

Machine Learning with Python

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu
2023 Spring

Day 2: **Supervised** learning

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu
2023 Spring

Agenda

- Day 1: Introduction to machine learning
 - Some key concepts in machine learning
 - Jupyter notebook and some packages usage
- Day 2: **Supervised** learning
 - Classification
 - Regression
 - Regularization
- Day 3: **Unsupervised** learning
 - Dimension reduction
 - Clustering



Notation of the slides

- Code or Pseudo-Code chunk starts with "➤", e.g.
➤ `print("Hello world!")`
- [Link](#) is underlined
- Important terminology is in **bold** font

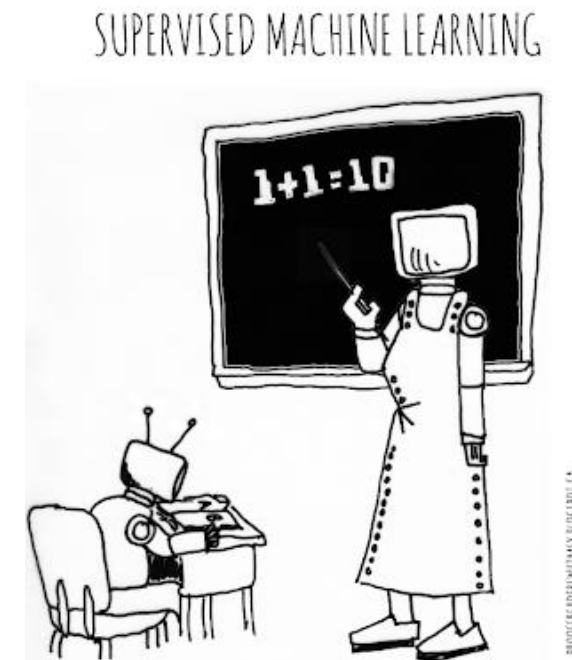
Overview

Time

- 3-hour workshop (45min + 45min + 30min + practice/Q&A)

Topics

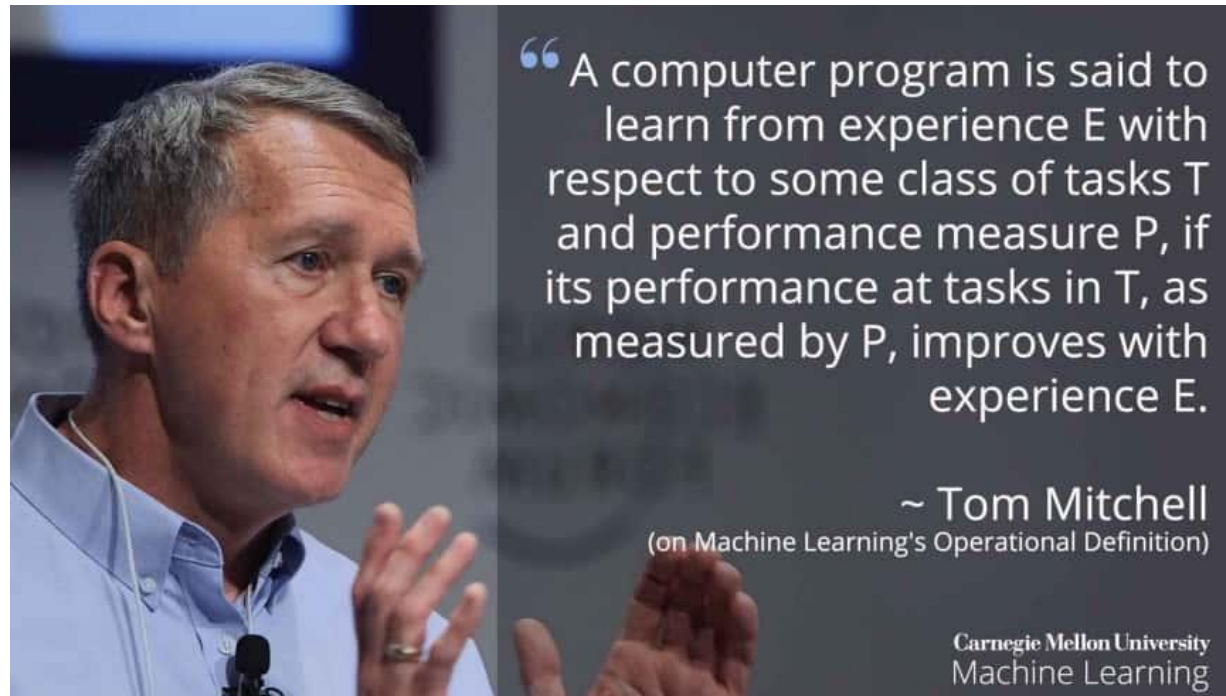
- ☐ Classification algorithms
- ☐ Performance measure
- ☐ Overfitting & underfitting
- ☐ Examples and practices



Summary – Day1

Key concepts in machine learning:

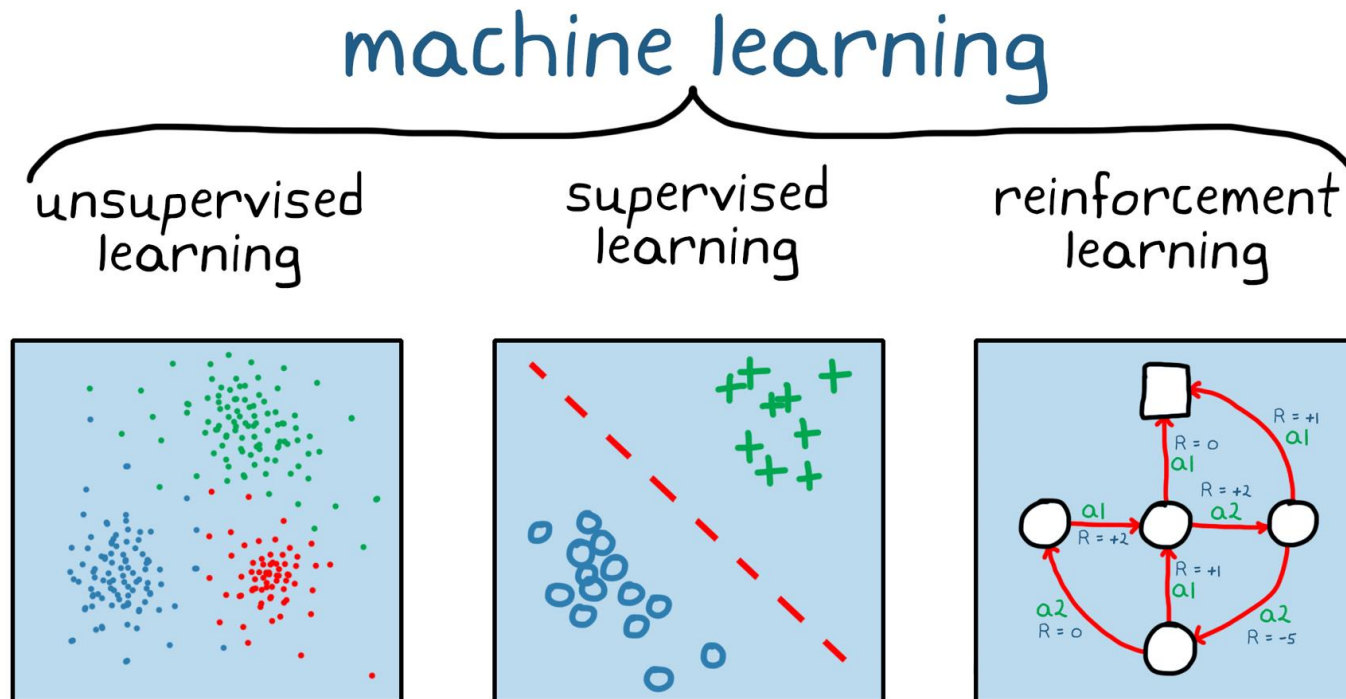
- ❑ What's machine learning



Summary – Day1

Key concepts in machine learning:

- ❑ What's machine learning
- ❑ 3 types of machine learning



Summary – Day1

Key concepts in machine learning:

- ❑ What's machine learning
- ❑ 3 types of machine learning
- ❑ The big picture of training a machine learning model



Summary – Day1

Key concepts in machine learning:

- ☐ What's machine learning
- ☐ 3 types of machine learning
- ☐ The big picture of training a machine learning model

More details about:

- ☐ Training/test set
- ☐ Loss function
- ☐ Overfitting/underfitting
- ☐ Hyperparameters tuning
- ☐ Cross validation
- ☐ Challenges in machine learning

Summary – Day1

Key concepts in machine learning:

- ☐ What's machine learning
- ☐ 3 types of machine learning
- ☐ The big picture of training a machine learning model

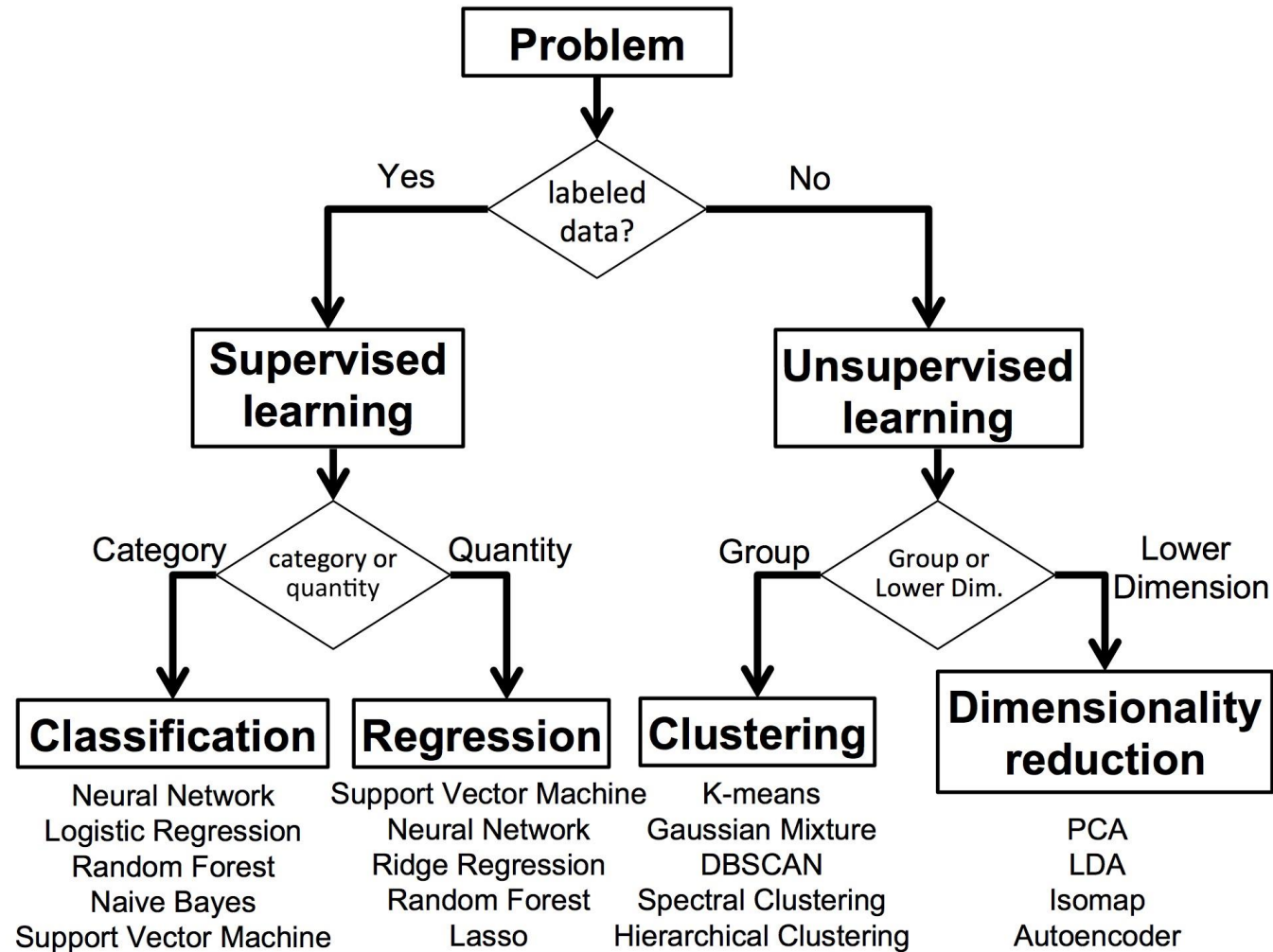
More details about:

- ☐ Training/test set
- ☐ Loss function
- ☐ Overfitting/underfitting
- ☐ Hyperparameters tuning
- ☐ Cross validation
- ☐ Challenges in machine learning

Practice:

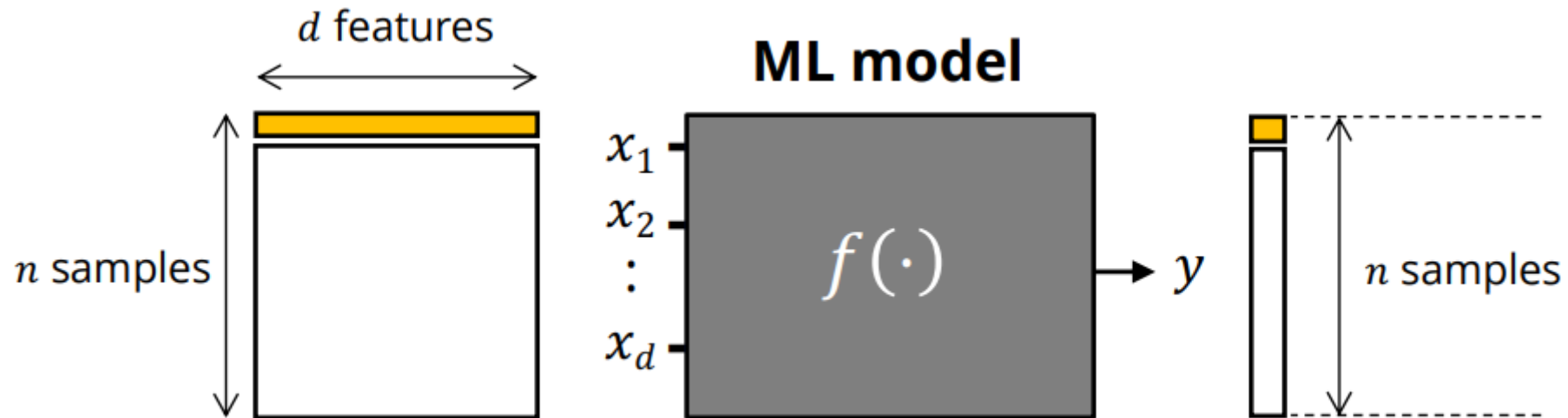
- ☐ Jupyter notebook usage
- ☐ Some useful libraries
- ☐ A supervised learning example

Types of machine learning



Supervised learning

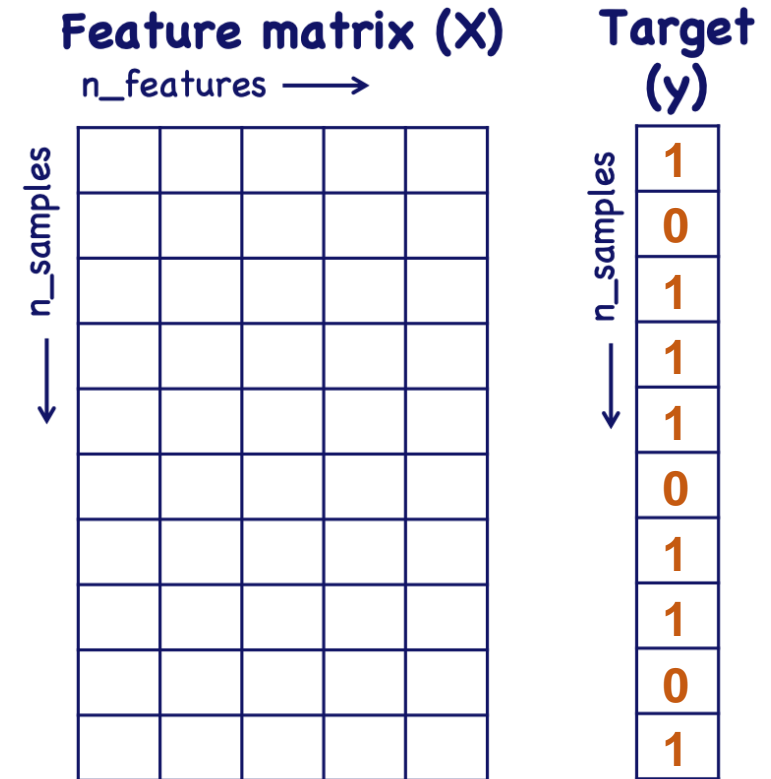
- Training data with n ***samples*** of ***features*** x and ***labels*** y
- Learn a function class $f(x)$ to describe y based on x



Different choice of $f()$ for classification tasks

- Logistic regression
- K-nearest neighbor
- Naïve bayes
- Support vector machine
- Decision trees
- Random forest
- Adaboost
- Gradient boosting (XGBoost)
- Neural network

...



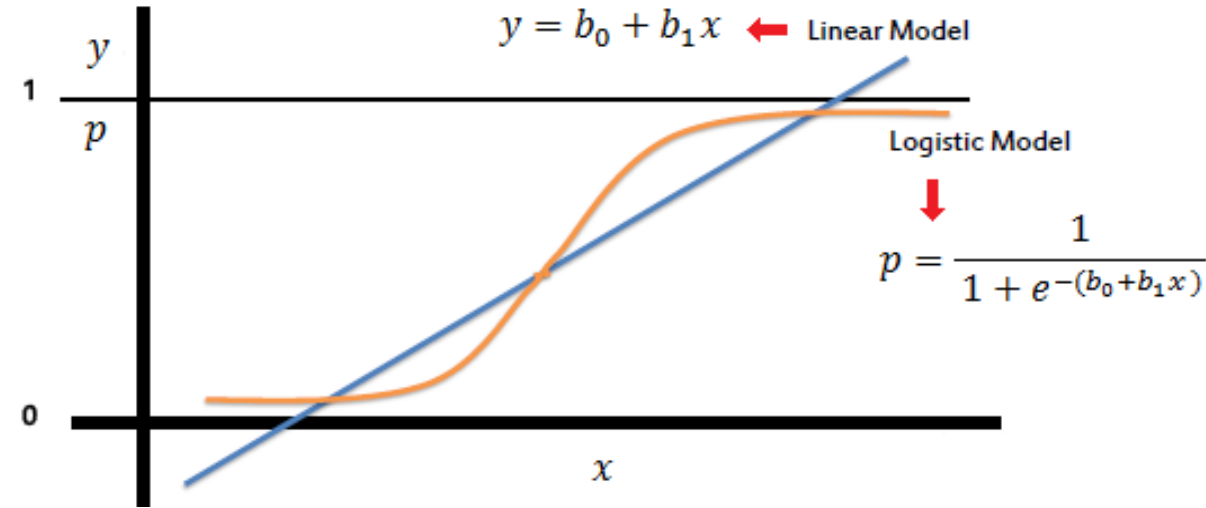
$$\hat{y} = f(x)$$

Logistic regression

- Model $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

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

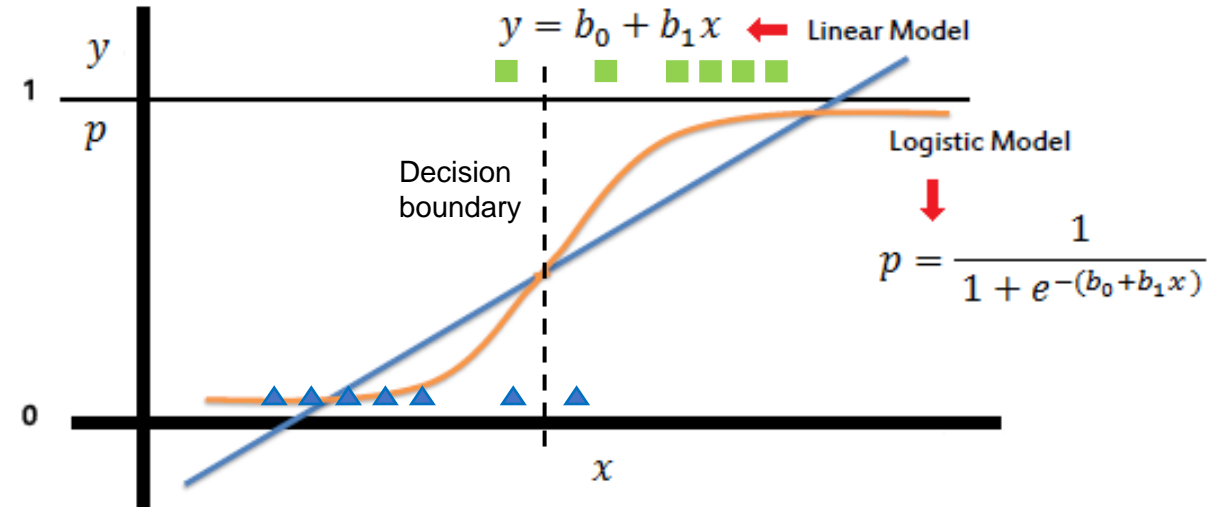


Logistic regression

- Model $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

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



Logistic regression

- Model $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$

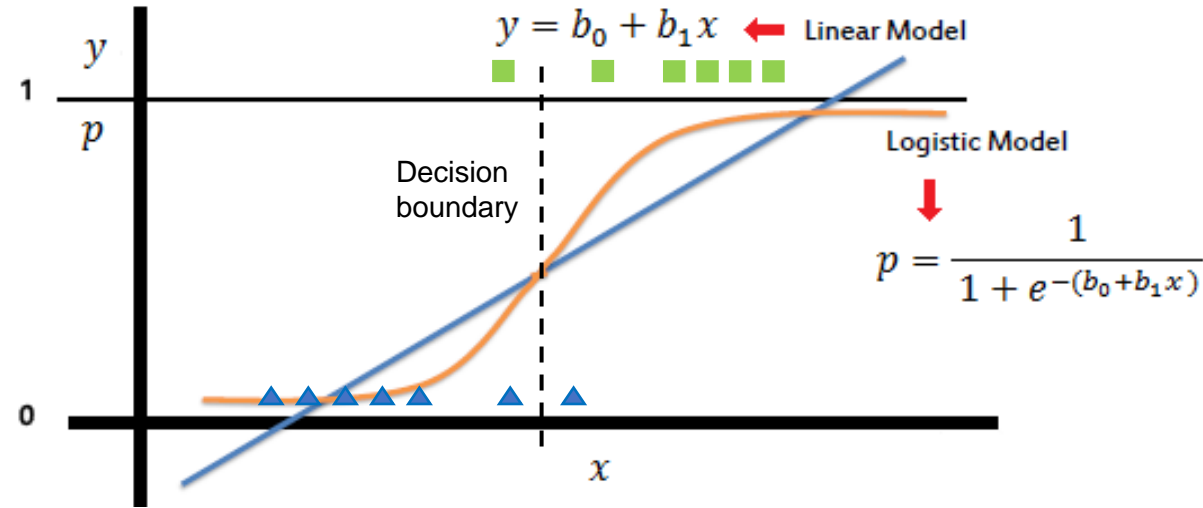
$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

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

- Loss function:

Equation 4-16. Cost function of a single training instance

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

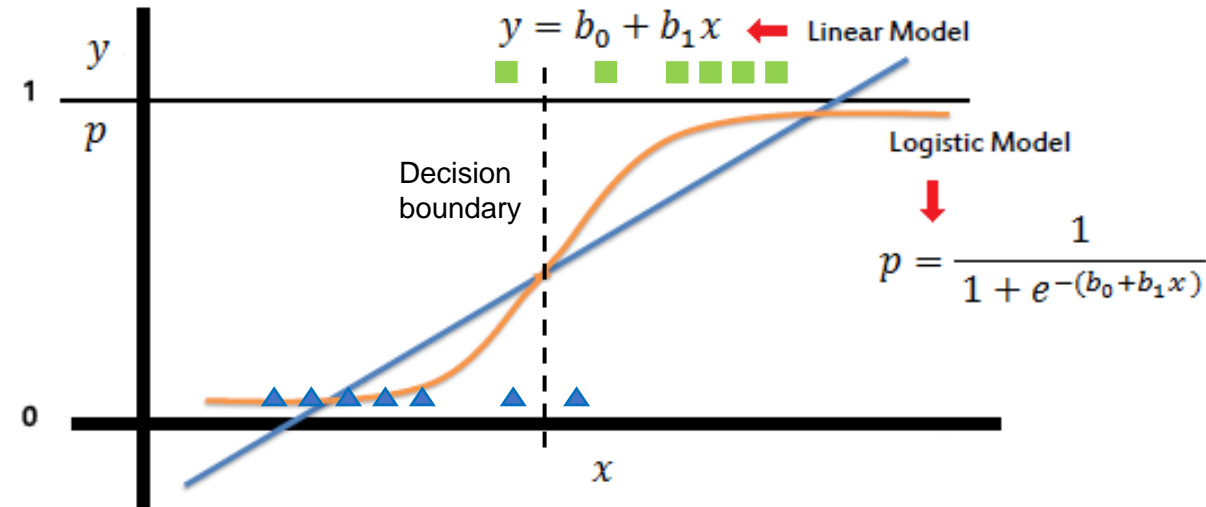


Logistic regression

- Model $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

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



- Loss function:

Equation 4-16. Cost function of a single training instance

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

Equation 4-17. Logistic Regression cost function (log loss)

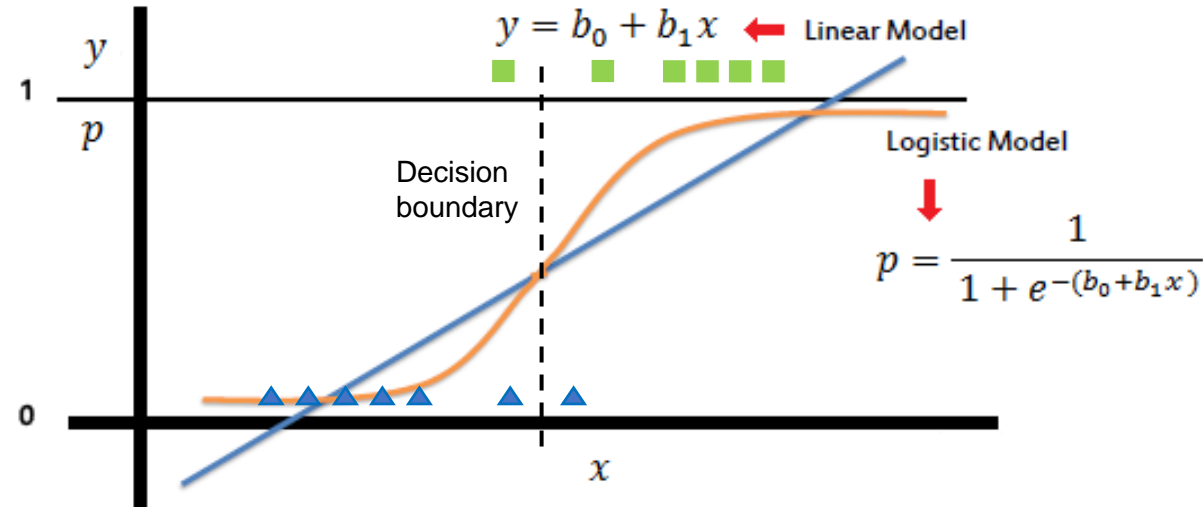
$$J(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \right]$$

Logistic regression

- Model $\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

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



- Loss function (generalize to multi-class):

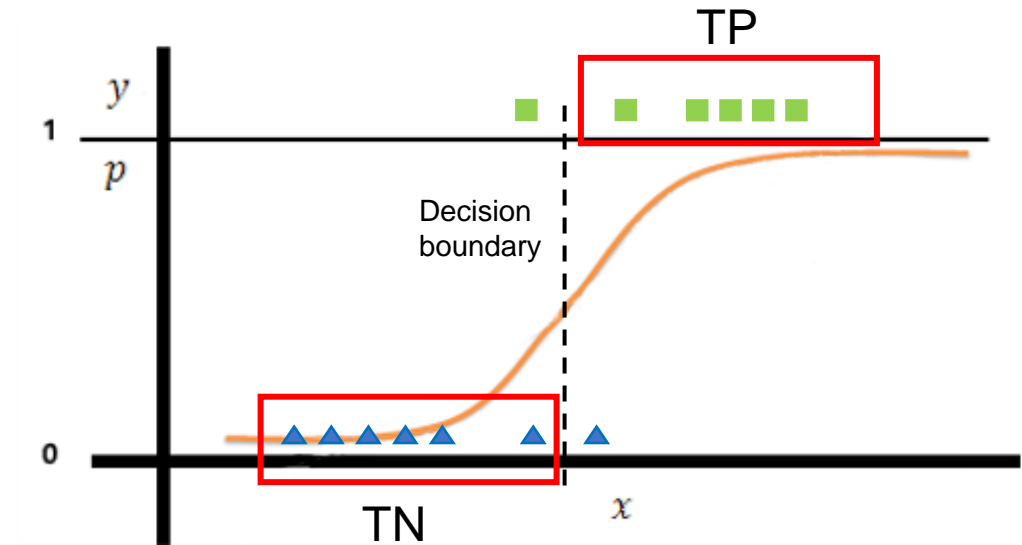
Equation 4-22. Cross entropy cost function

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

Performance measure

- Accuracy
$$\frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Q: Is accuracy always a good measure?

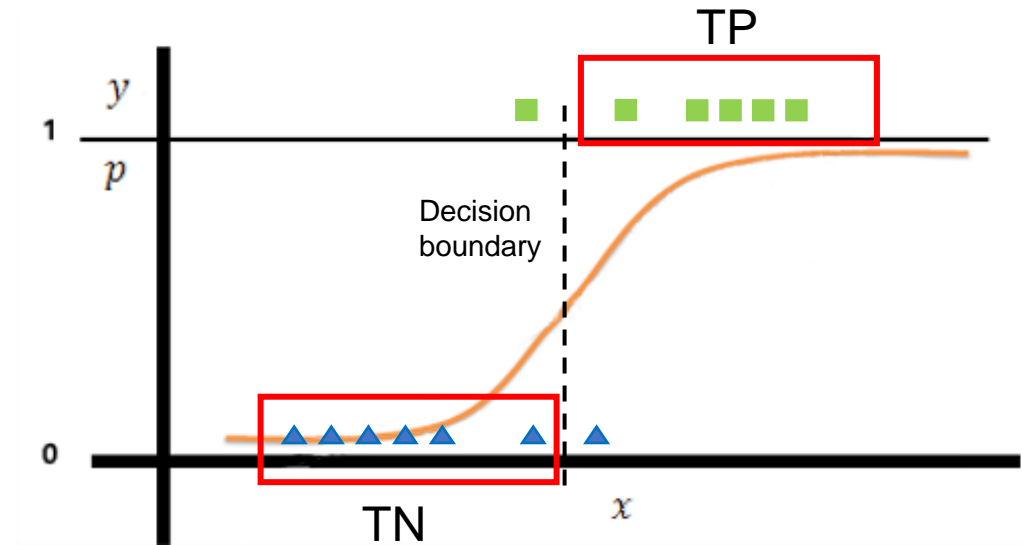


Performance measure

- Accuracy
$$\frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Q: Is accuracy always a good measure?

Be cautious with *skewed dataset* !!!



Performance measure

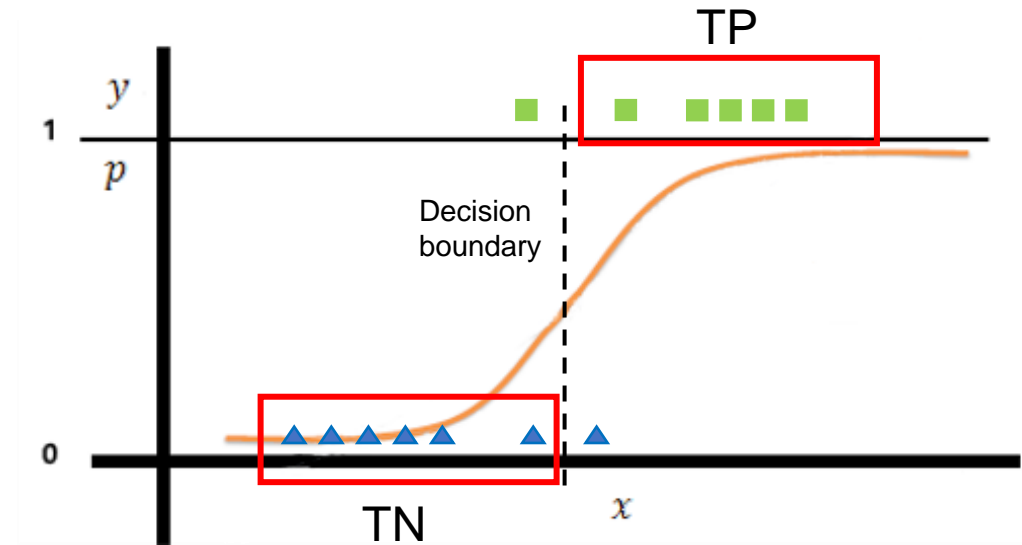
- Accuracy
$$\frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Q: Is accuracy always a good measure?

Be cautious with *skewed dataset* !!!

- Confusion matrix

	Actual Positive (1)	Actual Negative (0)
Predicted Positive (1)	TP	FP
Predicted Negative (0)	FN	TN



$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Performance measure (varying threshold)

- Precision-recall tradeoff

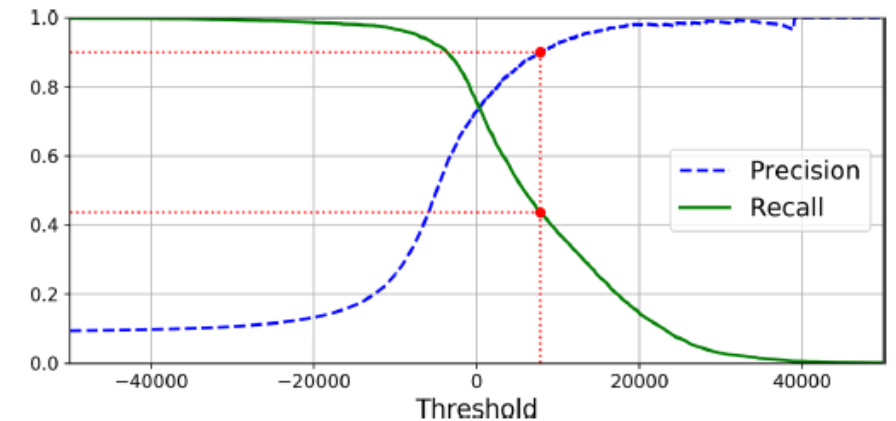
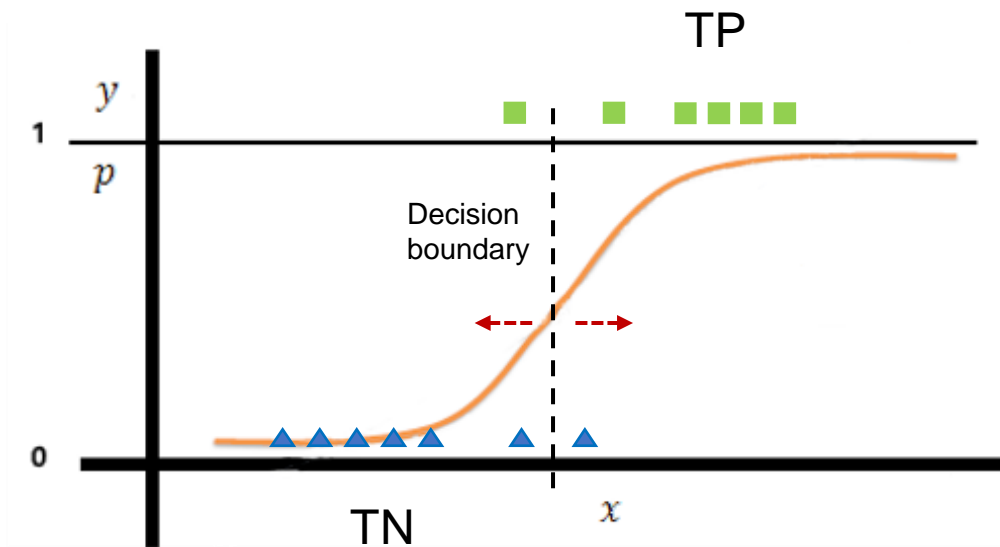


Figure 3-4. Precision and recall versus the decision threshold

Performance measure (varying threshold)

- Precision-recall tradeoff

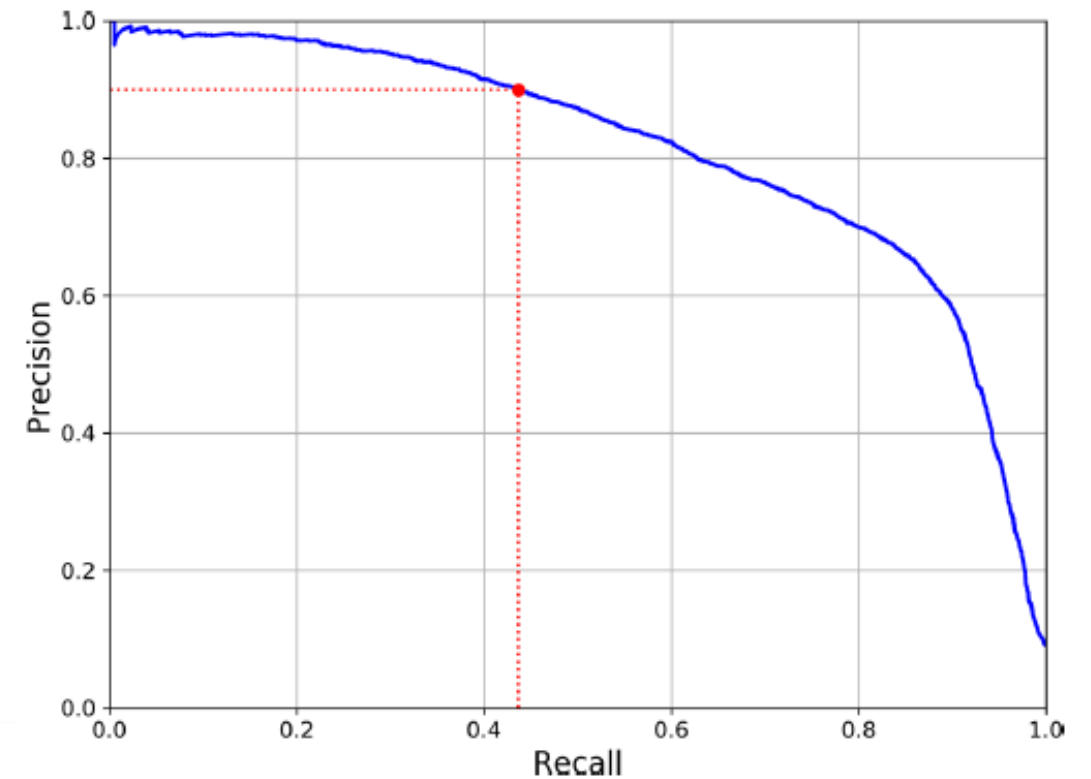
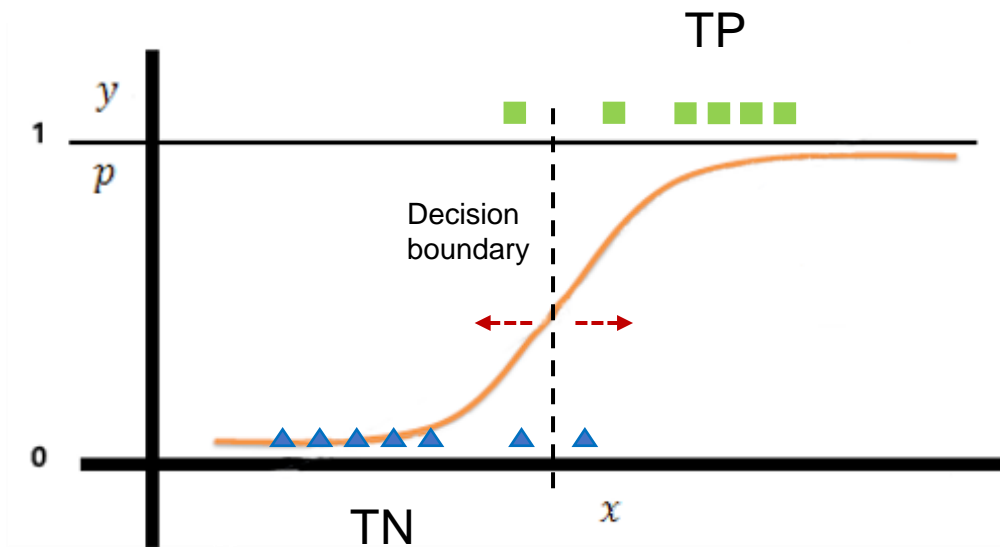


Figure 3-5. Precision versus recall

PR-ROC

Performance measure (varying threshold)

- ROC curve & AUC

- Receiver Operating Characteristic curve
- Area under curve

	Actual Positive (1)	Actual Negative (0)
Predicted Positive (1)	TP	FP
Predicted Negative (0)	FN	TN

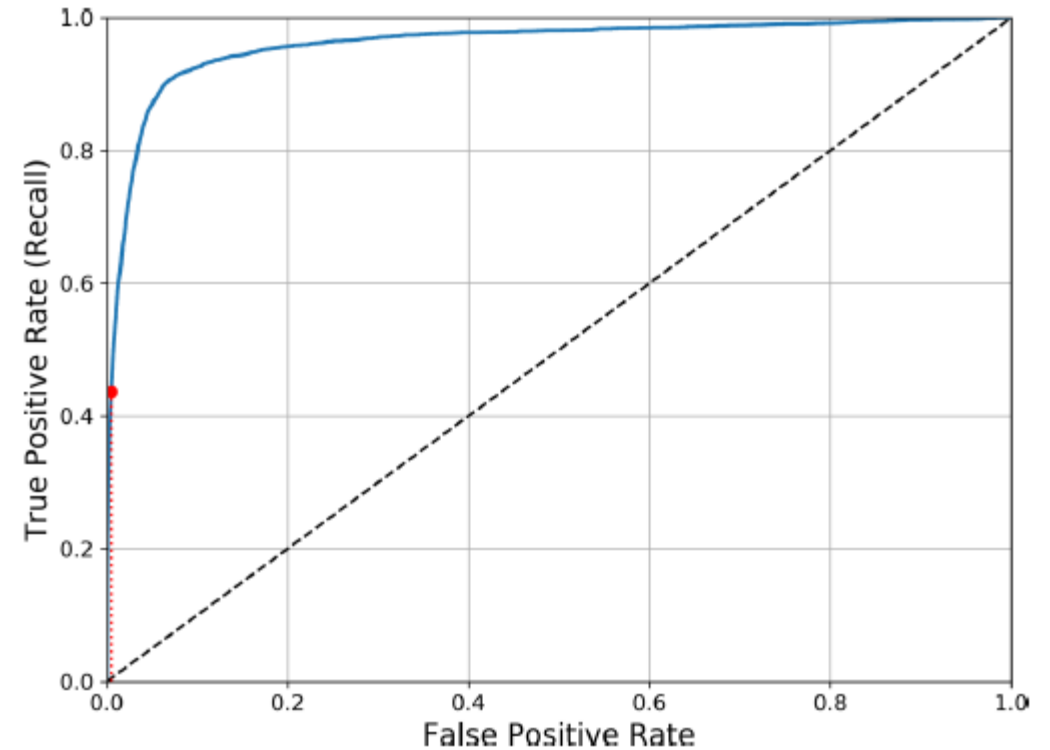


Figure 3-6. This ROC curve plots the false positive rate against the true positive rate for all possible thresholds; the red circle highlights the chosen ratio (at 43.68% recall)

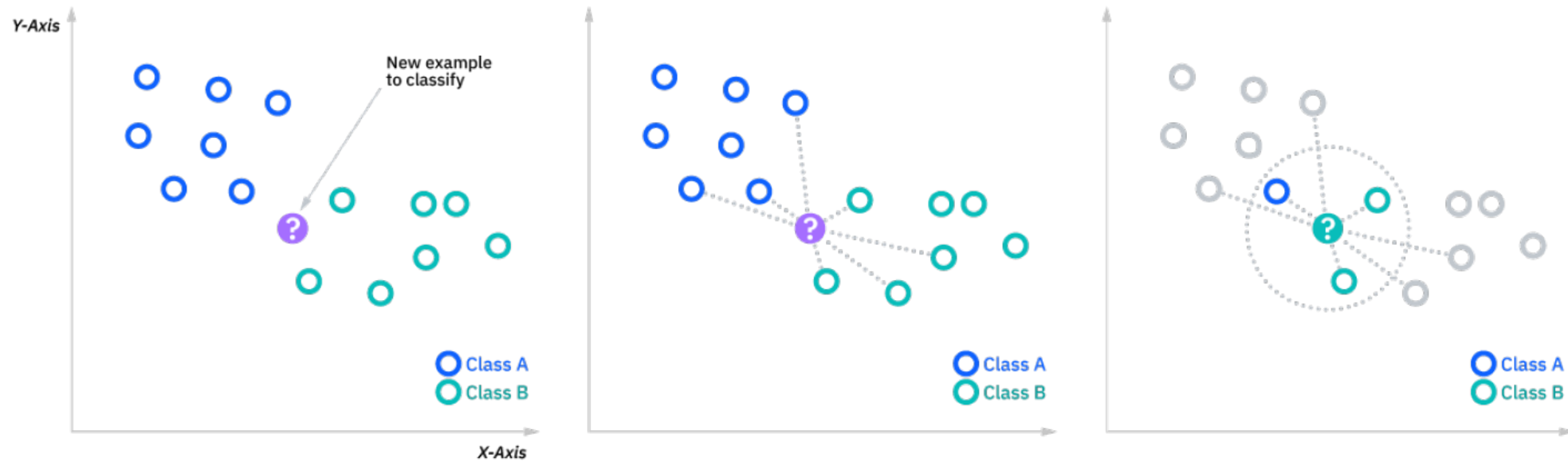
True positive rate (TPR), $TP/(TP+FN)$
False positive rate (FPR), $FP/(FP+TN)$

Summary

- Logistic regression algorithm
- Cross-entropy loss for classification
- Performance measure
 - Accuracy
 - Confusion matrix
 - Precision, recall, and the tradeoff between them
 - ROC, AUC

K-nearest neighbor (KNN)

- An instance based-learning algorithm



KNN

- An instance based-learning algorithm
- Key: distance metrics
 - Euclidean distance (L2 norm)

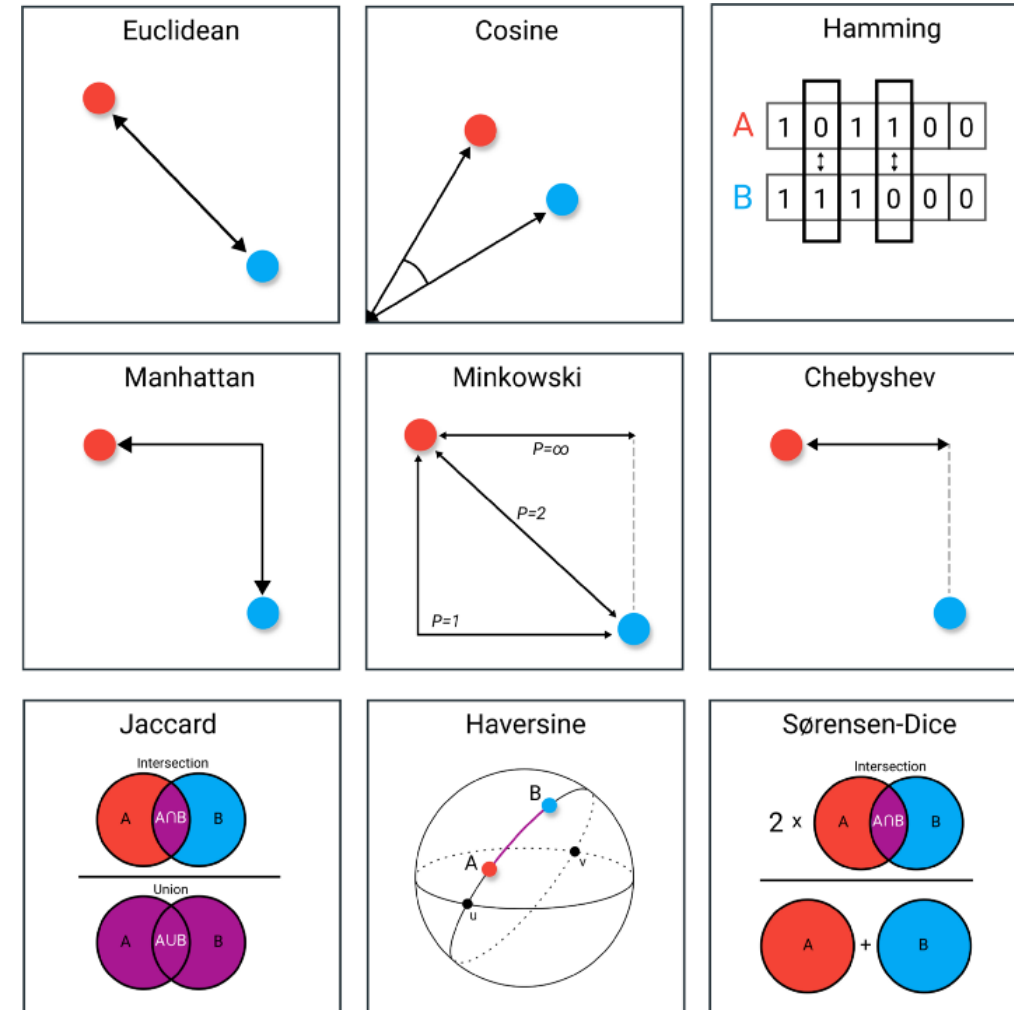
$$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Manhattan distance (L1 norm)

$$D(x, y) = \sum_{i=1}^k |x_i - y_i|$$

- Minkowski distance (Lp norm)

$$D(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$



Naïve bayes

The diagram shows the Naïve Bayes formula with arrows pointing from descriptive labels to the corresponding terms in the equation:

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

Labels and their corresponding terms:

- Likelihood** points to $P(x | c)$
- Class Prior Probability** points to $P(c)$
- Posterior Probability** points to $P(c | x)$
- Predictor Prior Probability** points to $P(x)$

$$P(c | X) \propto P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

Q: Why Naïve Bayes is called naïve?

Naïve bayes

The diagram shows the Naïve Bayes formula with arrows pointing from descriptive labels to the corresponding terms in the equation:

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

Labels and their corresponding terms:

- Likelihood** points to $P(x | c)$
- Class Prior Probability** points to $P(c)$
- Posterior Probability** points to $P(c | x)$
- Predictor Prior Probability** points to $P(x)$

$$P(c | X) \propto P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

Q: Why Naïve Bayes is called naïve?

it makes the assumption that features are *conditionally independent* of each other (condition on class)

Naïve bayes

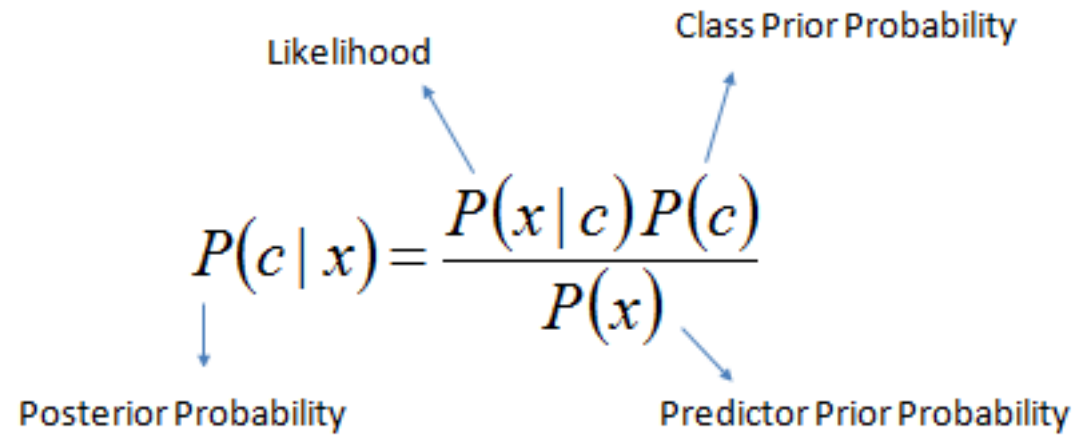


Diagram illustrating the components of Bayes' theorem:

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

Labels and arrows:

- $P(c | x)$ is labeled **Posterior Probability** (arrow pointing down).
- $P(x | c)$ is labeled **Likelihood** (arrow pointing up-left).
- $P(c)$ is labeled **Class Prior Probability** (arrow pointing up-right).
- $P(x)$ is labeled **Predictor Prior Probability** (arrow pointing down-right).

$$P(c | X) \propto P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

- Gaussian Naïve Bayes

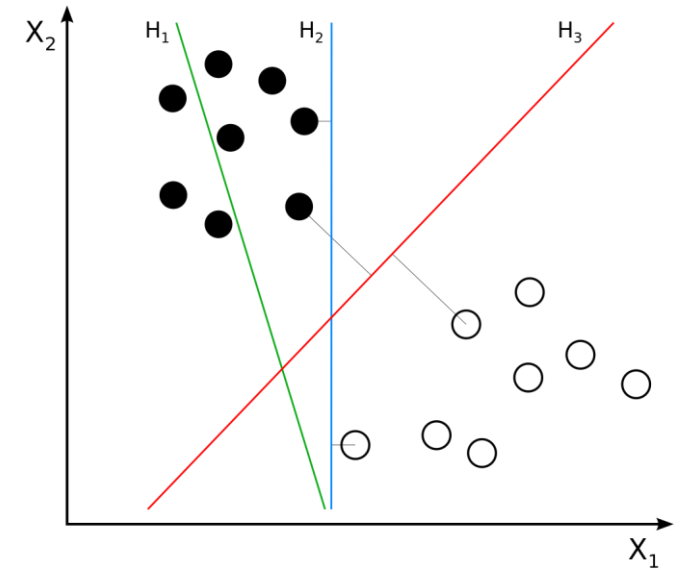
$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Support vector machine

- Decision function

Equation 5-2. Linear SVM classifier prediction

$$\hat{y} = \begin{cases} 0 & \text{if } \mathbf{w}^T \mathbf{x} + b < 0, \\ 1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \end{cases}$$



Question: Which line is the best?

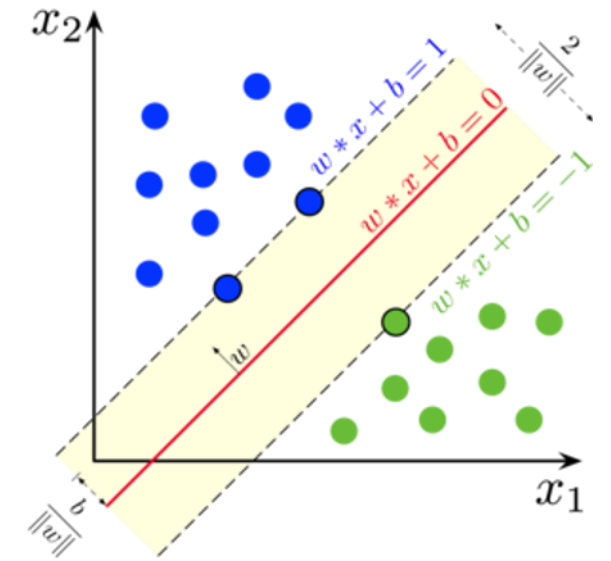
Support vector machine

- Large margin classification

Equation 5-3. Hard margin linear SVM classifier objective

$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w}$$

$$\text{subject to} \quad t^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1, 2, \dots, m$$



- Support vectors

- The **decision boundary** is fully determined (or “supported”) by the instances located on the edge of the street, these instance are called the **support vectors**
- Adding more training instances “off the street” will not affect decision boundary

Support vector machine

- Large margin classification

Equation 5-3. Hard margin linear SVM classifier objective

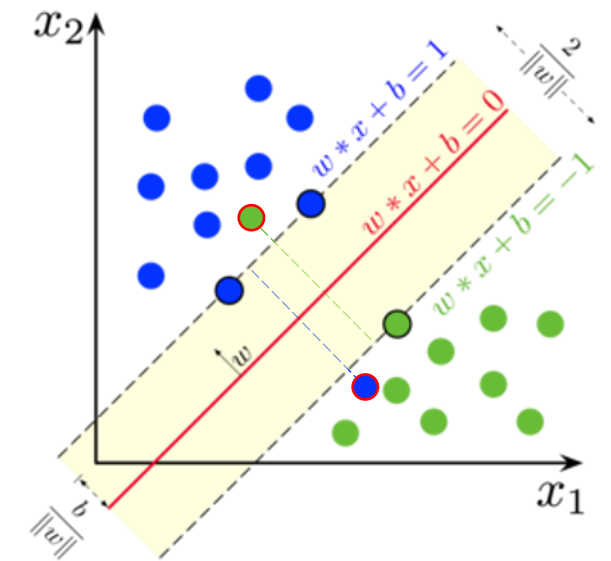
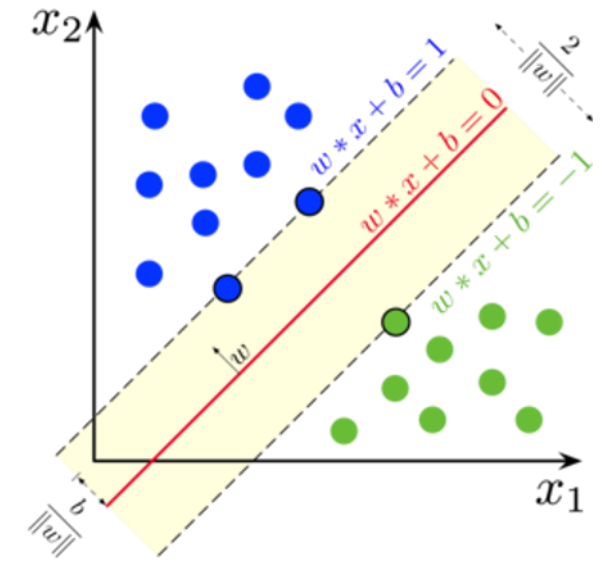
$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w}$$

$$\text{subject to} \quad t^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b \right) \geq 1 \quad \text{for } i = 1, 2, \dots, m$$

- Soft-margin classification

$$\underset{\mathbf{w}, b, \zeta}{\text{minimize}} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^m \zeta^{(i)}$$

$$\text{subject to} \quad t^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b \right) \geq 1 - \zeta^{(i)} \quad \text{and} \quad \zeta^{(i)} \geq 0 \quad \text{for } i = 1, 2, \dots, m$$



Support vector machine (nonlinear)

- With polynomial kernel

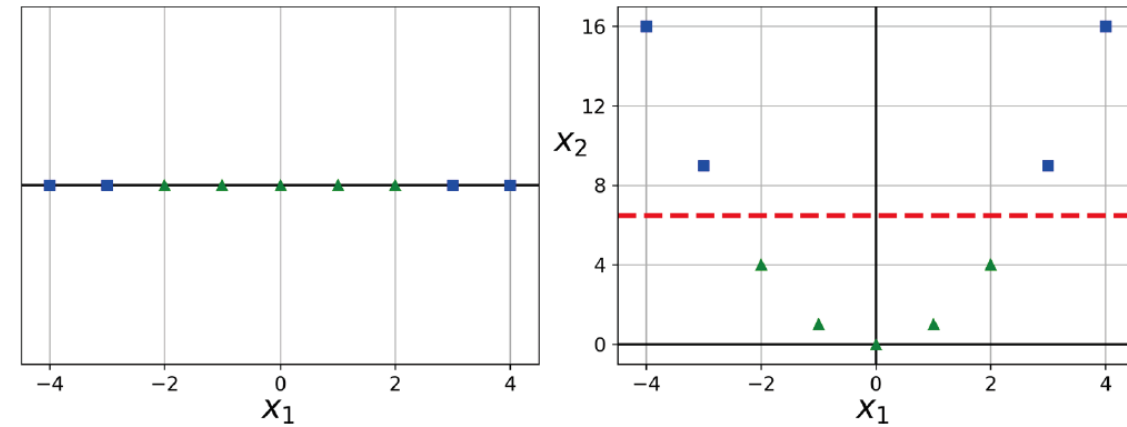


Figure 5-5. Adding features to make a dataset linearly separable

- With similarity measure
 - Gaussian *Radial Basis Function* (RBF)

$$\phi_{\gamma}(\mathbf{x}, \ell) = \exp\left(-\gamma\|\mathbf{x} - \ell\|^2\right)$$

ℓ : a particular landmark

γ : a hyperparameter

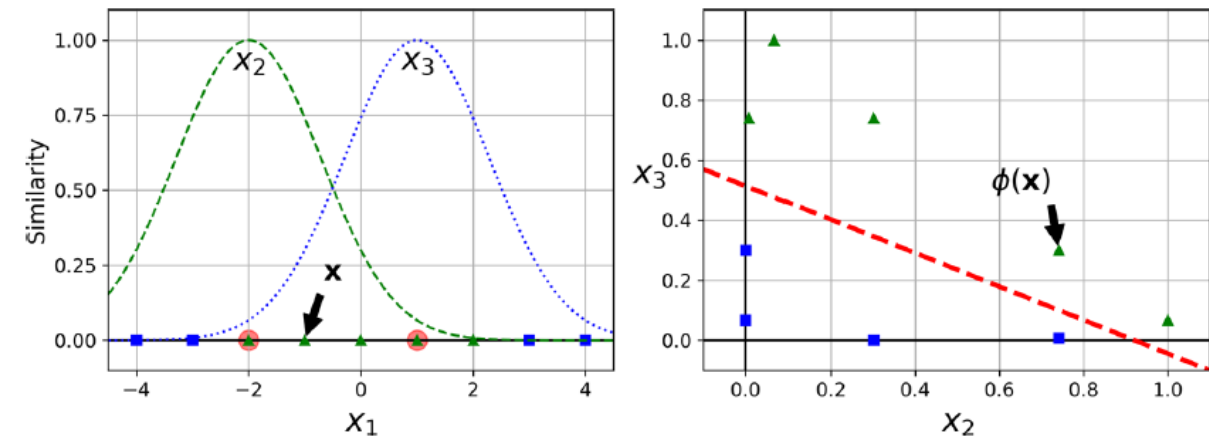


Figure 5-8. Similarity features using the Gaussian RBF

Decision trees

- A white-box algorithm, which is intuitive and its decision is easy to interpret

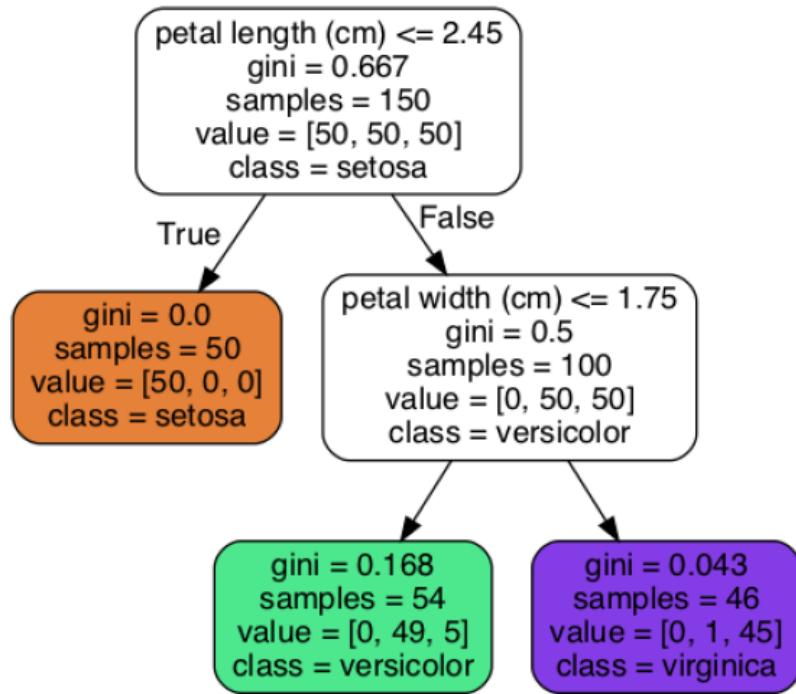
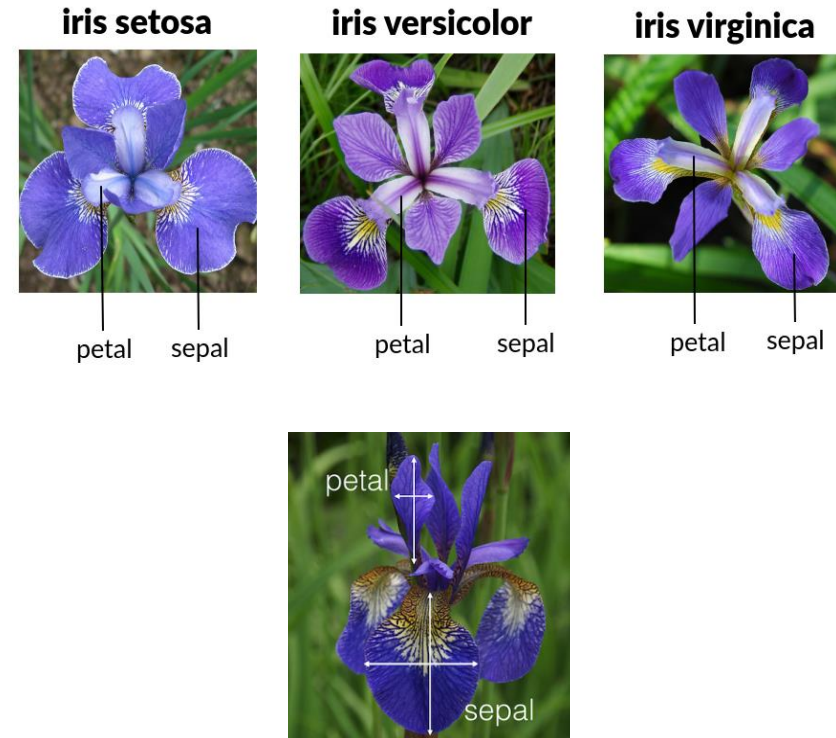


Figure 6-1. Iris Decision Tree



Decision trees

- A white-box algorithm, which is intuitive and its decision is easy to interpret

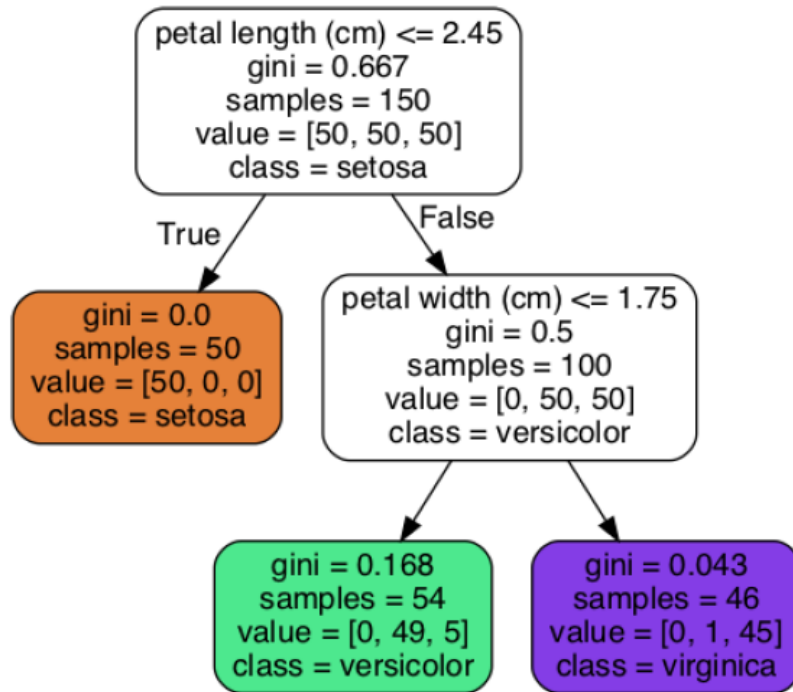


Figure 6-1. Iris Decision Tree

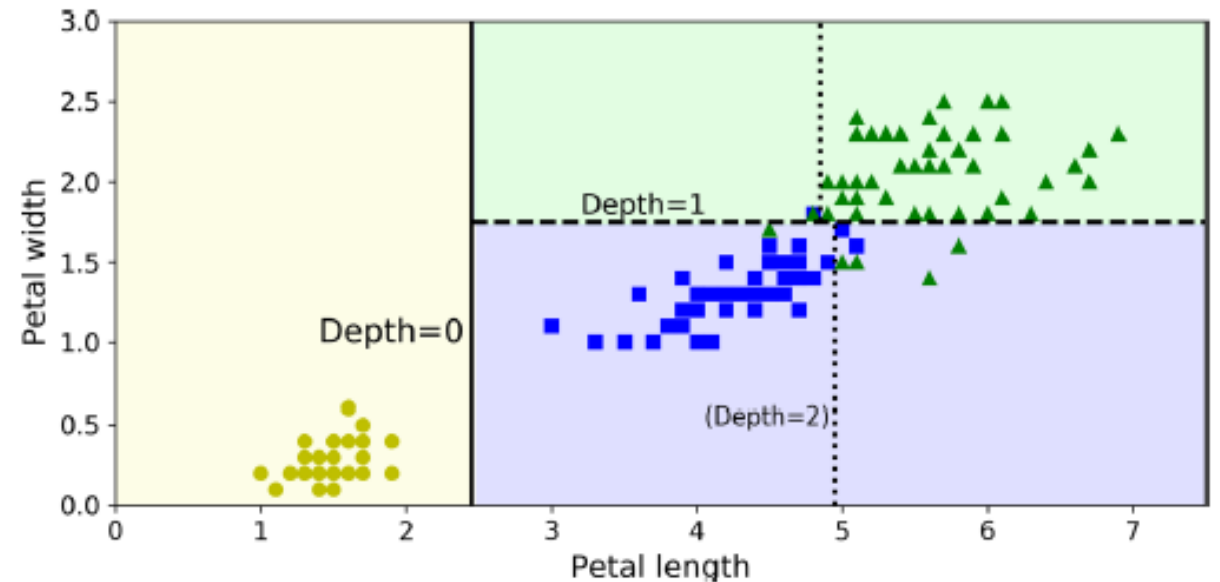


Figure 6-2. Decision Tree decision boundaries

Decision trees

- A white-box algorithm, which is intuitive and its decision is easy to interpret

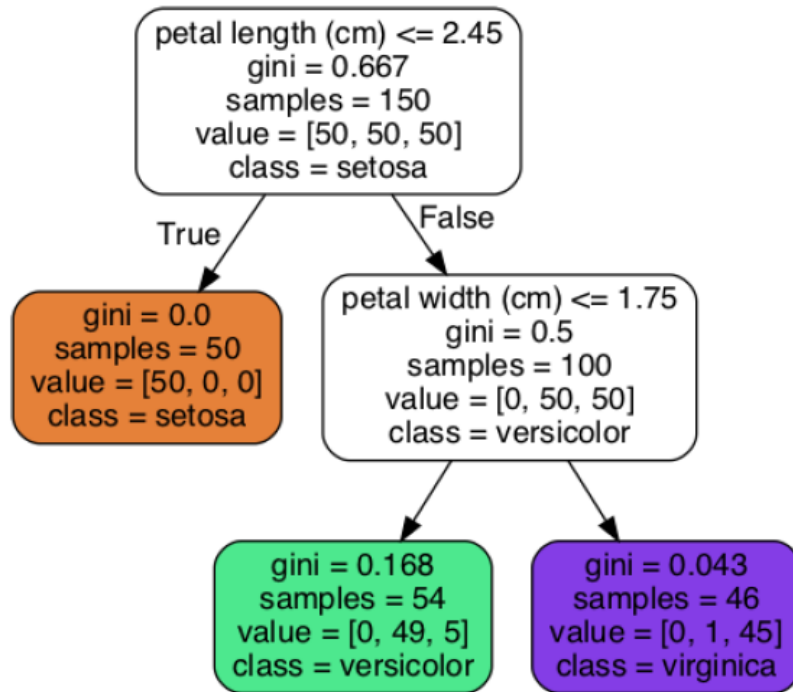


Figure 6-1. Iris Decision Tree

- The algorithm search for a feature k and threshold t that produce a purest subset

Equation 6-2. CART cost function for classification

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} & \text{measures the impurity of the left/right subset,} \\ m_{\text{left/right}} & \text{is the number of instances in the left/right subset.} \end{cases}$

Decision trees

- A white-box algorithm, which is intuitive and its decision is easy to interpret

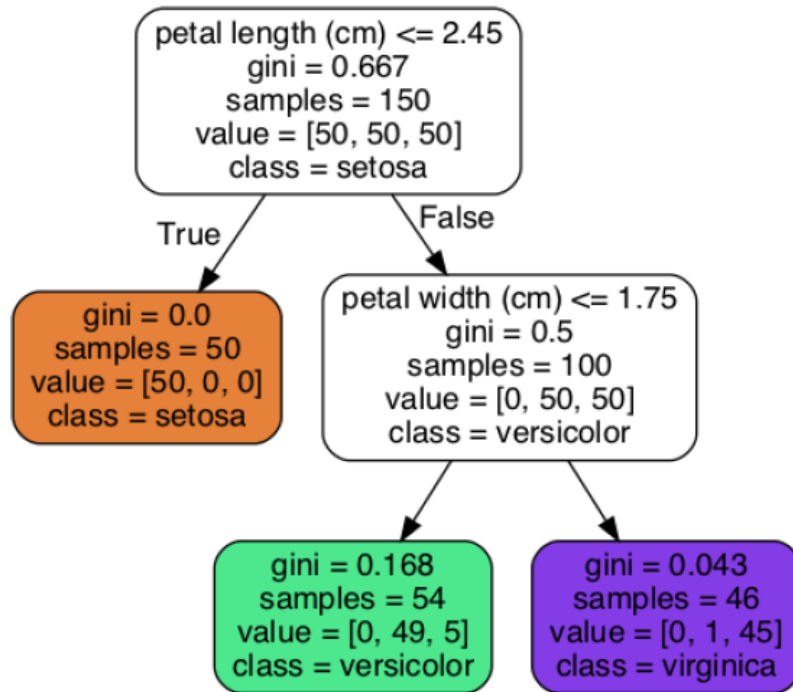


Figure 6-1. Iris Decision Tree

- The algorithm search for a feature k and threshold t that produce a purest subset

Equation 6-2. CART cost function for classification

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} & \text{measures the impurity of the left/right subset,} \\ m_{\text{left/right}} & \text{is the number of instances in the left/right subset.} \end{cases}$

- Purity measure: Gini index

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

($p_{i,k}$ is the ratio of class k instance in node i)

Decision trees limitation

- Instability: sensitive to small variation in training data

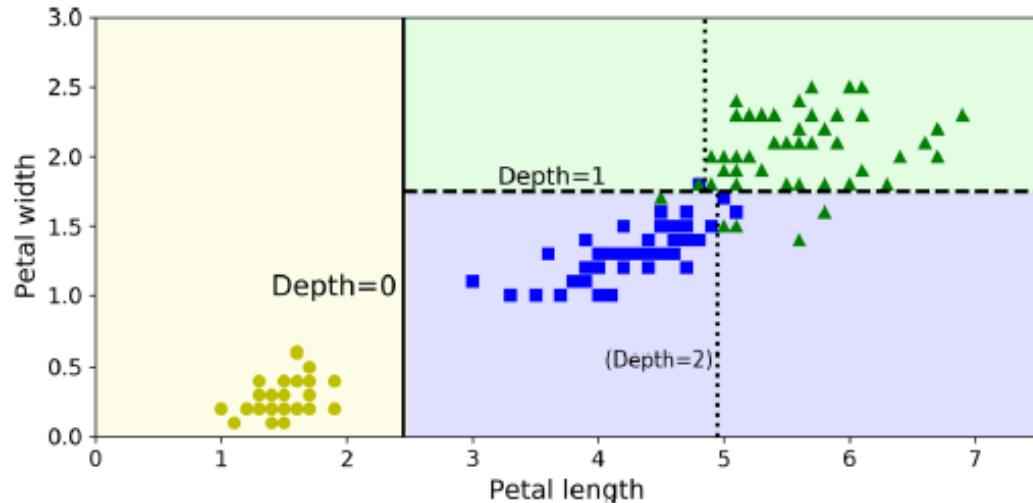


Figure 6-2. Decision Tree decision boundaries

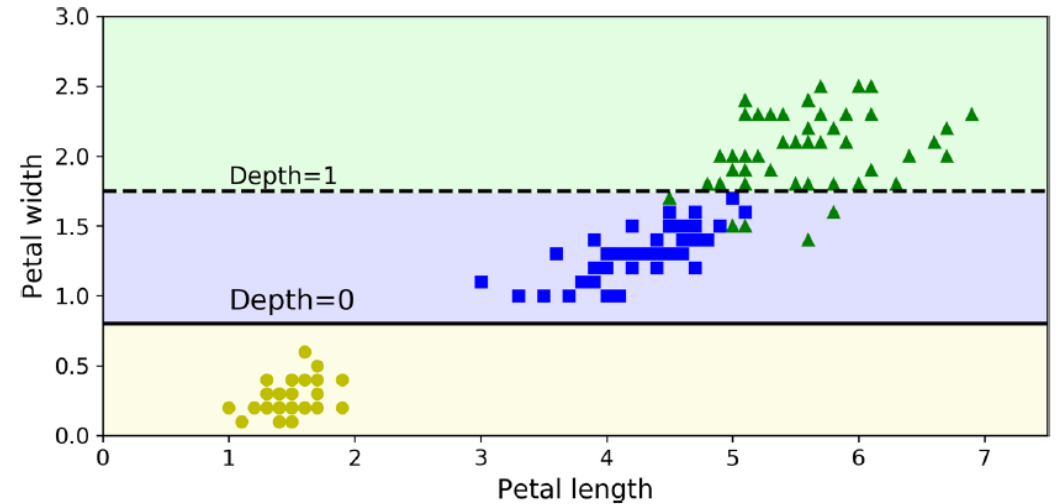


Figure 6-8. Sensitivity to training set details

Ensemble learning – the wisdom of the crowd

Even if each classifier is a *weak learner* (meaning it does only slightly better than random guessing), the ensemble can still be a *strong learner* (achieving high accuracy), provided there are a sufficient number of weak learners and they are *sufficiently diverse*

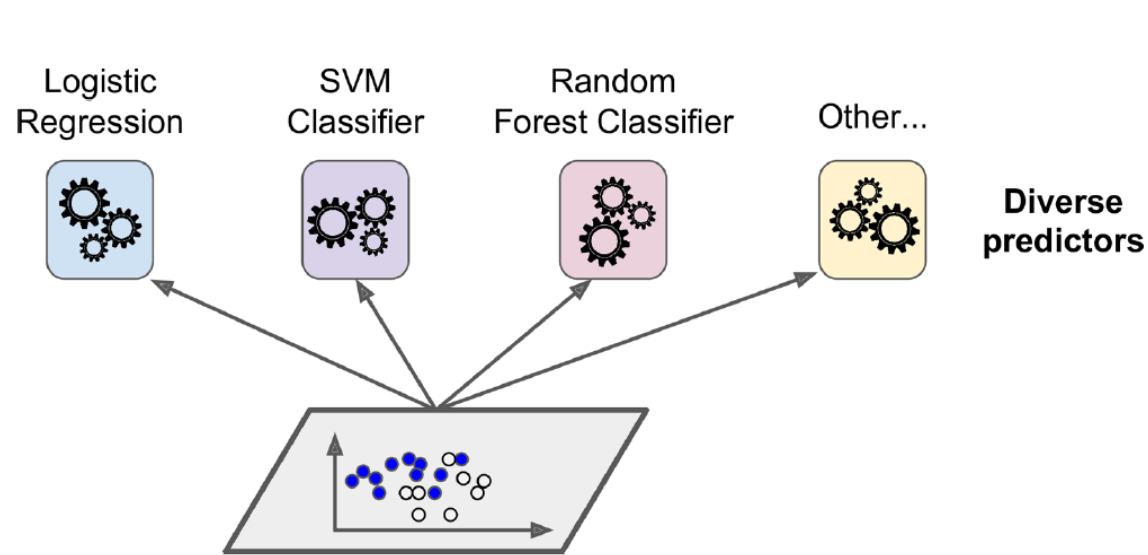


Figure 7-1. Training diverse classifiers

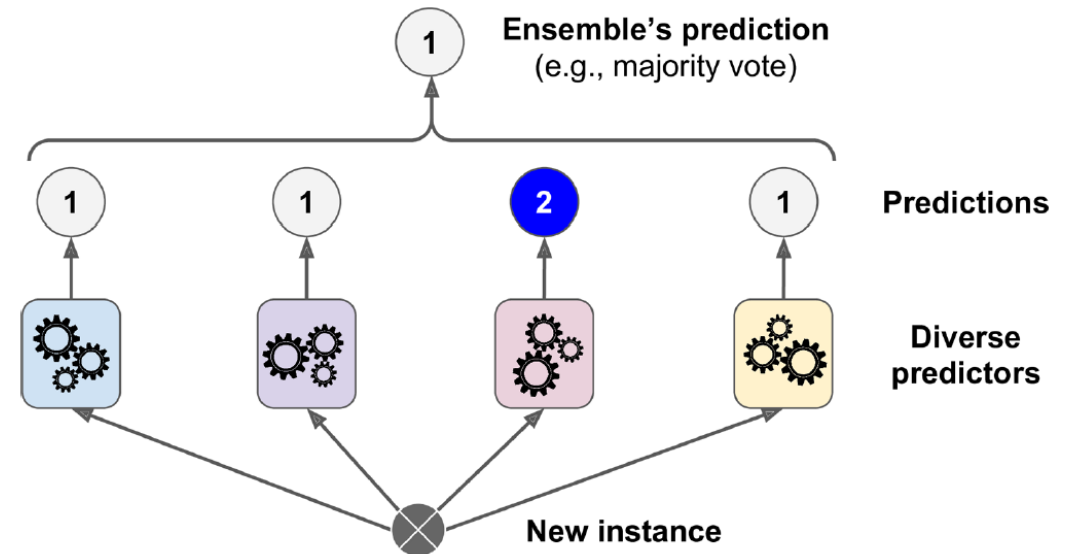
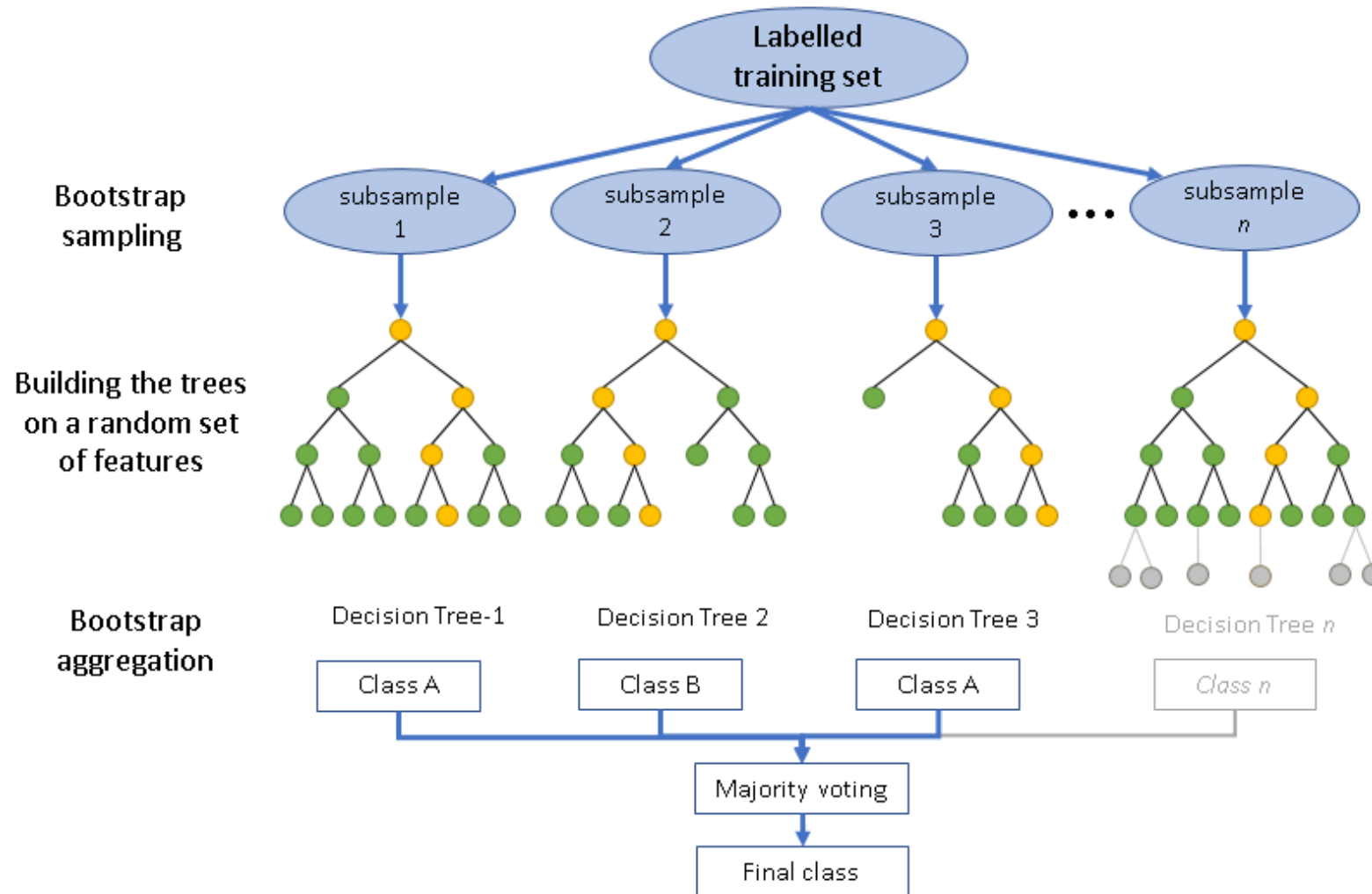


Figure 7-2. Hard voting classifier predictions

Usually ensemble method can achieve better prediction than the best individual predictor

Random forest

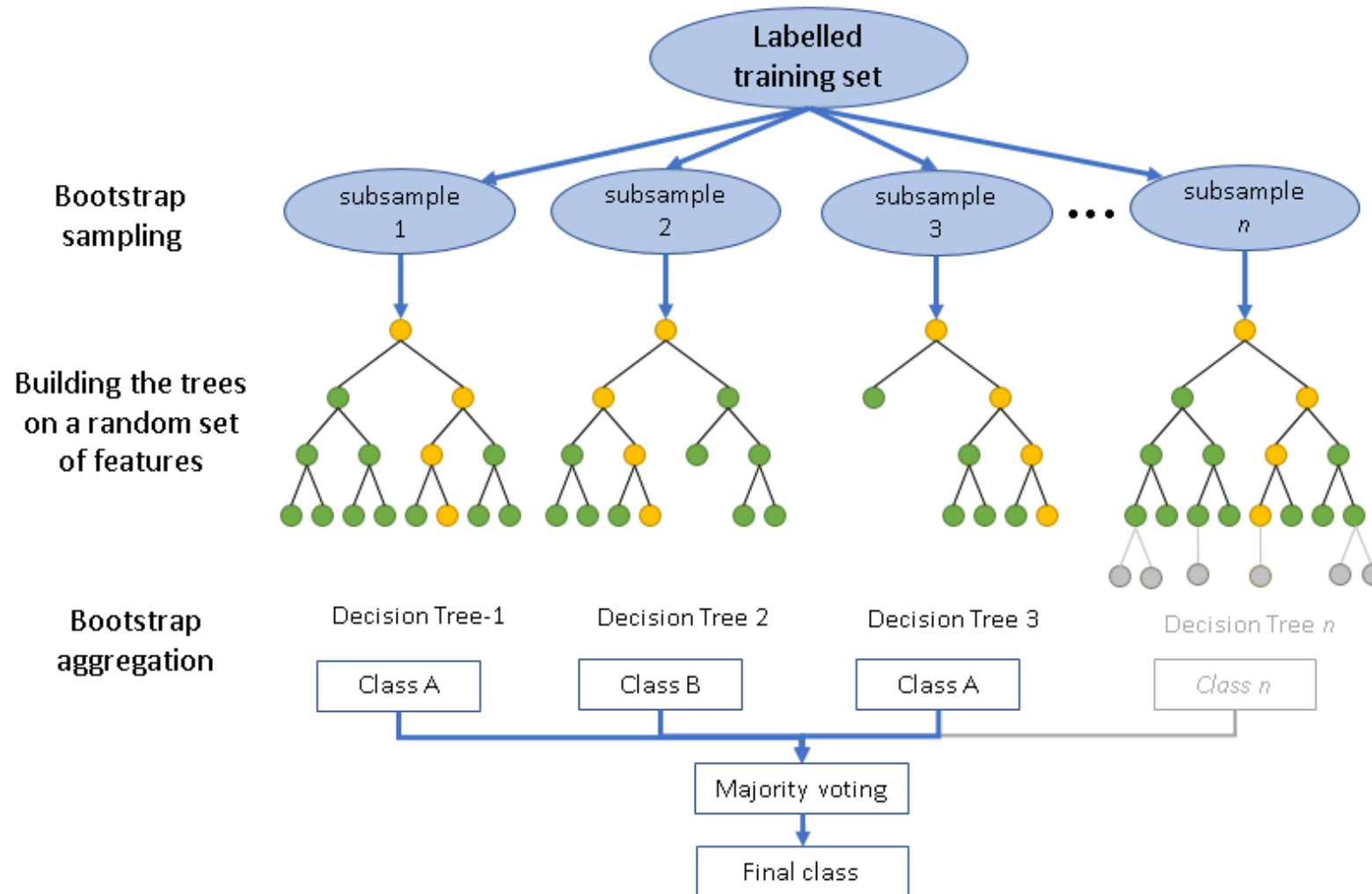
- An ensemble of decision trees



Reduce the variance of a single tree

Random forest

- An ensemble of decision trees

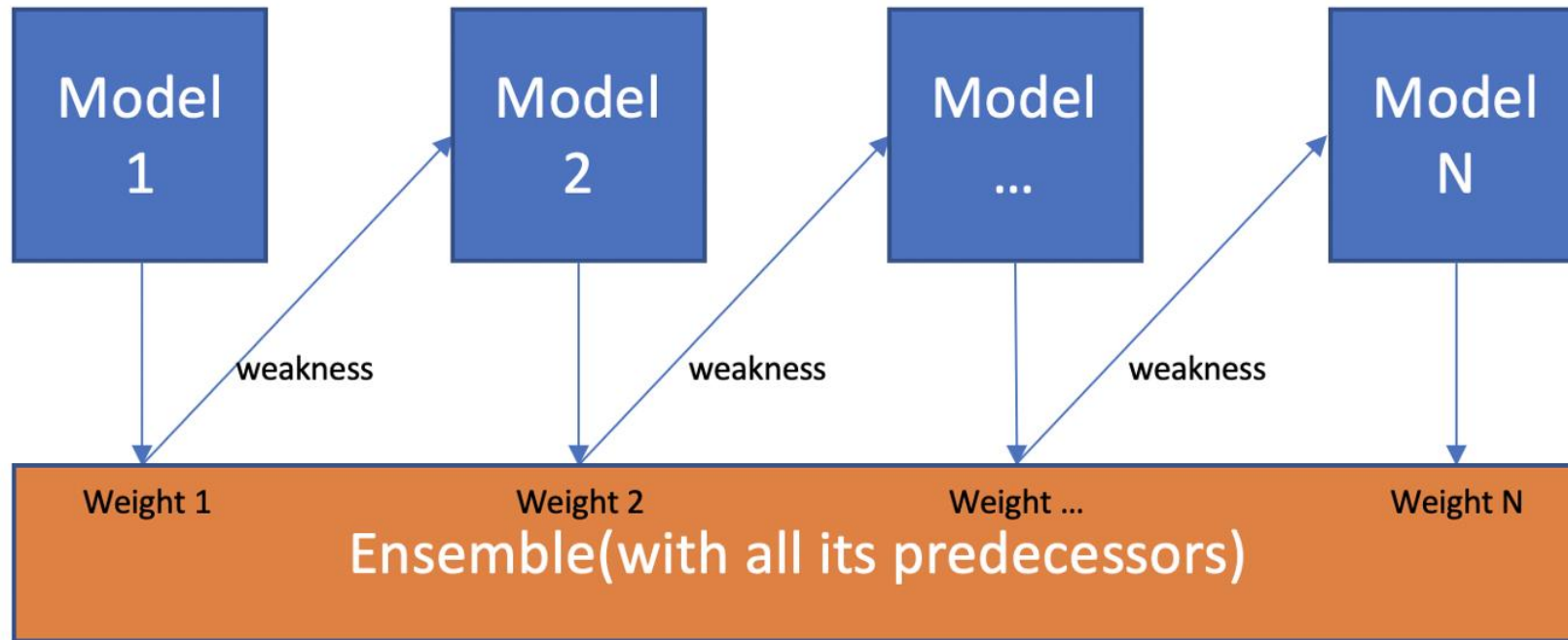


Feature importance:
how much the tree nodes that use that feature reduce impurity on average (across all trees in the forest)

Boosting method:

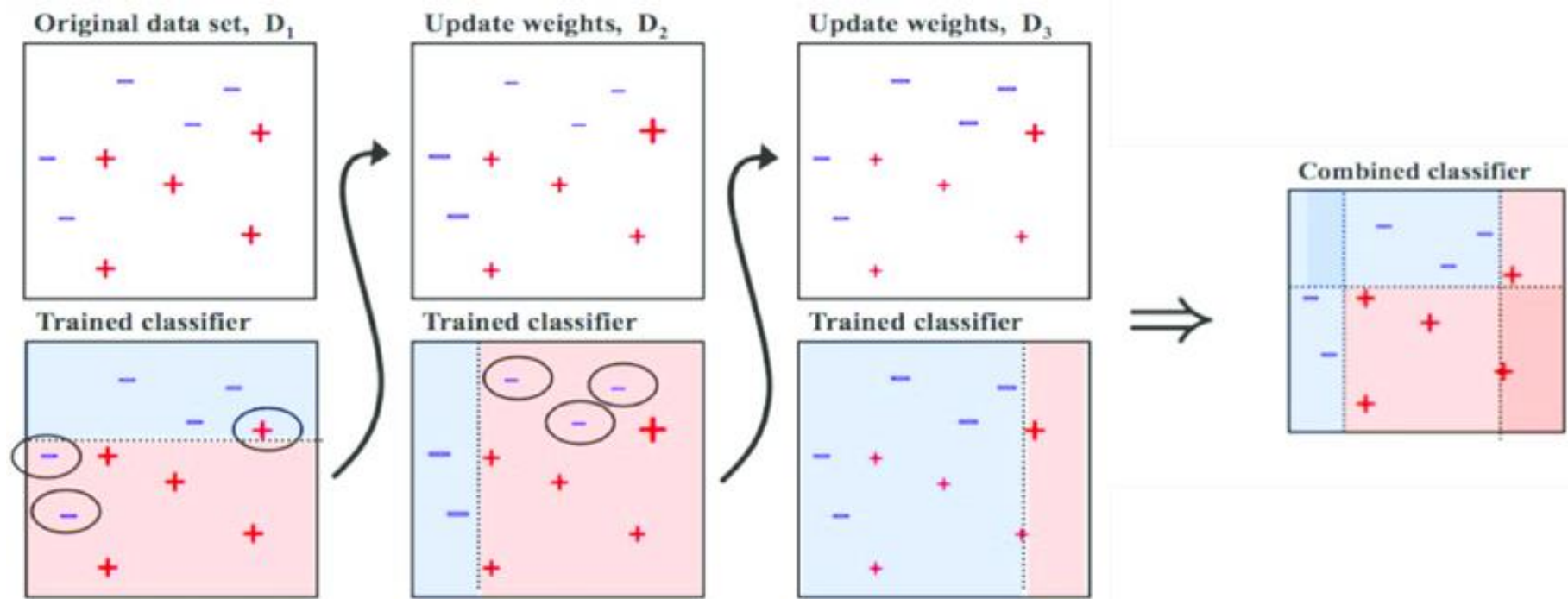
- Combine weak learner to form a strong learner
- Train predictors **sequentially**, each trying to correct its predecessor

Model 1,2,..., N are individual models (e.g. decision tree)



Adaboost

- Pay a bit more attention to the training instances that the predecessor underfitted



Adaboost

- Core algorithm:

- Initialize with equal weights for each instance i

- Compute weighted error rate for j^{th} predictor

- Compute the predictor's weight

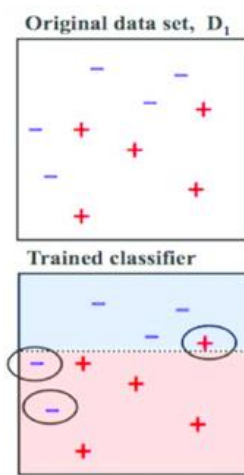
- Weight update for instance

- Repeat until designed number of predictors is reached or perfect predictors is found

$$r_j = \frac{\sum_{i=1}^m w^{(i)} \mathbb{1}_{\hat{y}_j^{(i)} \neq y^{(i)}}}{\sum_{i=1}^m w^{(i)}}$$
$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

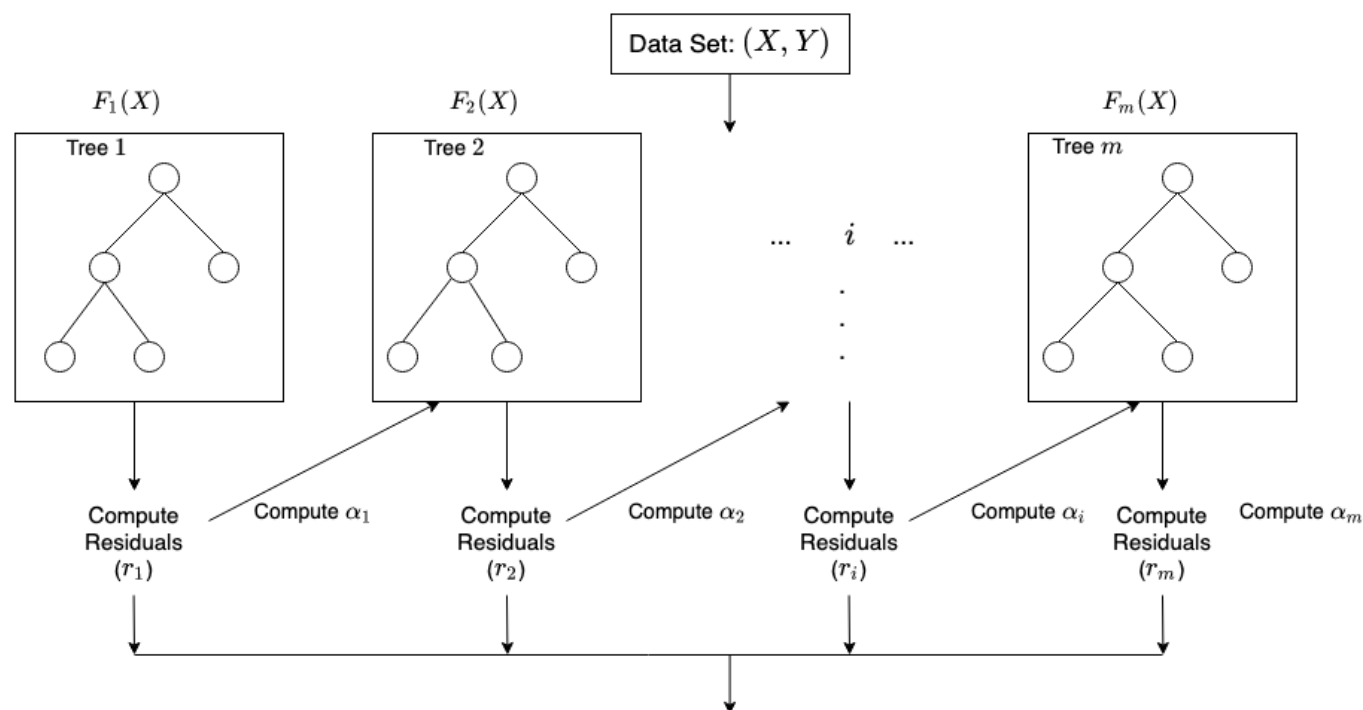
$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} \exp(\alpha_j) & \text{if } \hat{y}_j^{(i)} \neq y^{(i)} \end{cases}$$

- To make predictions, computes the predictions of all the predictors and calculate weighted average



Gradient boosting

- Fit a new predictor to the residual errors made by the previous predictor



$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

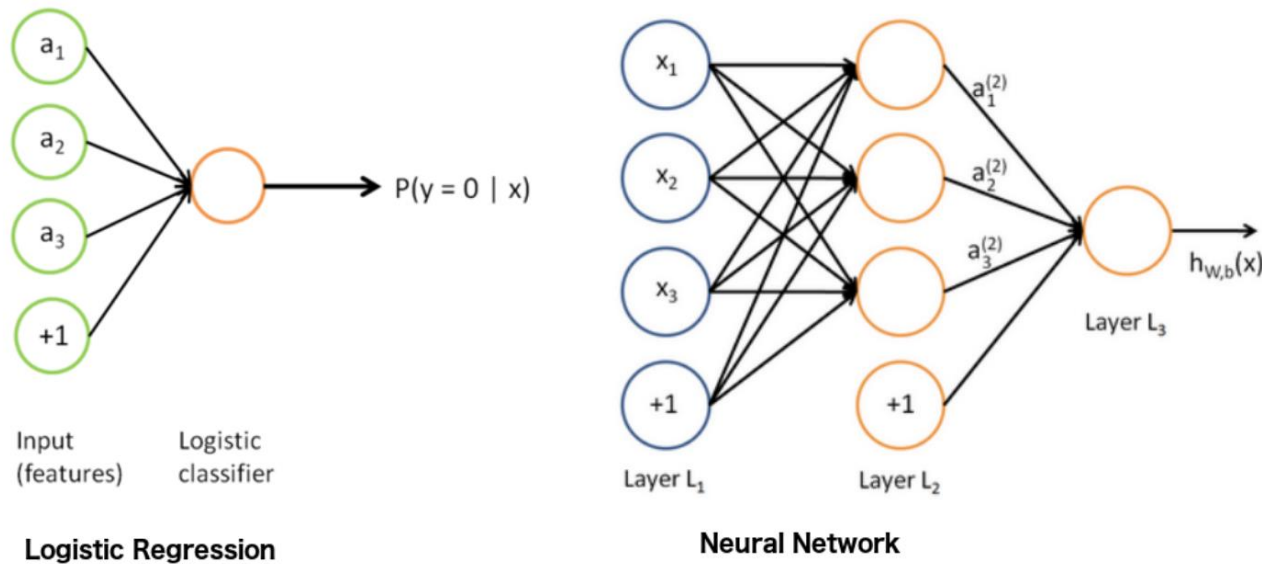
where α_i , and r_i are the regularization parameters and residuals computed with the i^{th} tree respectively, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals

computed, r_i and compute the following: $\arg \min_{\alpha} = \sum_{i=1}^m L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where

$L(Y, F(X))$ is a differentiable loss function.

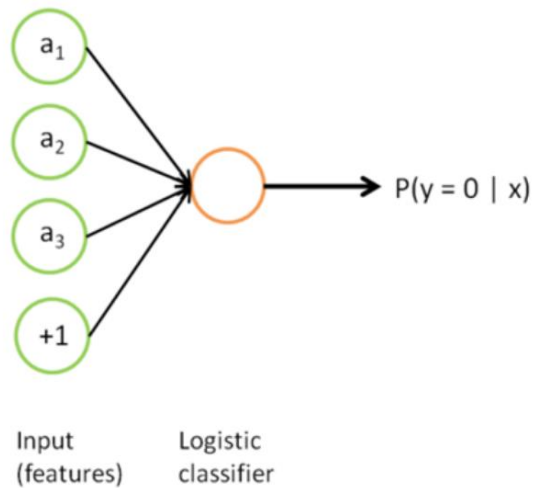
Neural network

- Logistic regression can be regarded as a single layer of Neural network with sigmoid activation function

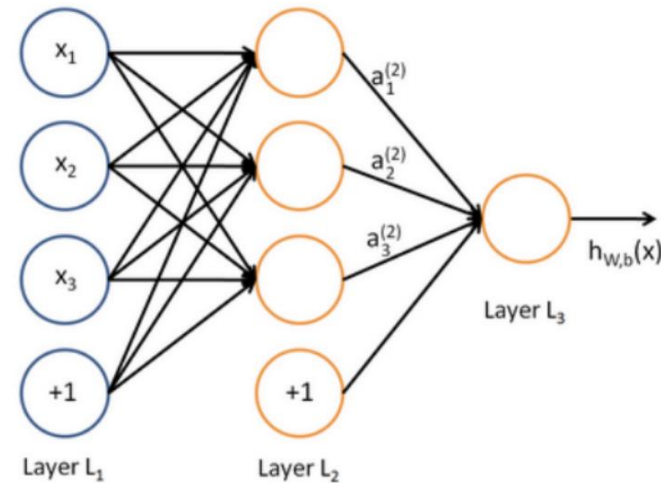


Neural network

- Logistic regression can be regarded as a single layer of Neural network with sigmoid activation function



Logistic Regression



Neural Network

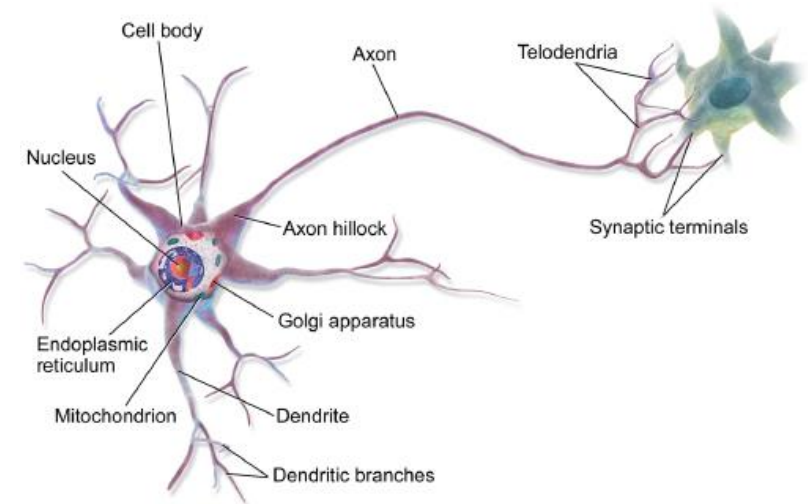
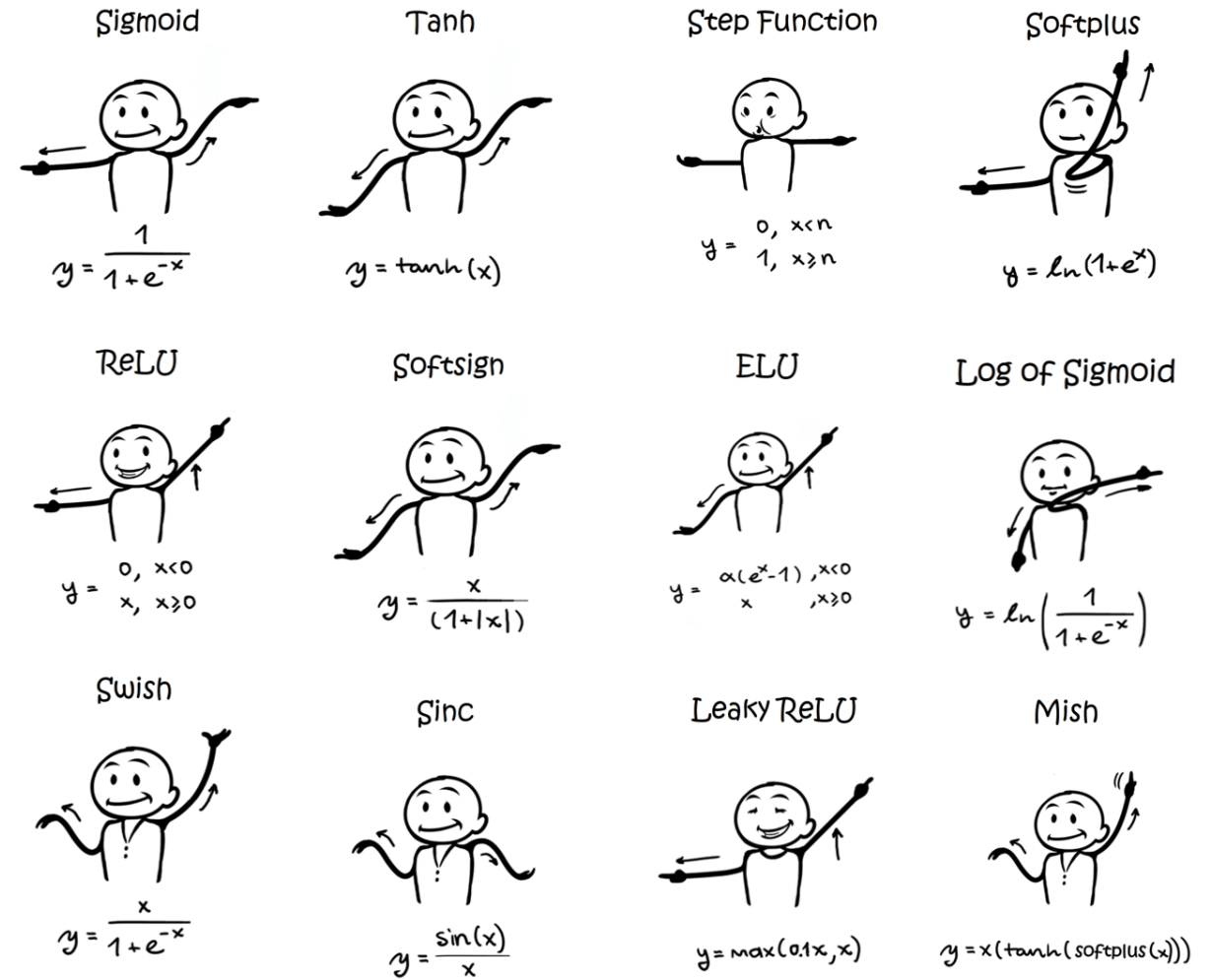


Figure 10-1. Biological neuron⁴

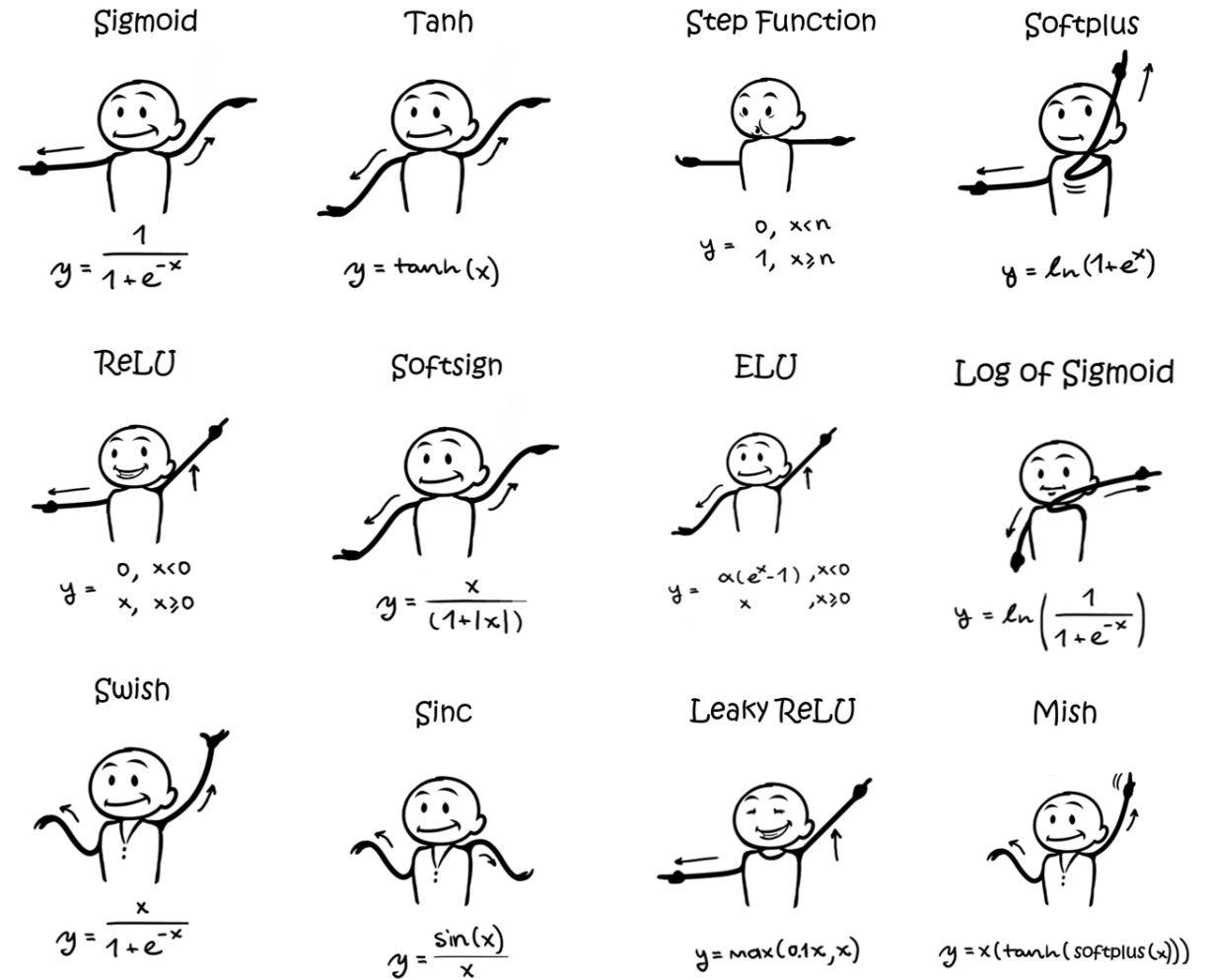
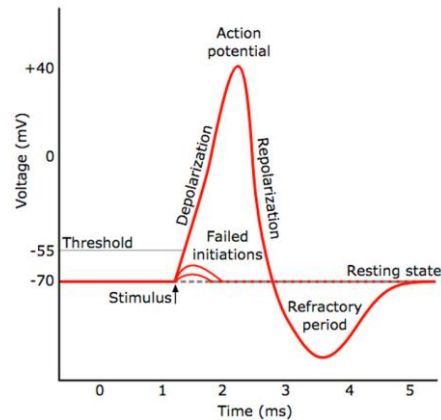
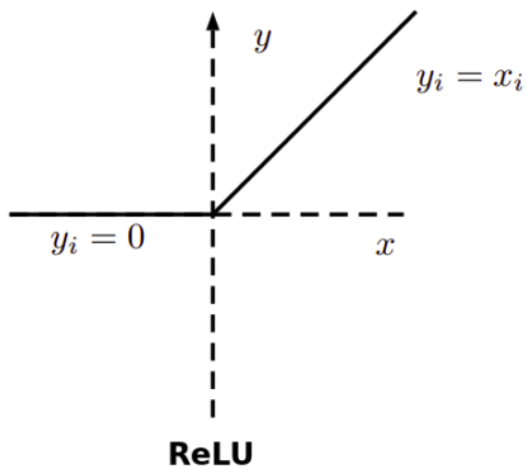
Neural network

- Activation function
 - What if there is no activation function?

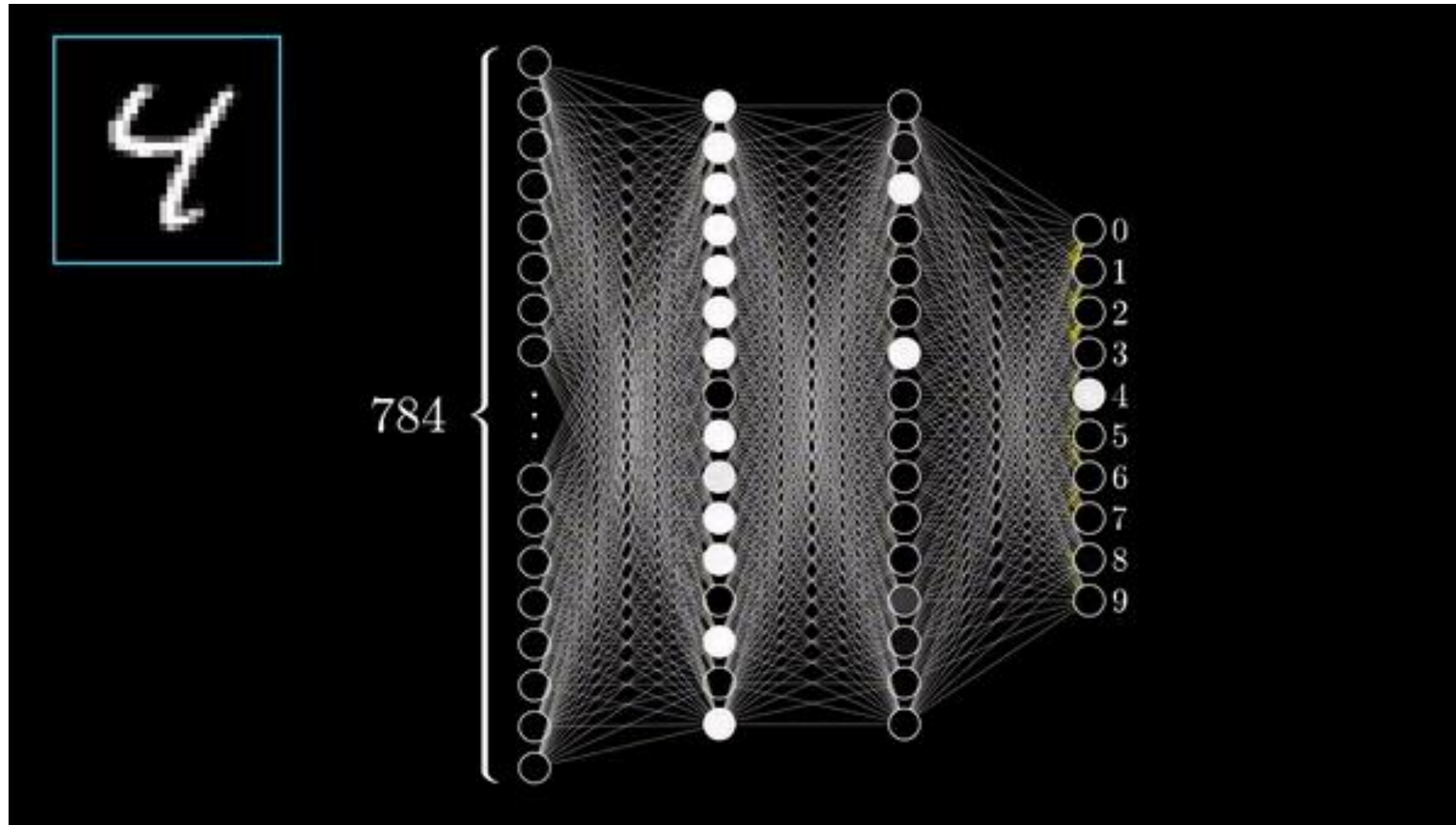


Neural network

- Activation function
 - ReLU is a good default



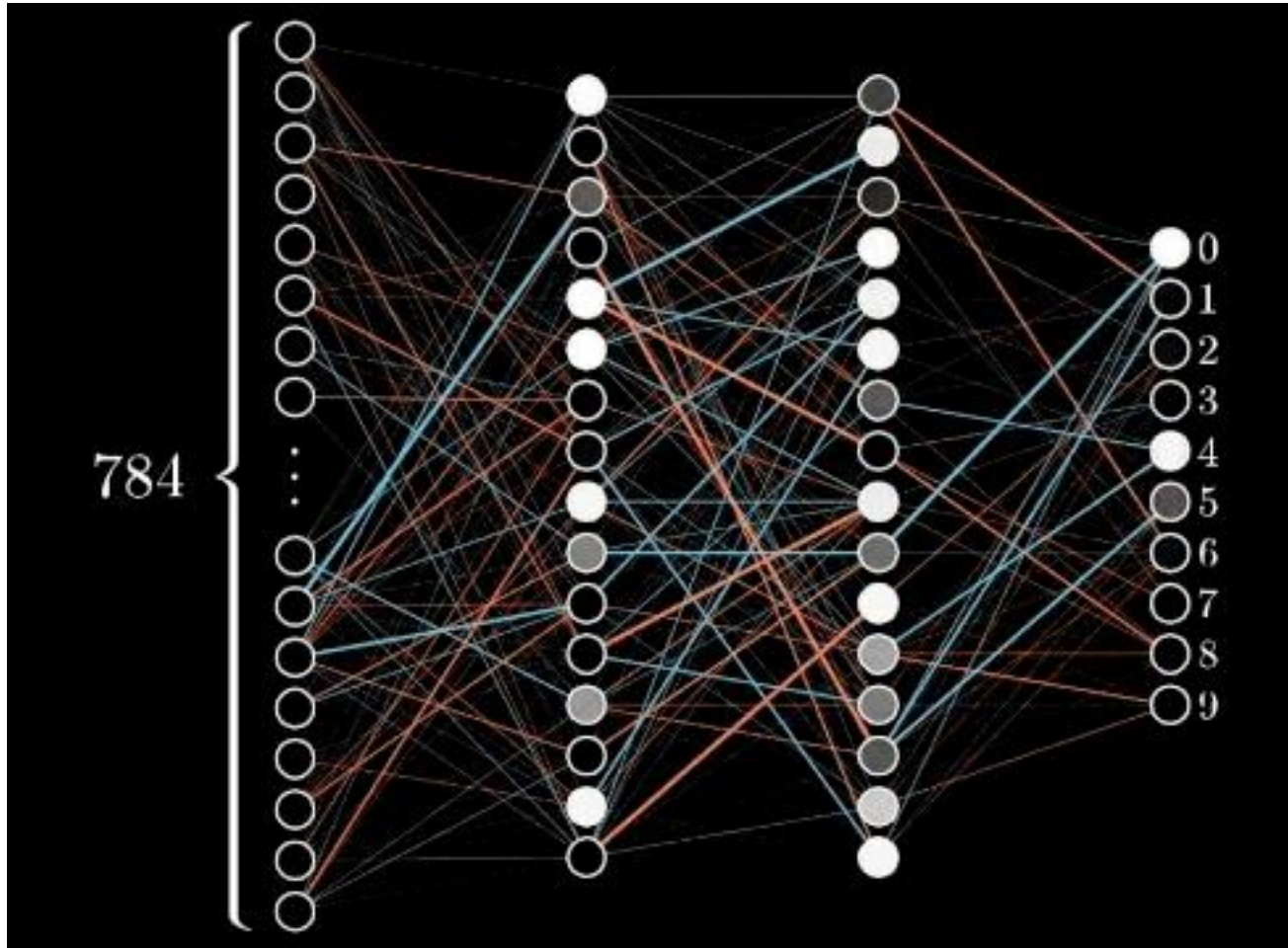
Neural network example (forward propagation)



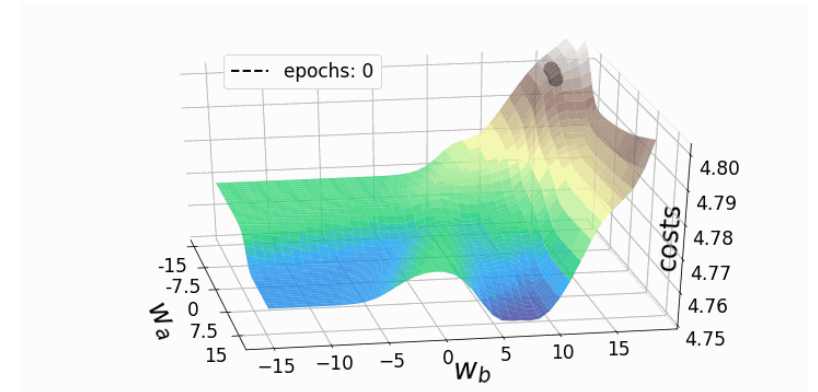
Cross Entropy Loss:

$$L(\Theta) = - \sum_{i=1}^k y_i \log(\hat{y}_i)$$

Neural network example (back propagation)



$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

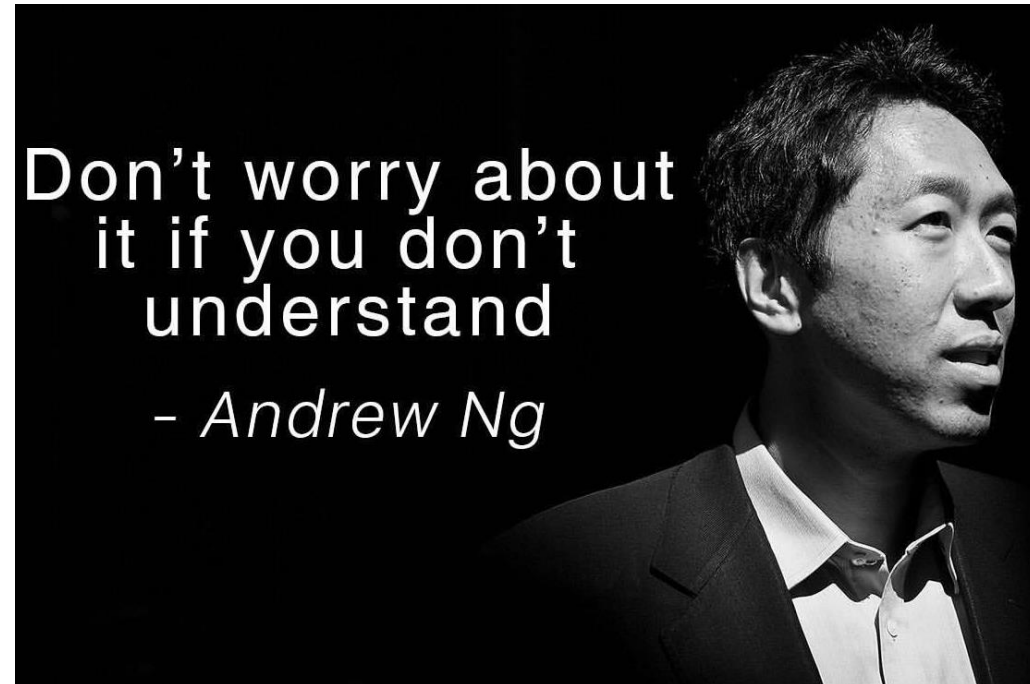


Summary

Algorithm	Time complexity		Space complexity	Advantage	Limitations
	Training	Testing			
Logistic Regression	$O(n*d)$	$O(d)$	$O(d)$	Simple, easy to implement and interpret Require less computation Can update easily using SGD	<ul style="list-style-type: none"> • Need regularization in high dimensional data • Cannot handle non-linear problem • Sensitive to outliers
KNN	$O(k*n*d)$	$O(n*d)$	$O(n*d)$	<ul style="list-style-type: none"> • Intuitive, easy to implement • Less assumption restriction • No pre-training is needed 	<ul style="list-style-type: none"> • Slow speed with big dataset • Don't perform well in imbalanced dataset or high dimensional data • May be sensitive to outliers
Naïve bayes	$O(n*d)$	$O(d)$	$O(c*d)$	• Require less computation	• Independence assumption may not hold
SVM	$O(n^2)$	$O(n'*d)$	$O(n*d)$	<ul style="list-style-type: none"> • Can handle non-linear problem by kernel tricks • Generalize well in practice 	• Don't scale up easily
Decision tree	$O(n*\log(n)*d)$	$O(d)$	$O(td)$	<ul style="list-style-type: none"> • Easy to interpret, white box • Require little data preparation • Scale well to large datasets 	<ul style="list-style-type: none"> • Not robust to small variation • May overfit
Random forest	$O(k*n*\log(n)*d)$	$O(k*d)$	$O(k*td)$	<ul style="list-style-type: none"> • Reduce overfitting, improve accuracy • Built-in feature importance 	<ul style="list-style-type: none"> • Blackbox • A little longer training time

(n: # samples; d: # features; c: # classes; n': # support vectors; k: # trees/neighbors; td: tree depth;)

Let's do some practice!



Machine learning be like

➤ `git clone https://github.com/wbvguo/qcbio-ML_w_Python.git`

Q&A