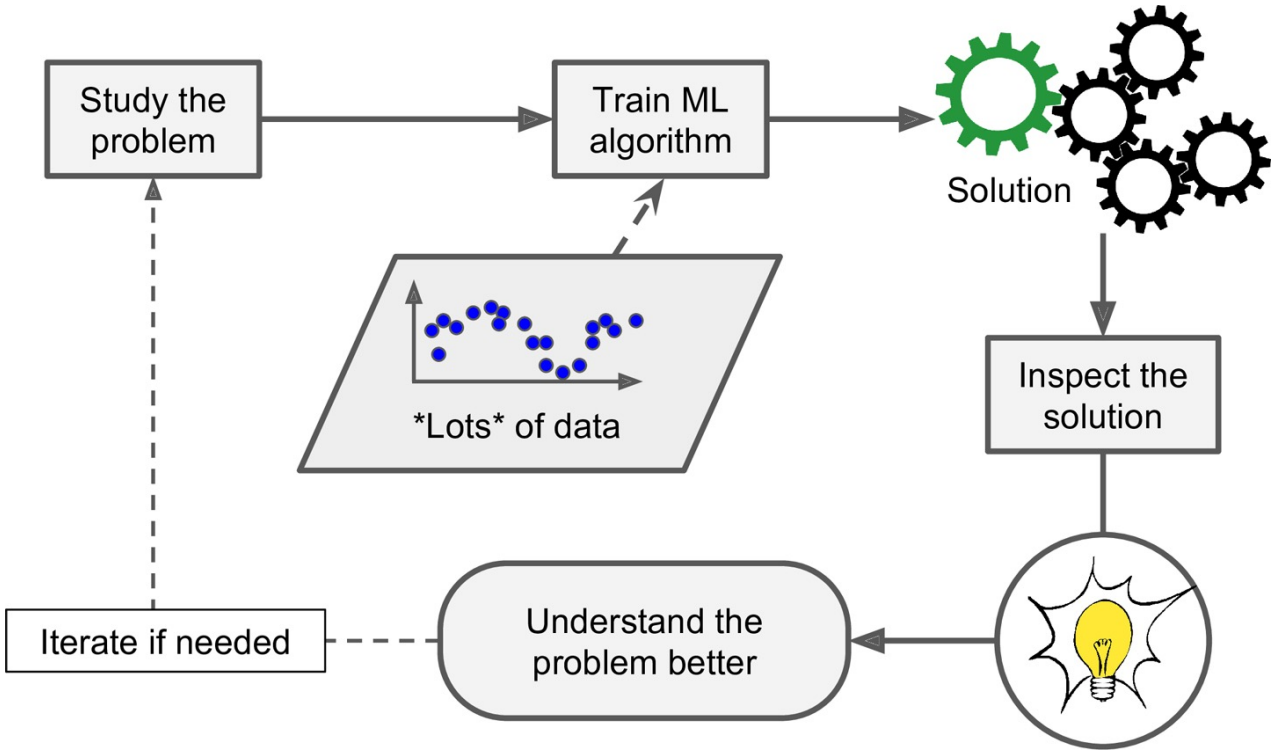
Why Bother?


Launch!
Study the problem
Write rules
Evaluate
Analyze errors

Why Bother?


Data
Launch!
Study the problem
Train ML algorithm
Evaluate solution
Analyze errors

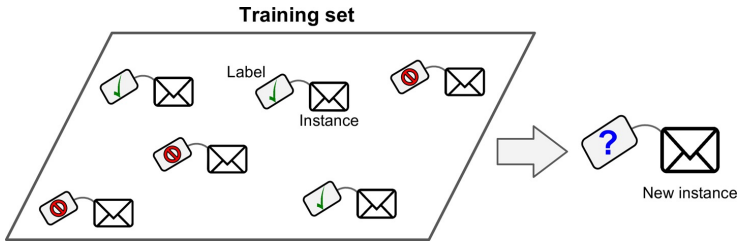
Why Bother?


Study the problem
Train ML algorithm
Solution
Lots of data
Inspect the solution
Iterate if needed
Understand the problem better

(Un- / Semi-) Supervised Learning

- Supervised

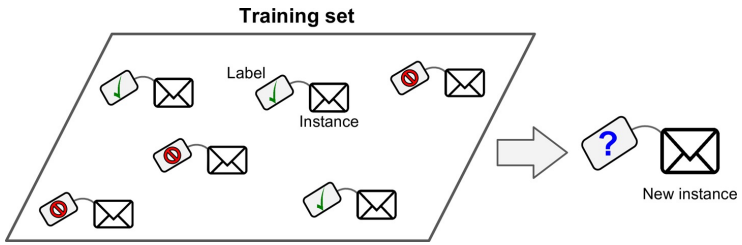
Classification



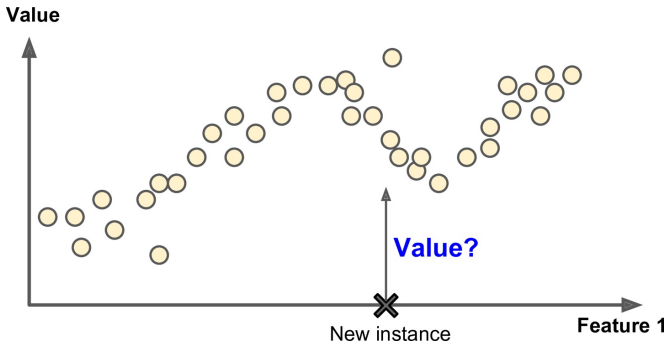
(Un- / Semi-) Supervised Learning

- Supervised

Classification



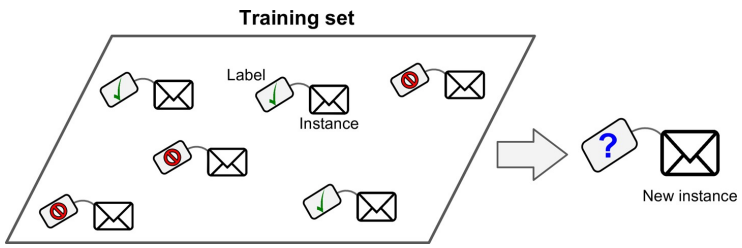
Regression



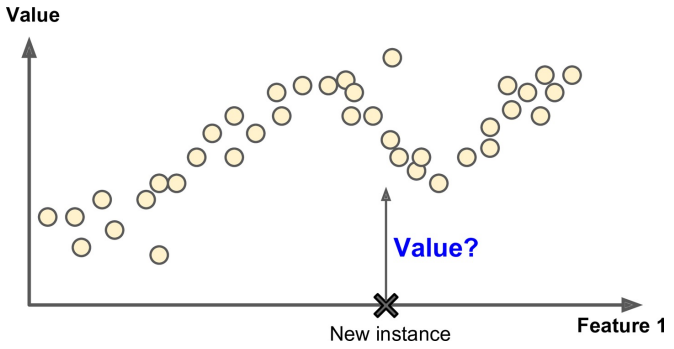
(Un- / Semi-) Supervised Learning

- Supervised

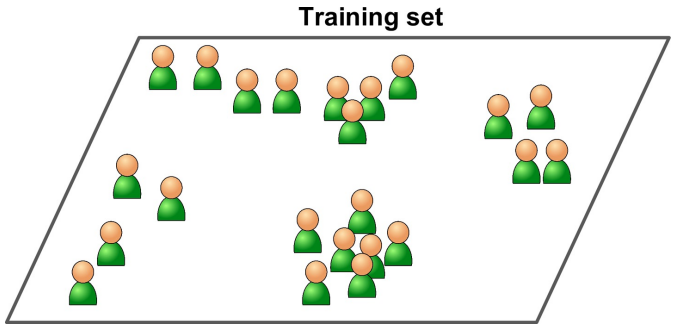
Classification

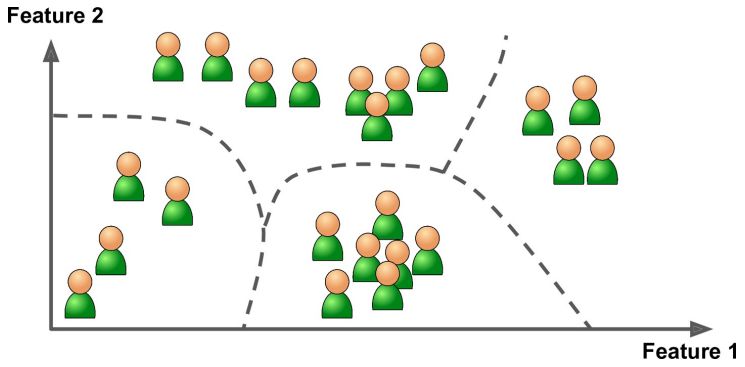


Regression



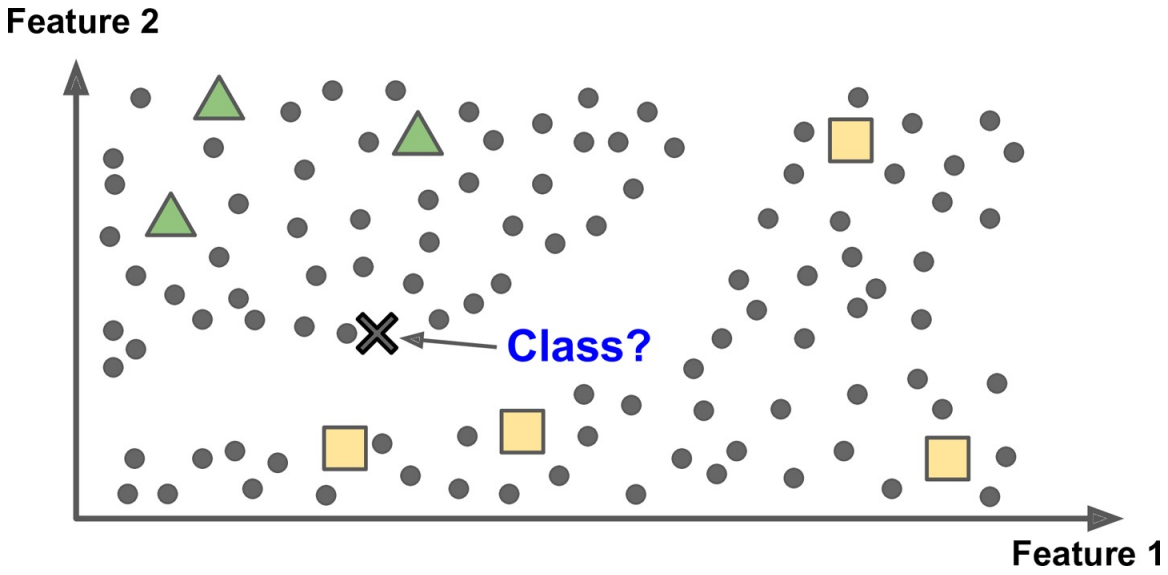
- Un-supervised





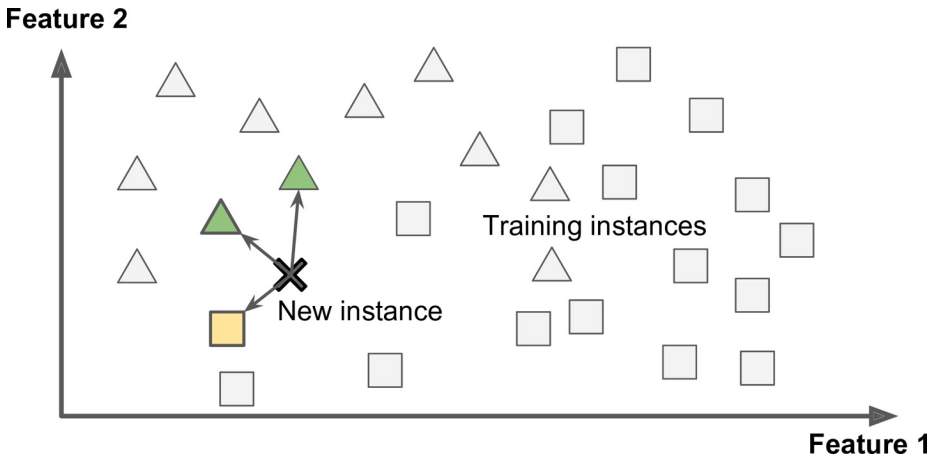
(Un- / Semi-) Supervised Learning

- Semi-supervised



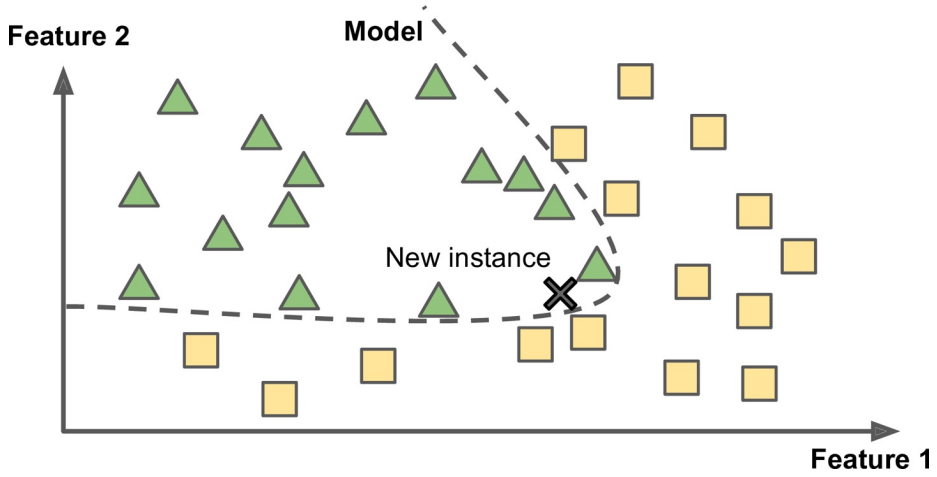
Instance- vs. Model-based ML

- Instance-based

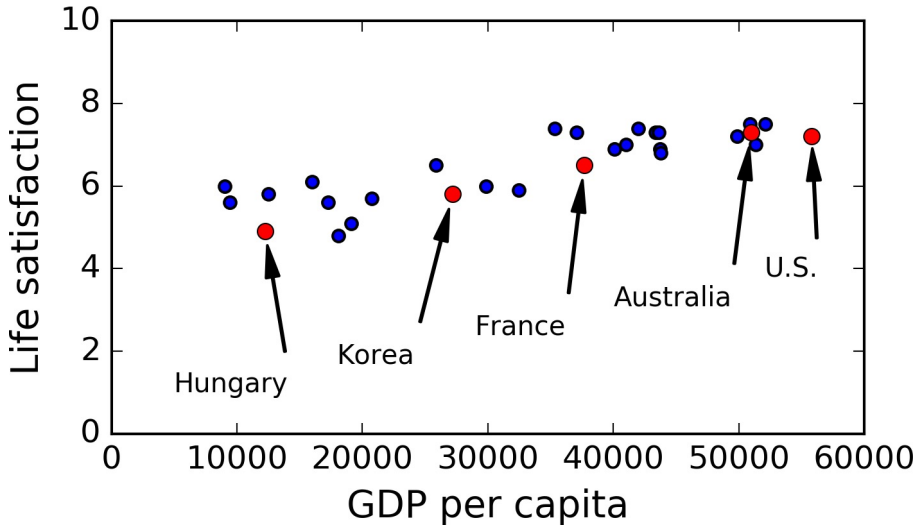


Instance- vs. Model-based ML

- Model-based



Linear Models

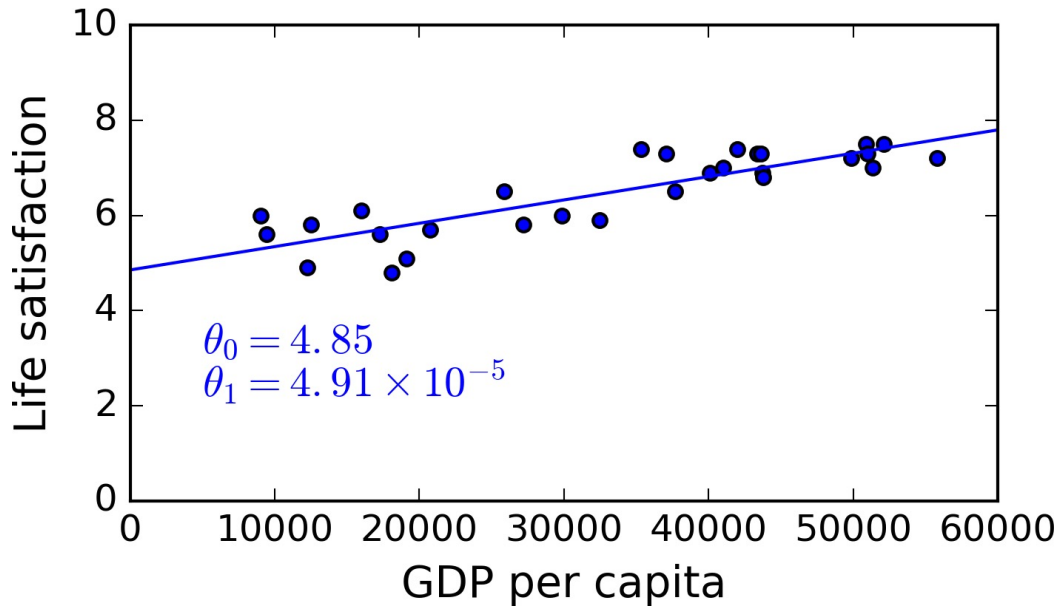


$$life\_satisfaction = \theta_0 + \theta_1\, GDP\_per\_capita$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

θ: $parameter$
x : $feature$
$\hat{y}$: $predicted\ value$

Linear Models



$$life\_satisfaction = \theta_0 + \theta_1 \, GDP\_per\_capita$$

Linear Regression as ML problem

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$

parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$

$$\hat{y} = \boldsymbol{\theta} \cdot \boldsymbol{x}$$

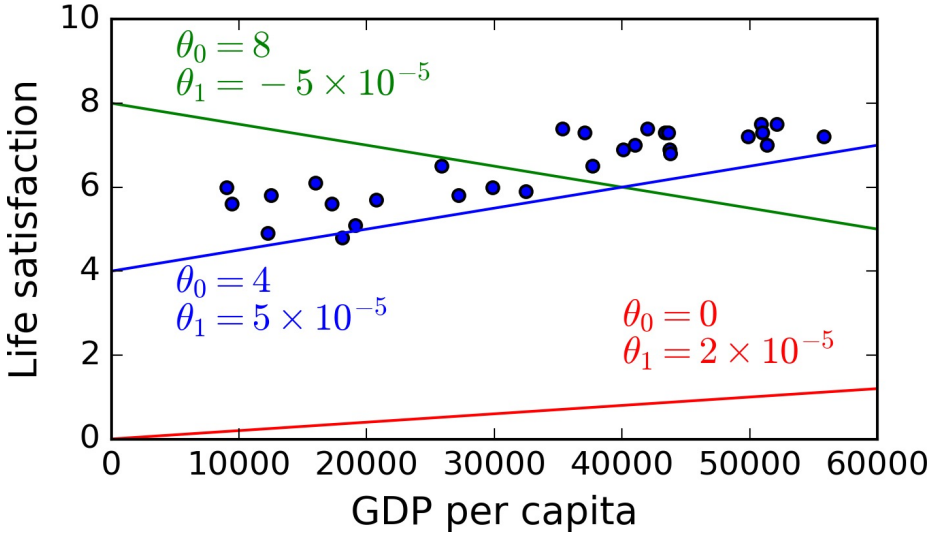
Dataset:

Samples $$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n}, \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

Labels $$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Linear Regression as ML problem

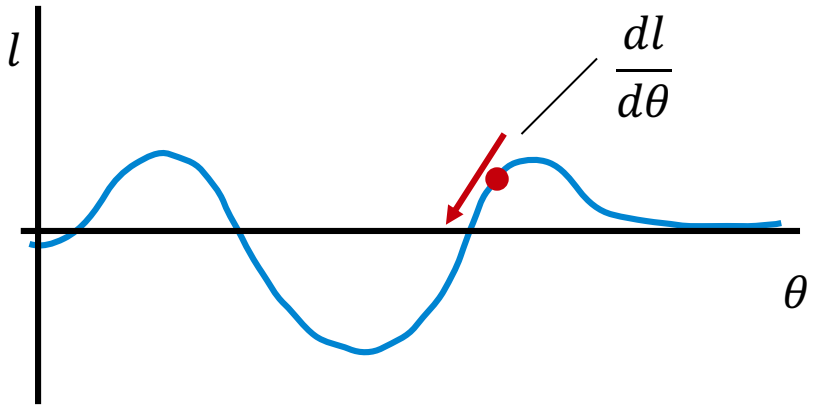
- Loss function



$$MSE(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{m_i} \sum_{i=1}^{m} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i)$$

Gradient Descent

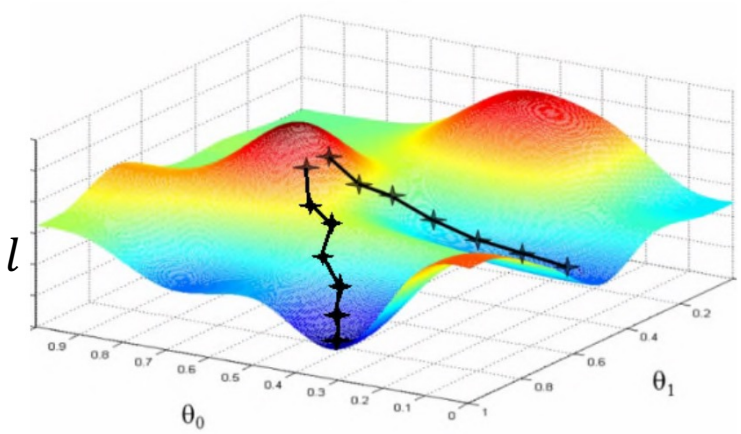
- Loss function → calculate the derivative of the loss function with respect to a parameter
- Adjust parameter in the direction of the gradient
- Partial derivatives with respect to each parameter



Gradient Descent

- Loss function → calculate the derivative of the loss function with respect to a parameter
- Adjust parameter in the direction of the gradient
- Partial derivatives with respect to each parameter

$$\nabla_\theta MSE(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\boldsymbol{\theta}) \end{pmatrix}$$



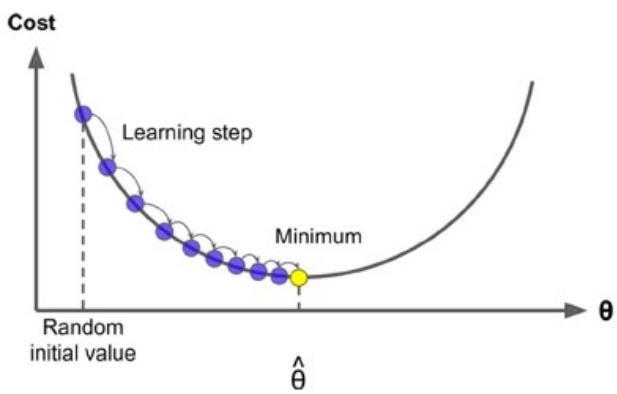
Gradient Descent

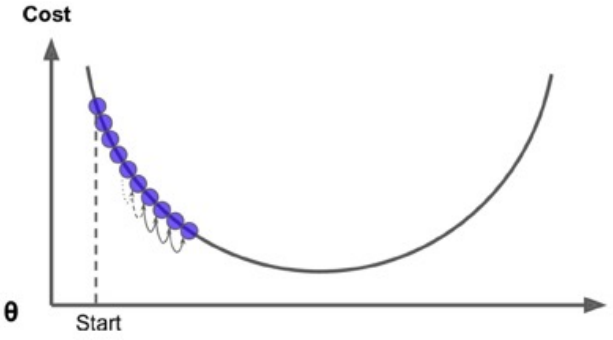
- Loss function → calculate the derivative of the loss function with respect to a parameter
- Adjust parameter in the direction of the gradient
- Partial derivatives with respect to each parameter
- How far should each step be? - Select an appropriate learning rate

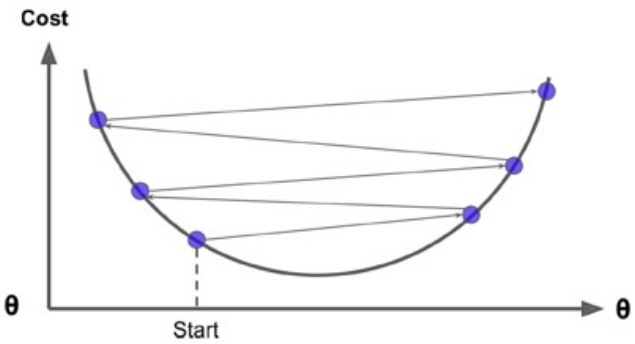
$$\boldsymbol{\theta}^{(next\ step)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta})$$

learning rate: η

Learning Rate

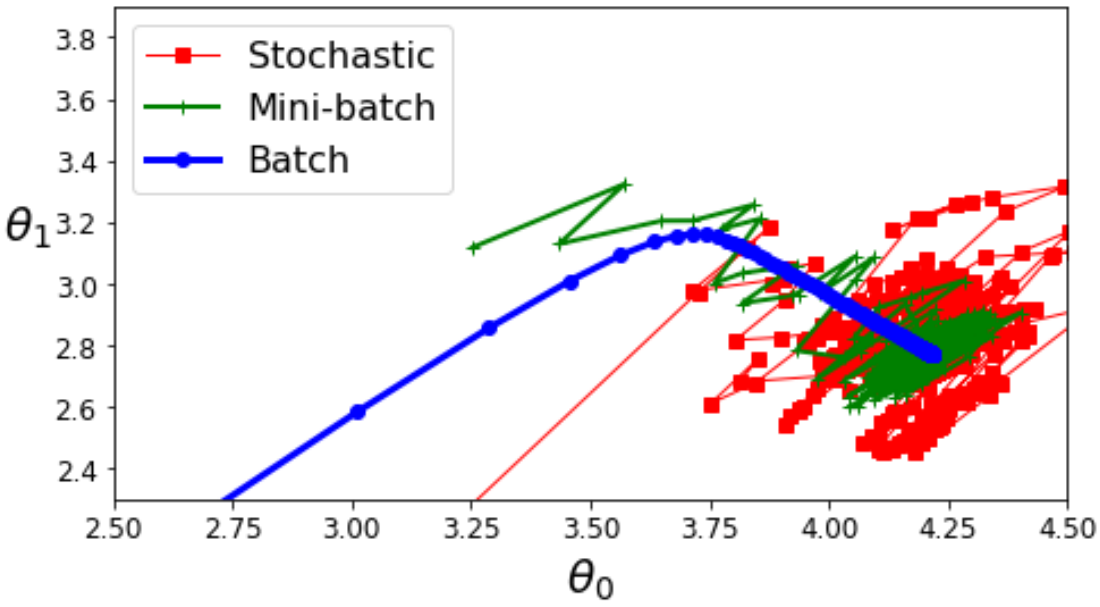




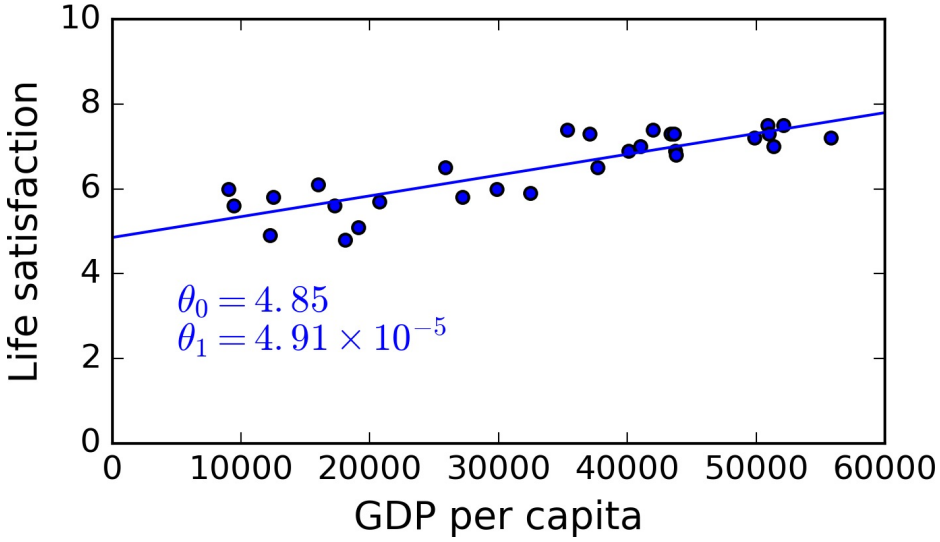


Gradient Descent

- Batch – whole training set at a time
- Stochastic – one training example at a time
- Mini-batch – a mini batch (set) of training examples at a time

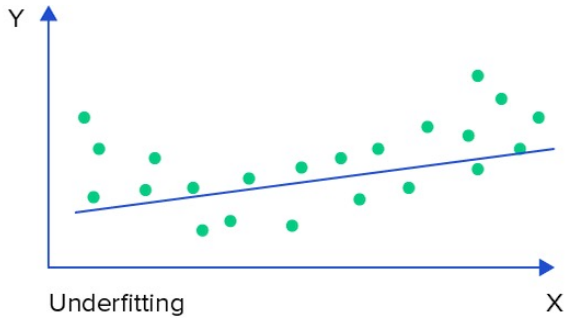


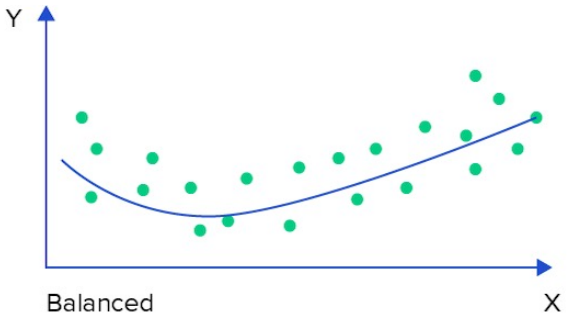
Overfitting / Underfitting

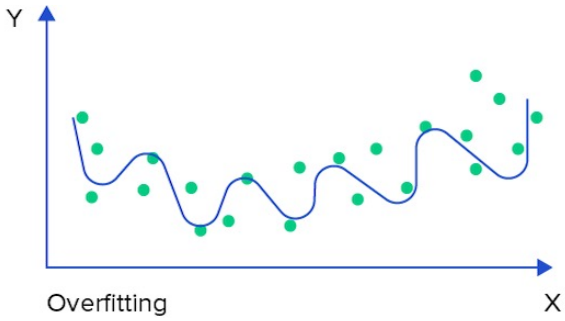


$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \ldots + \theta_n x_n^n$$

Overfitting / Underfitting

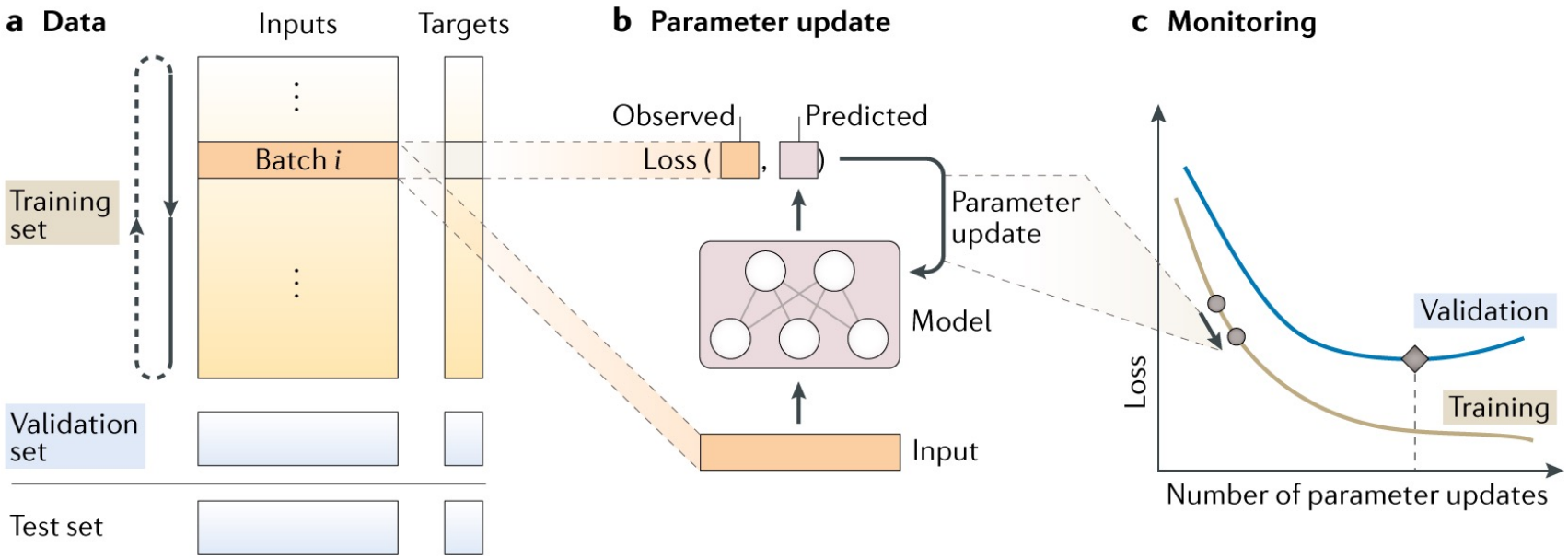






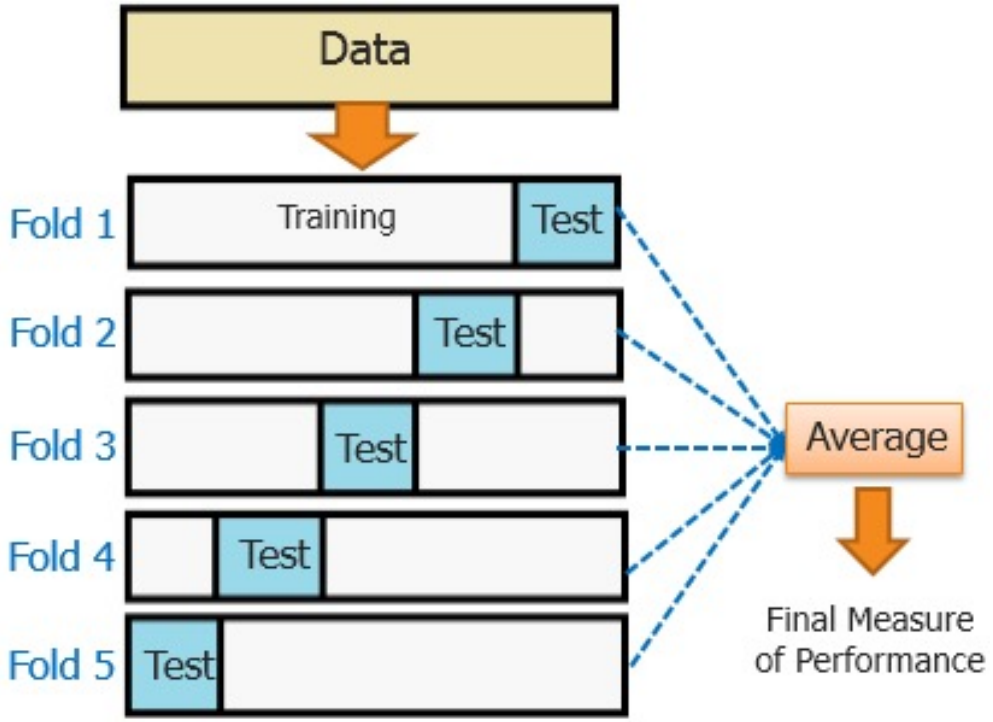
Source towardsdatascience.com

Test & Validation Sets



Gökcen *et al.* Nature Reviews Genetics 2019

Cross-Validation



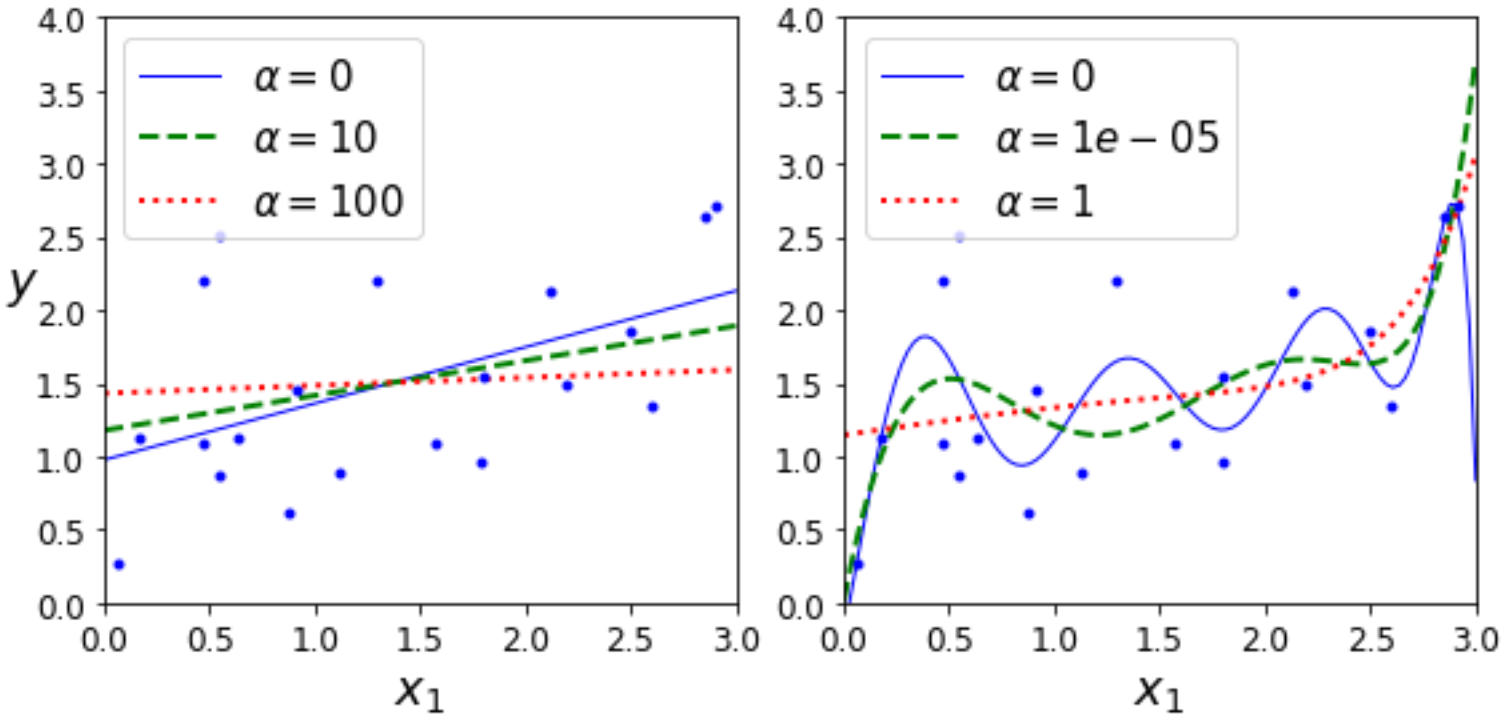
Source blog.contactsunny.com

Regularisation

- Early stopping
- Ridge (l2): $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$

Regularisation

- Early stopping
- Ridge (l2): $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$

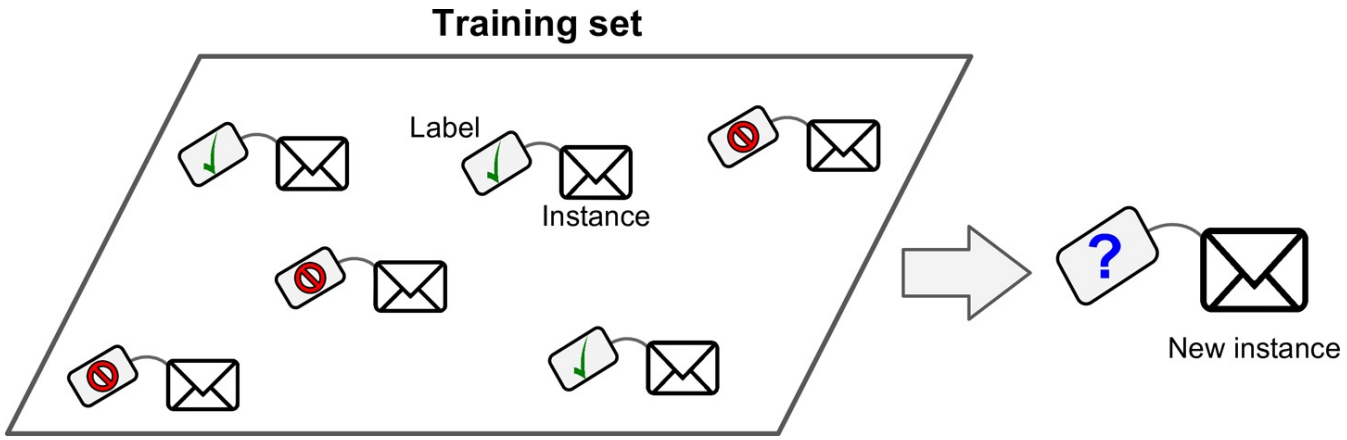


Regularisation

- Early stopping
- Ridge (l2): $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$
- Lasso (l1): $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} |\theta_i|$
- Dropout …

Coffee Break!

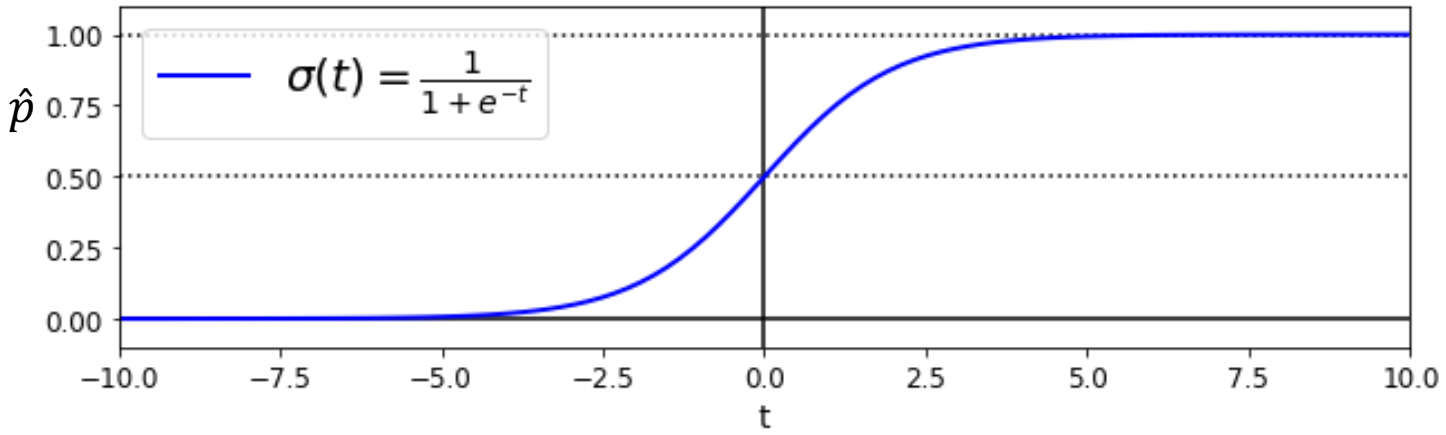
Classification



Logistic Regression

Predict class probabilities for 1 (2) class(es)

$$\hat{p} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \ldots + \theta_n x_n^n)$$



Model prediction

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p} < 0.5 \\ 1 \; if \; \hat{p} \geq 0.5 \end{cases}$$

Logistic Regression

Loss function (one instance)

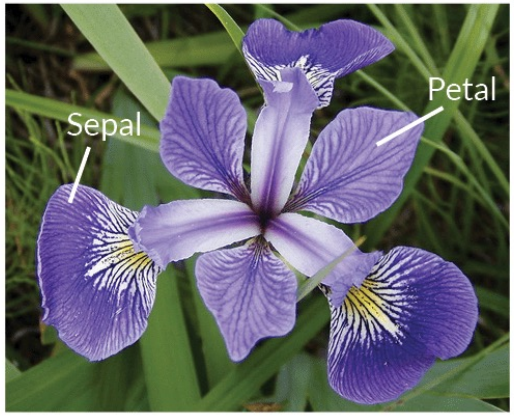
$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & if\ y = 1 \\ -\log(1 - \hat{p}\ ) & if\ y = 0 \end{cases}$$

Log loss

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{p}) + (1 - y^{(i)})\log(1 - \hat{p})\right]$$

Softmax Regression

Predict class probabilities for *k* classes



Iris Versicolor

Iris Setosa

Iris Virginica

Softmax Regression

Predict class probabilities for *K* classes

Softmax score for class k $s_k = \boldsymbol{x}^T \boldsymbol{\theta}^{(k)}$

Softmax Regression

Predict class probabilities for *K* classes

Softmax score for class k $s_k = \boldsymbol{x}^T \boldsymbol{\theta}^{(k)}$

Softmax function $\hat{p} = \sigma\big(\boldsymbol{s}(\boldsymbol{x})\big) = \dfrac{\exp(s_k(\boldsymbol{x}))}{\sum_{j=1}^{K} \exp(s_j(\boldsymbol{x}))}$

$\boldsymbol{s}(\boldsymbol{x})$: vector of scores for each class for instance x

Softmax Regression

Predict class probabilities for *K* classes

Softmax score for class k $s_k = \boldsymbol{x}^T \boldsymbol{\theta}^{(k)}$

Softmax function $\hat{p} = \sigma\big(\boldsymbol{s}(\boldsymbol{x})\big) = \frac{\exp(s_k(\boldsymbol{x}))}{\sum_{j=1}^{K} \exp(s_j(\boldsymbol{x}))}$

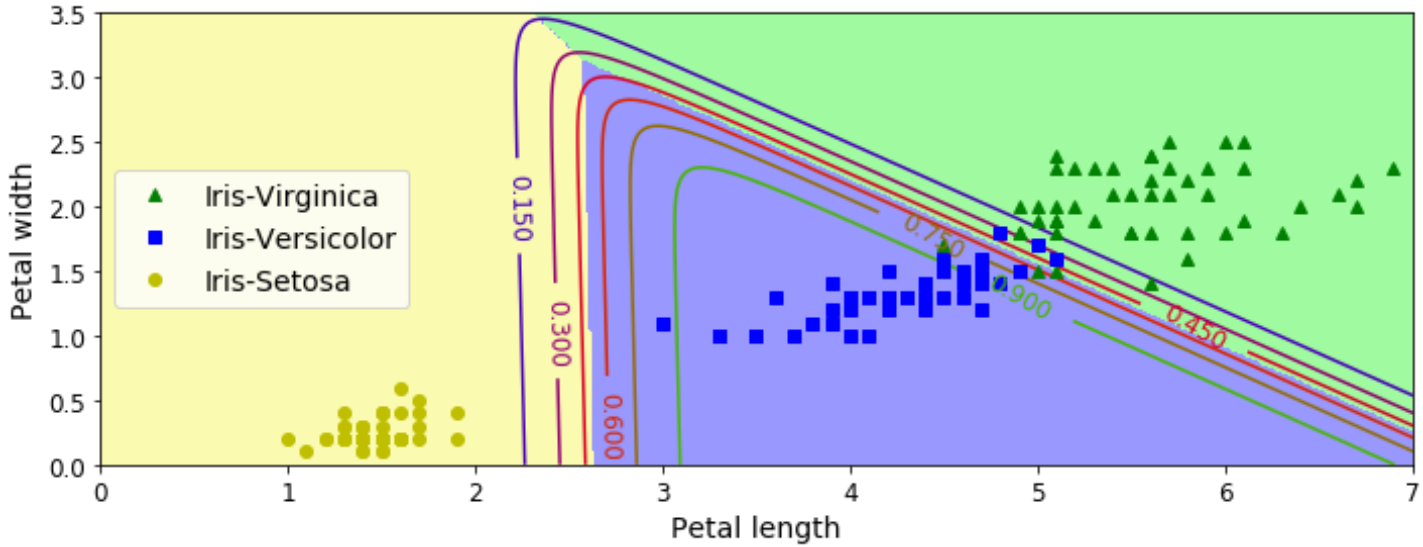
$\boldsymbol{s}(\boldsymbol{x})$: vector of scores for each class for instance x

Cross entropy $J(\boldsymbol{\Theta}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)})$

$\boldsymbol{\Theta}$: parameter matrix $y_k^{(i)}$: probability that i[th] instance belongs to class *k*

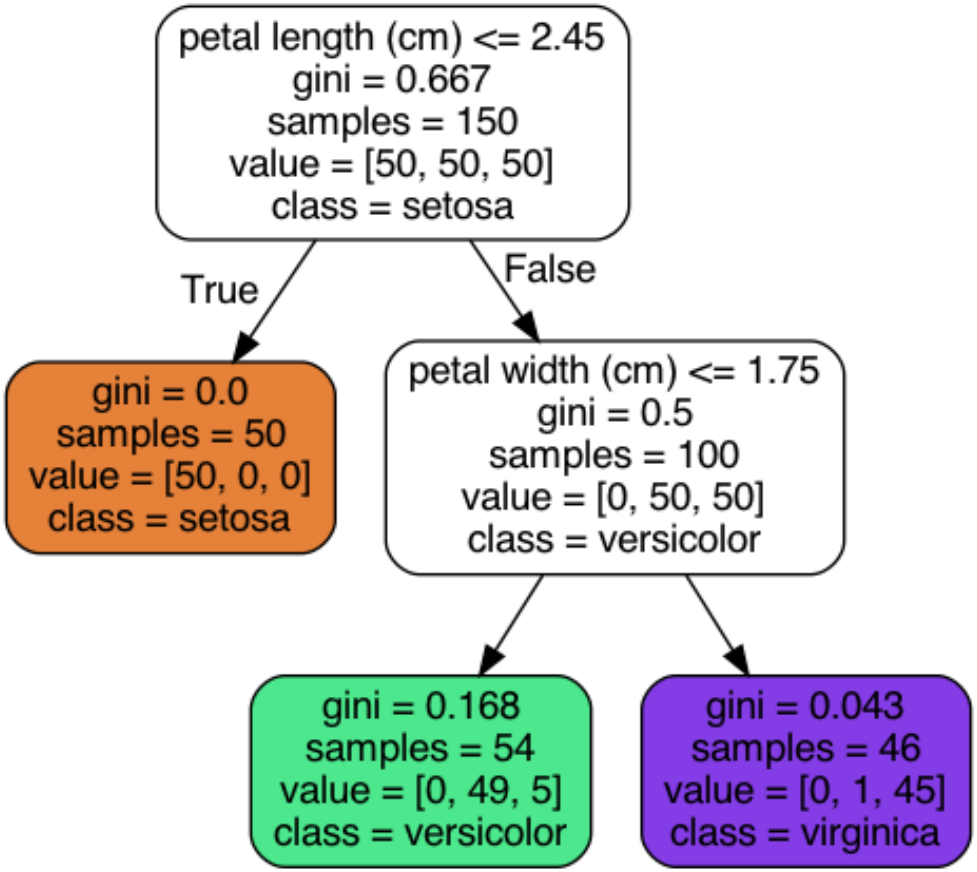
Softmax Regression

Predict class probabilities for *k* classes

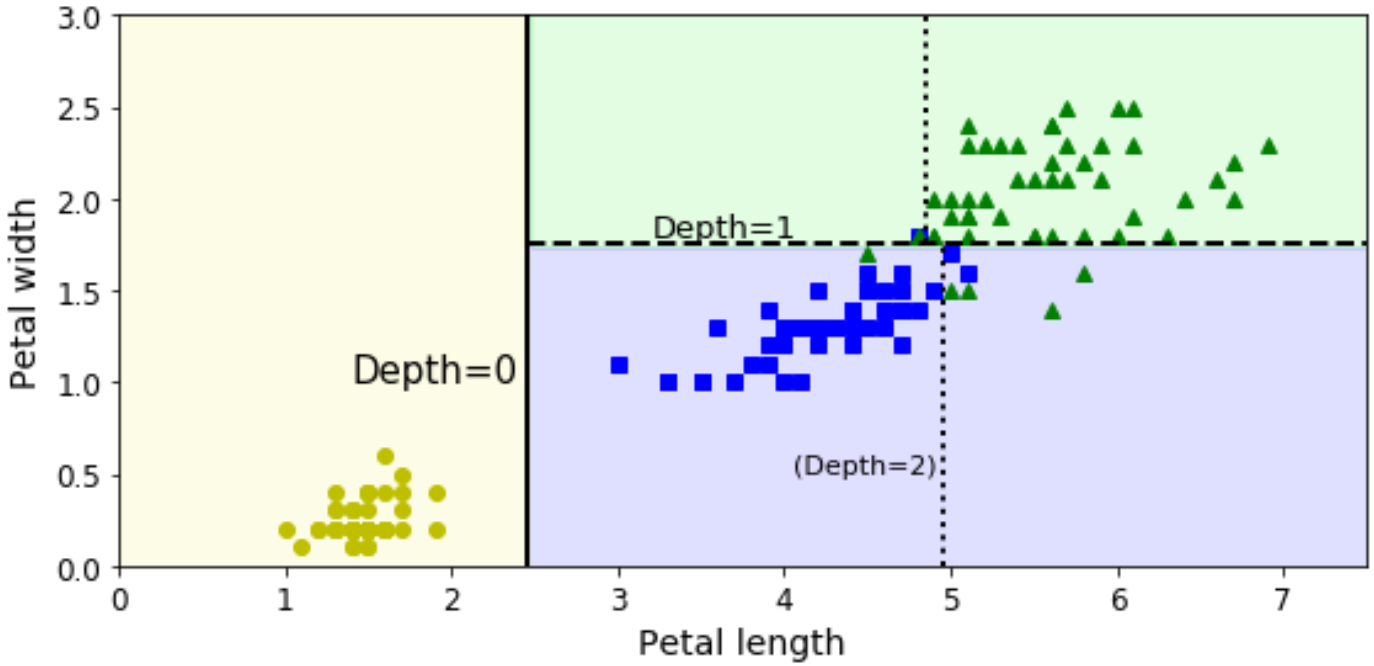


Decision Trees

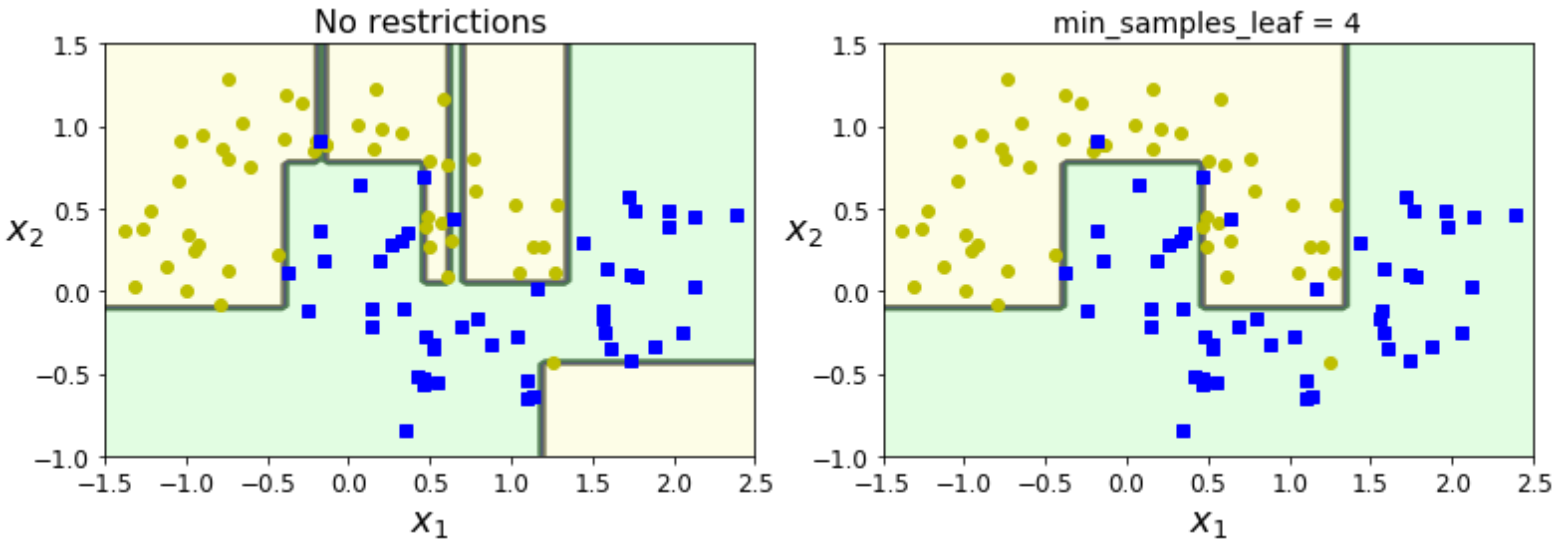
Decision Trees



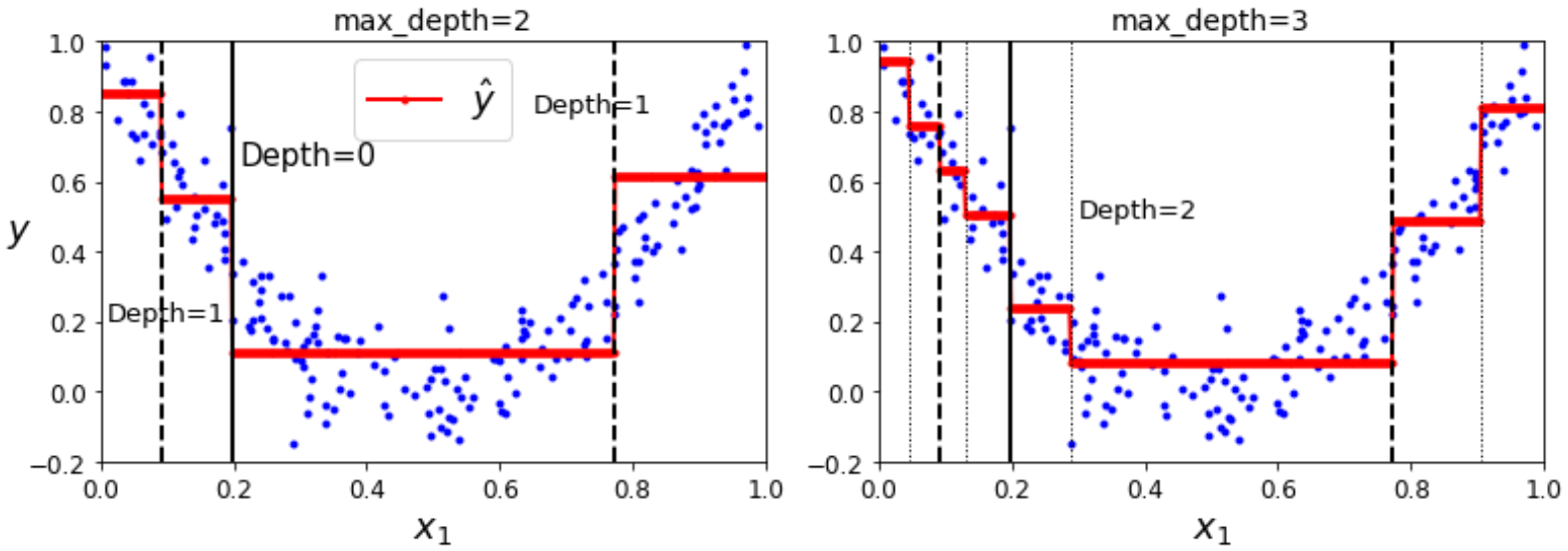
Decision Trees



Decision Trees



Decision Trees



Ensembles

Ensembles

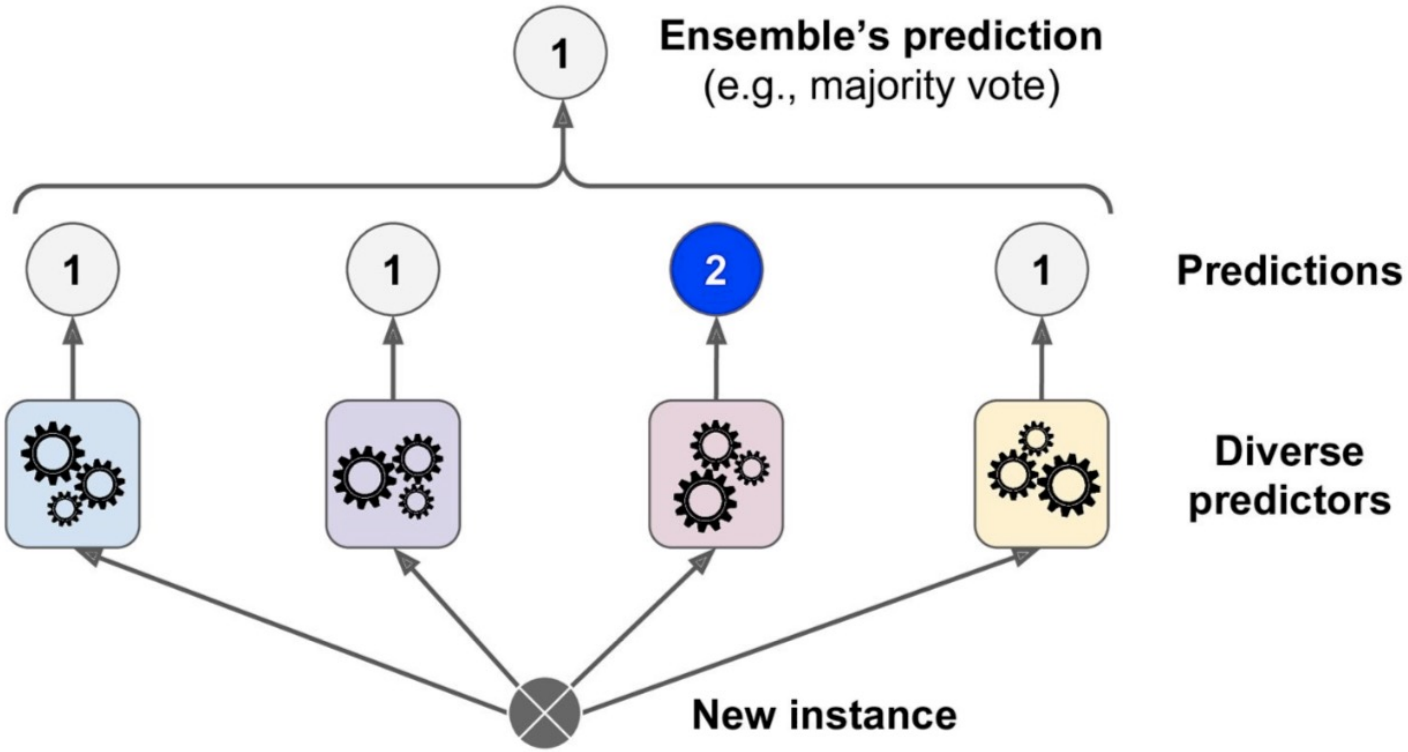
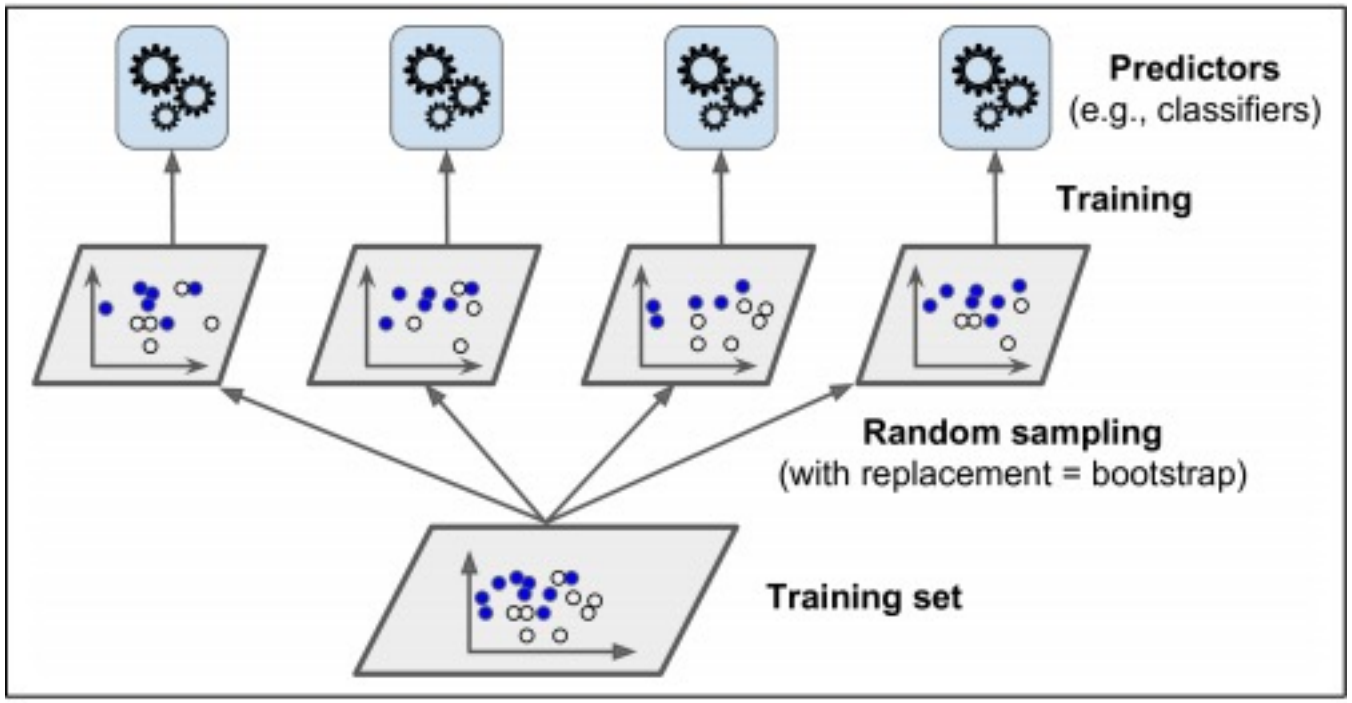


Figure 7-2. Hard voting classifier predictions

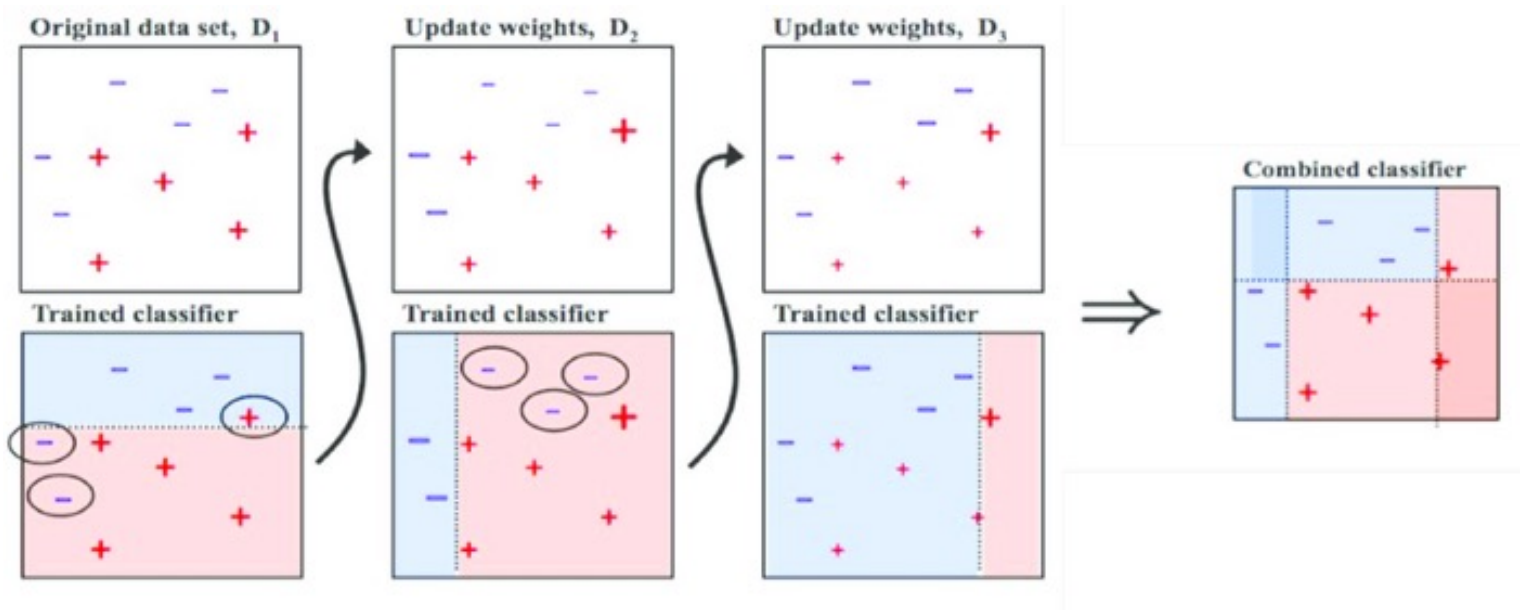
Ensembles

- Works best if single models are as independent as possible:
- Bagging, Pasting



Ensembles

- Works best if single models are as independent as possible:
- Boosting
  - AdaBoost – upweight miss classified instances for subsequent models



Ensembles

- Boosting
  - Gradient boosting – train classifiers on the residuals of initial predictions