UCLA QCBio Collaboratory

Machine Learning with Python

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu

2023 Spring

UCLA Bioinformatics

Thanks Dr. Seyoon Ko for some code materials

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

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

Day 2: Supervised learning

Wenbin Guo

Bioinformatics IDP, UCLA

wbguo@ucla.edu

2023 Spring

Overview

Time

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

Topics

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

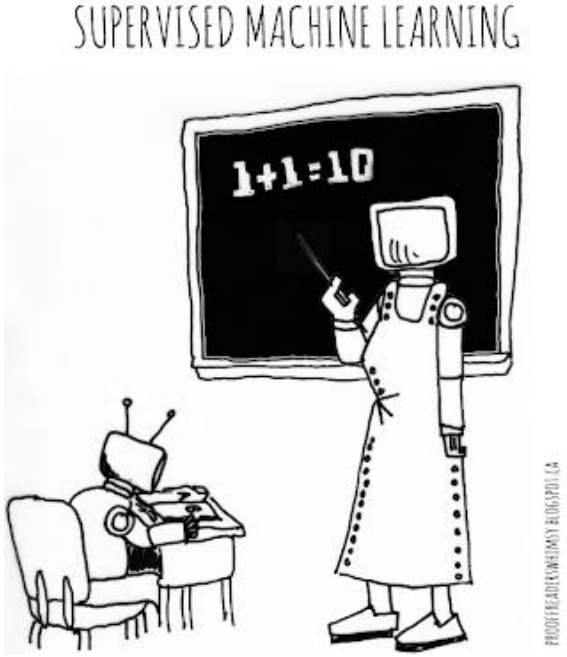
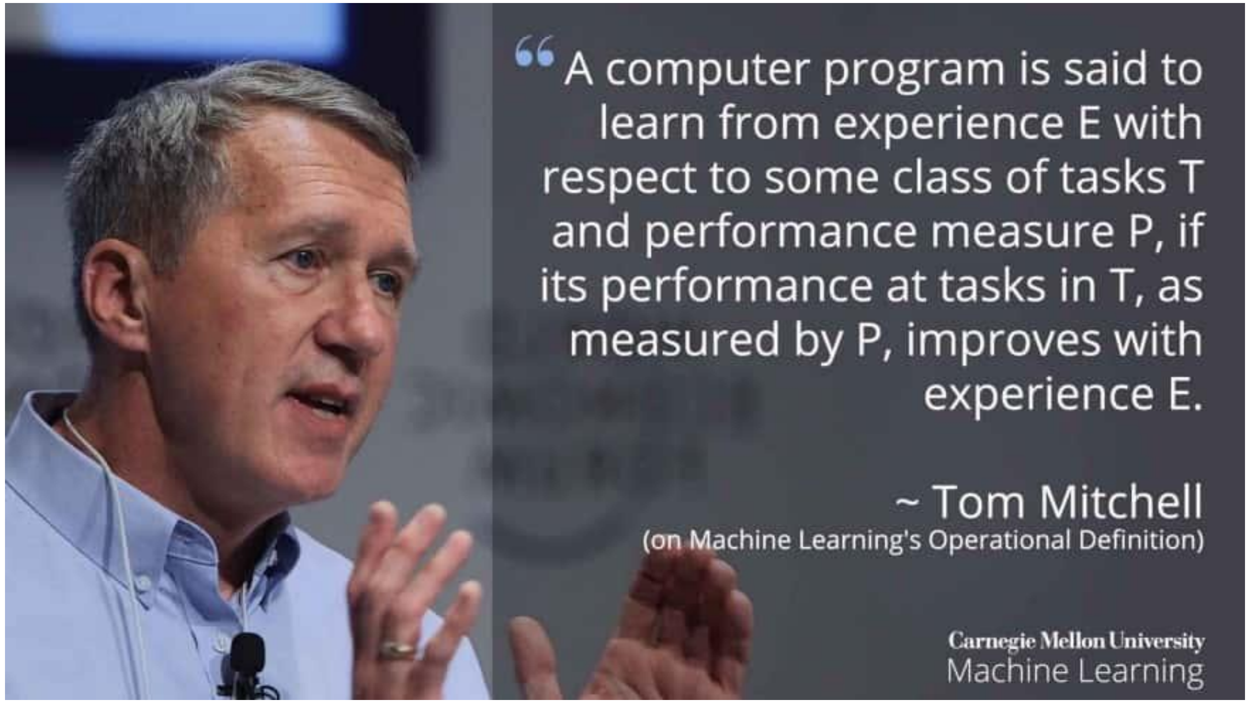


Image from internet

Summary – Day1

Key concepts in machine learning:

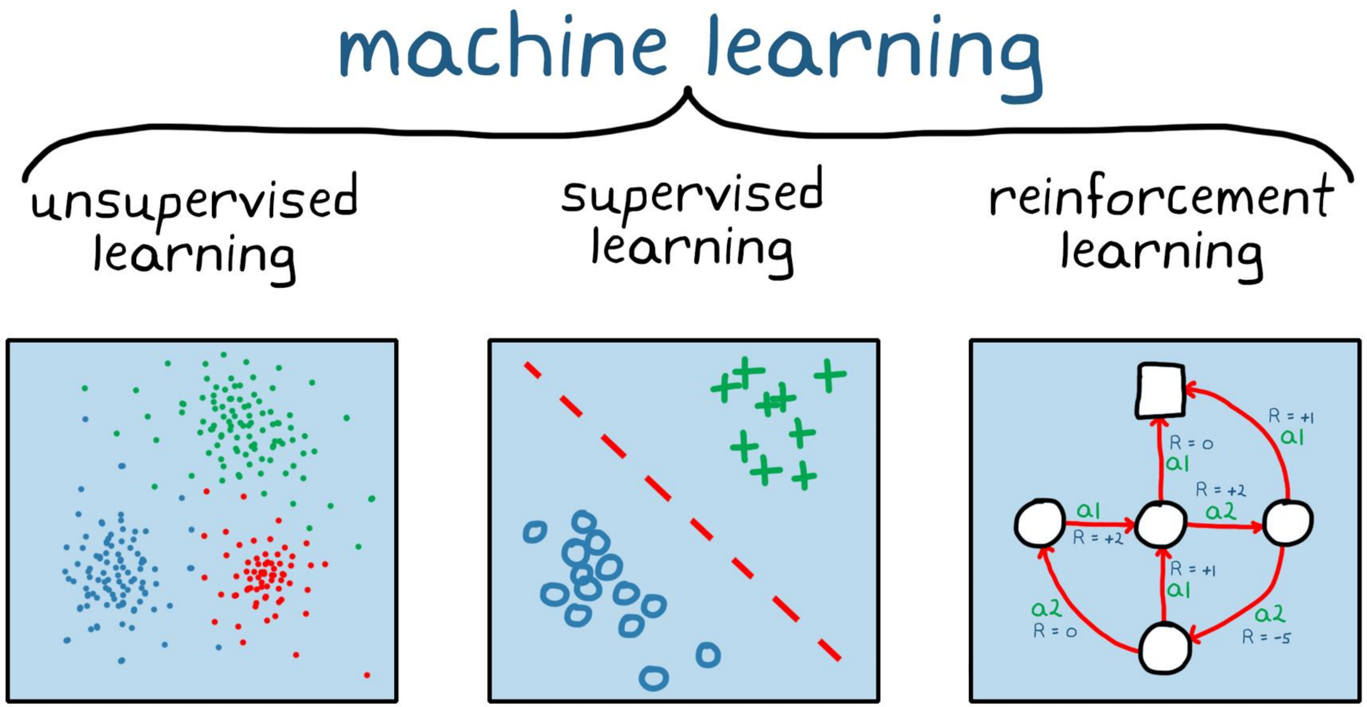
- ❑ What's machine learning



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 tunning
- ❑ 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

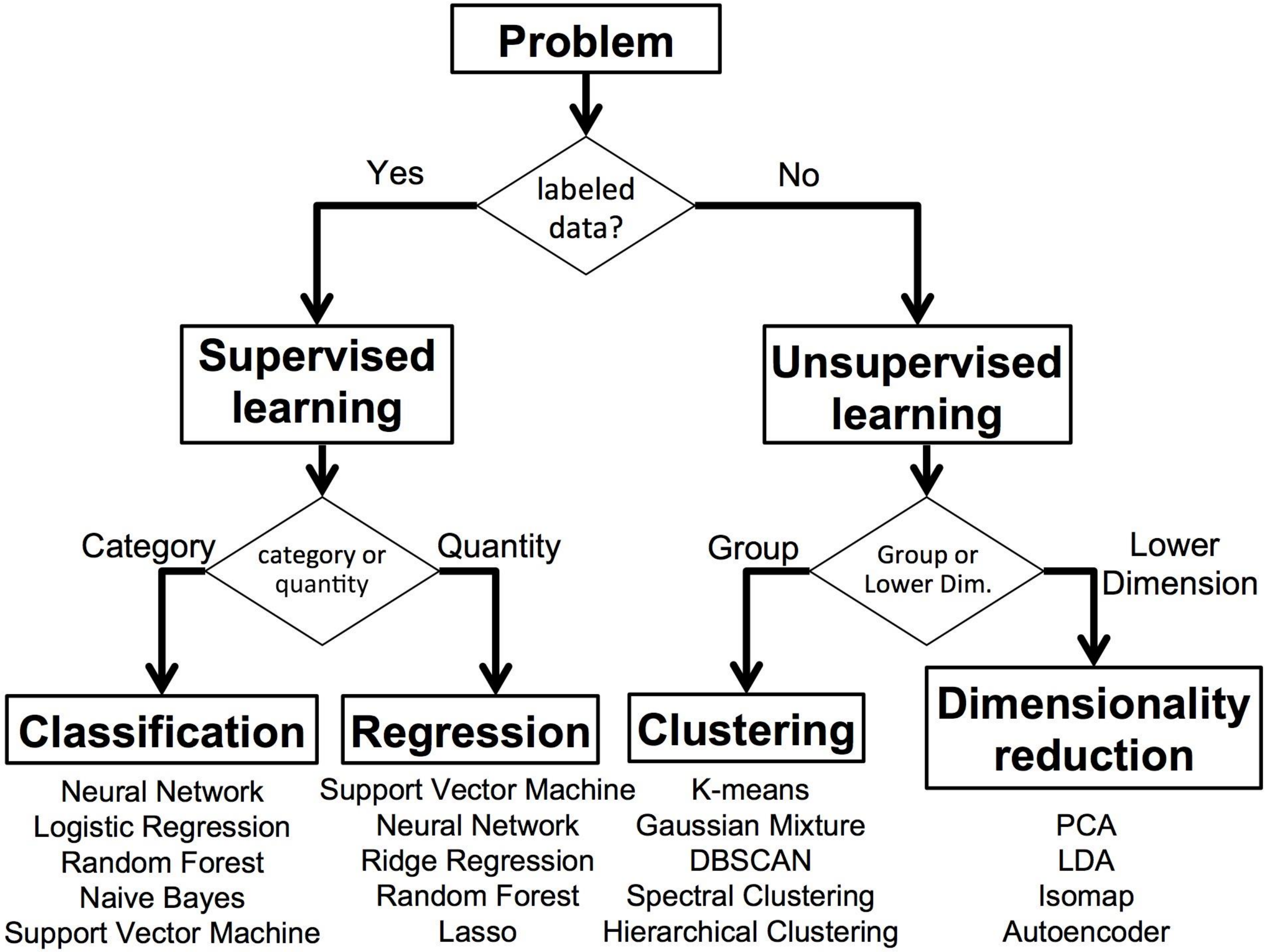
More details about:

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

Practice:

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

Types of machine learning



https://pythonnumericalmethods.berkeley.edu/notebooks/chapter25.01-Concept-of-Machine-Learning.html

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**

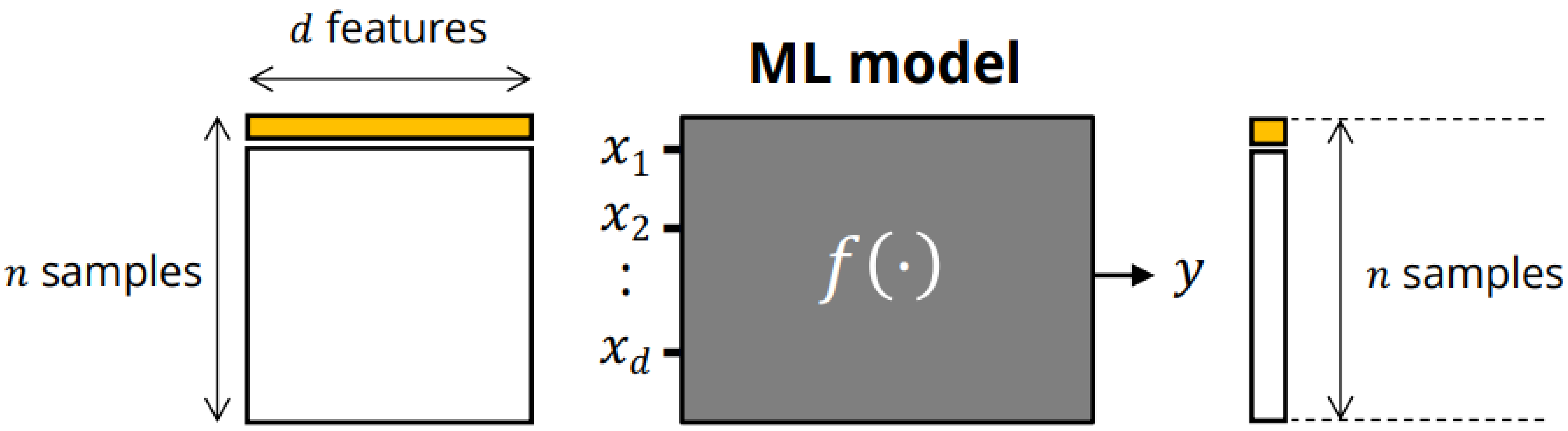
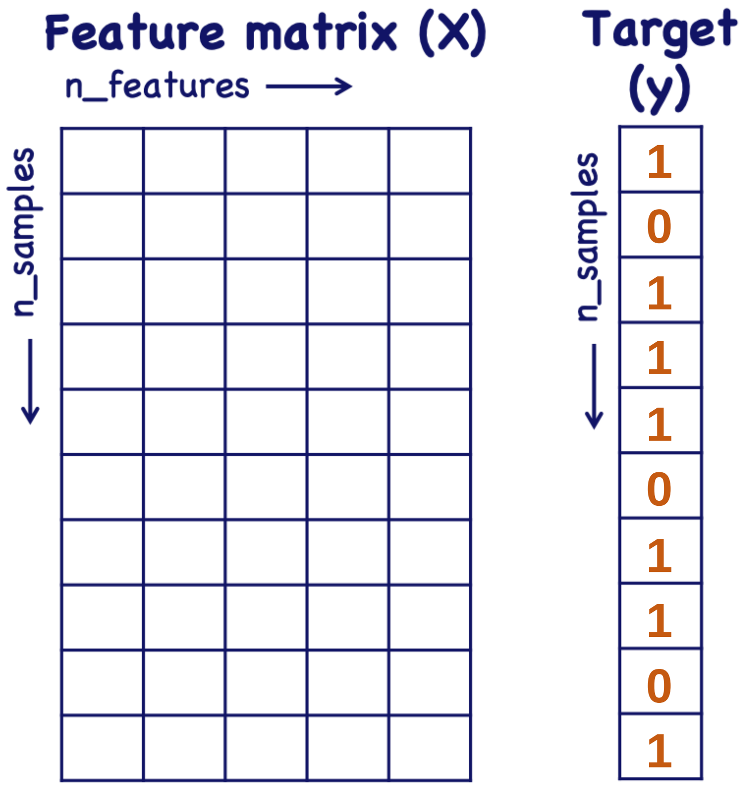


Figure adopted from CSEP 590B Explainable AI

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

…



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

https://pythonnumericalmethods.berkeley.edu/notebooks/chapter25.01-Concept-of-Machine-Learning.html

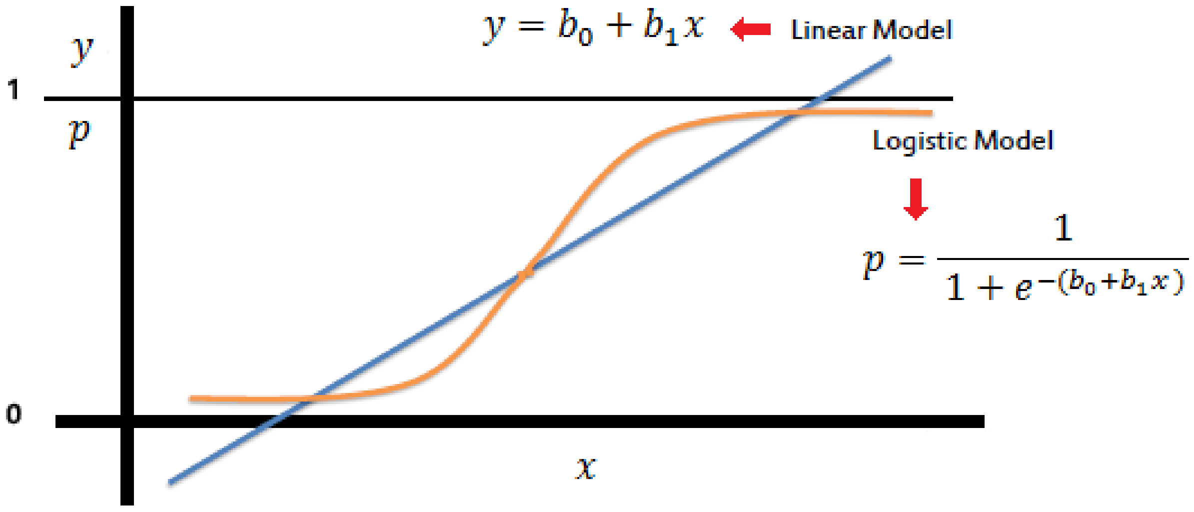
Logistic regression

- Model

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\mathbf{x}^\top \boldsymbol{\theta}\right)$$

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

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



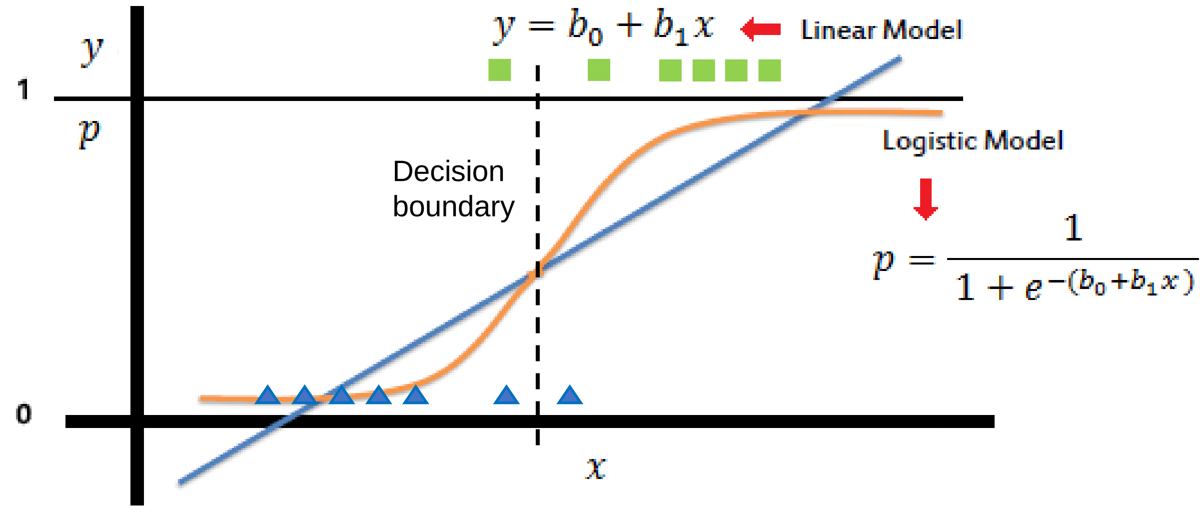
https://www.saedsayad.com/logistic_regression.htm

Logistic regression

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

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

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

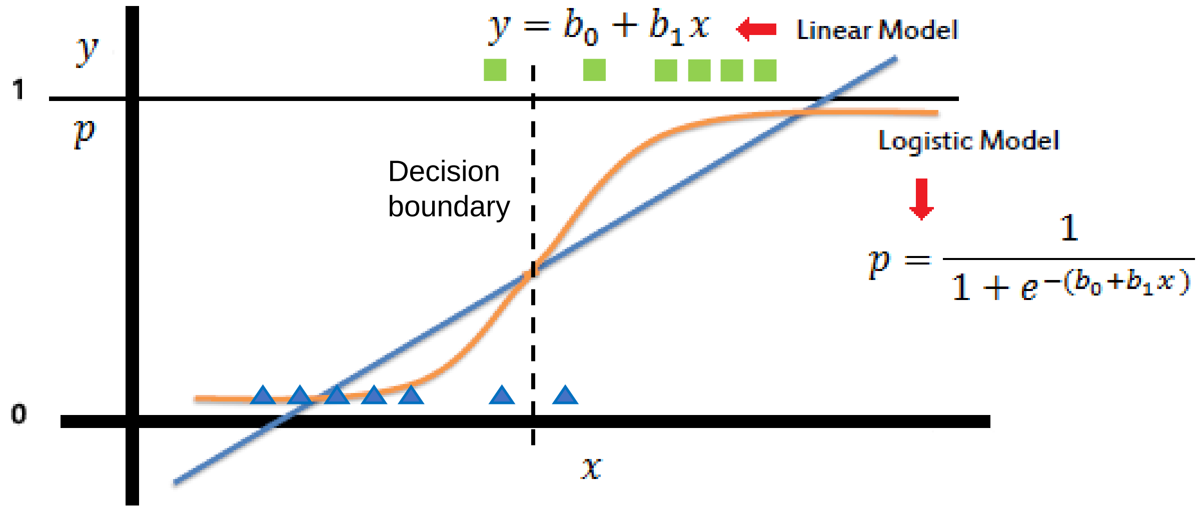


Logistic regression

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

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

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



- Loss function:

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

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

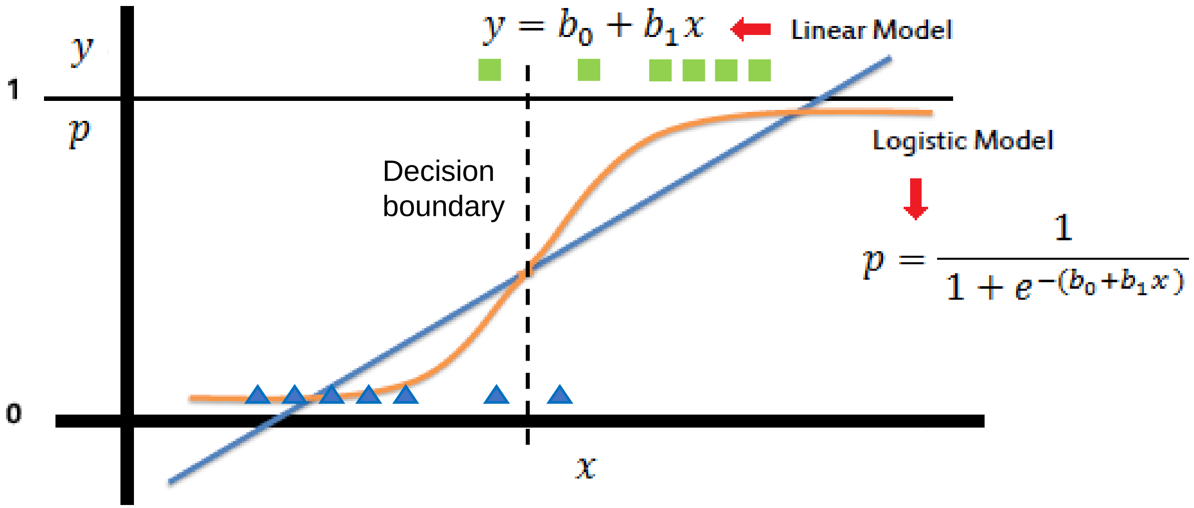
Logistic regression

- Model

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

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

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



- 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)

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

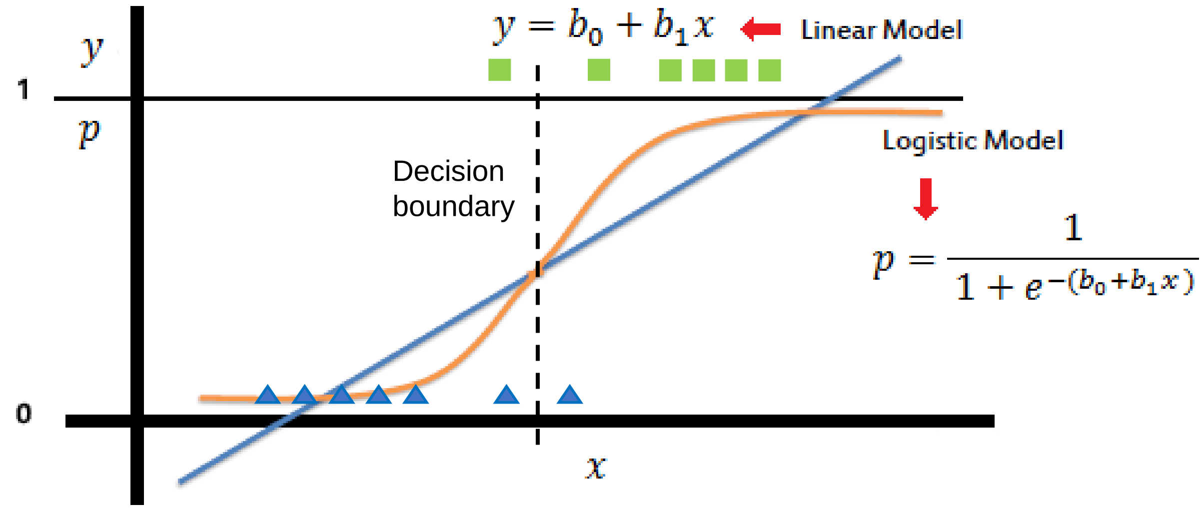
Logistic regression

- Model

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

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

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



- Loss function (generalize to multi-class):

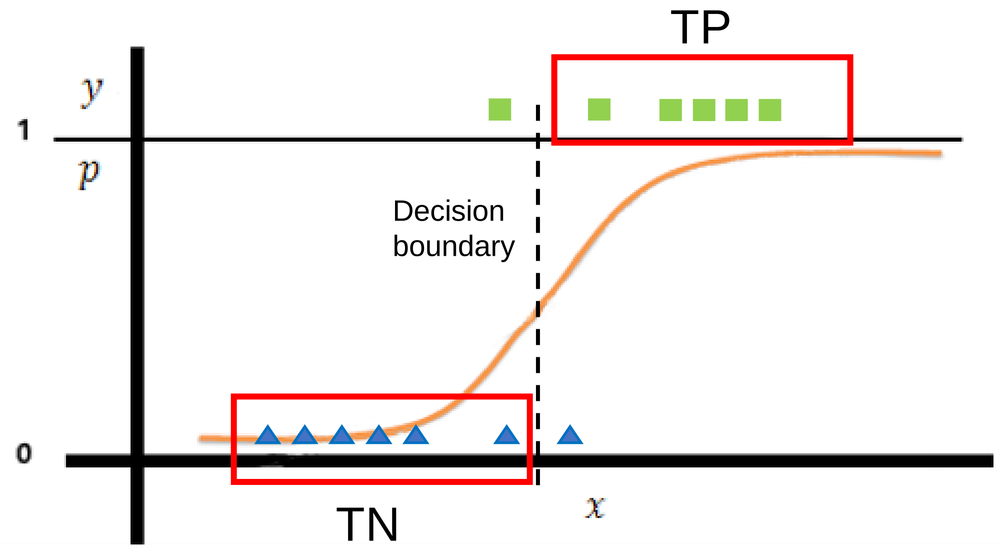
Equation 4-22. Cross entropy cost function

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

Performance measure

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

Q: Is accuracy always a good measure?

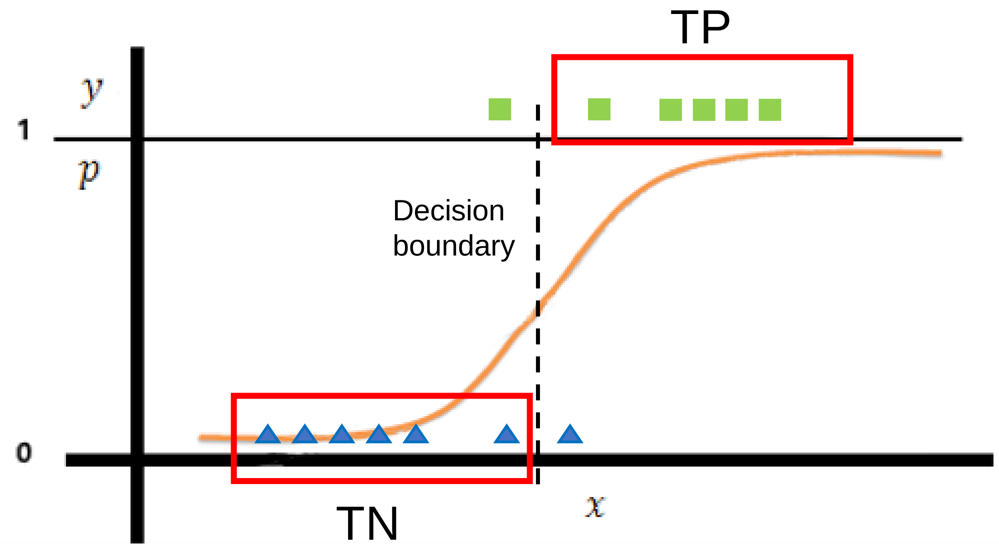


Performance measure

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

Q: Is accuracy always a good measure?

Be cautious with *skewed dataset* !!!

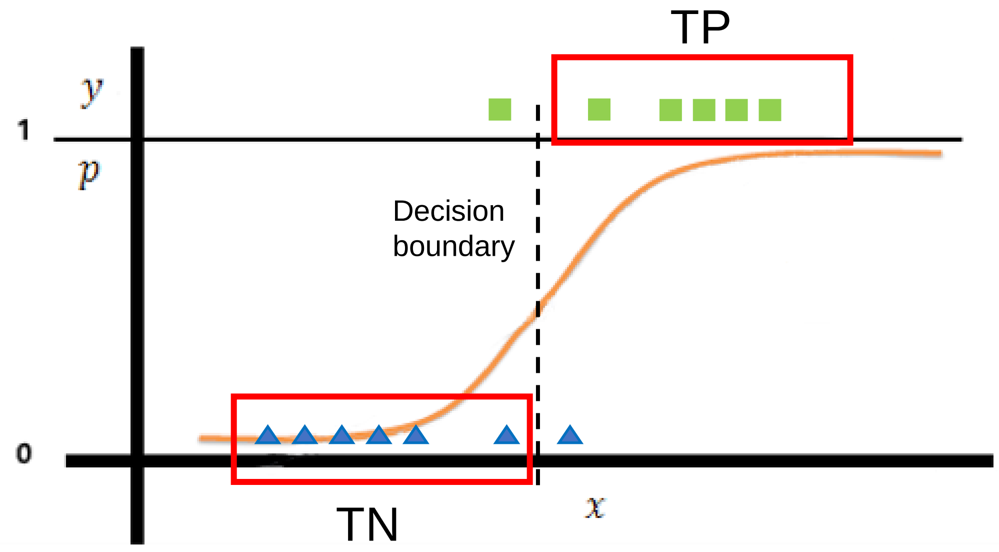


Performance measure

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

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**

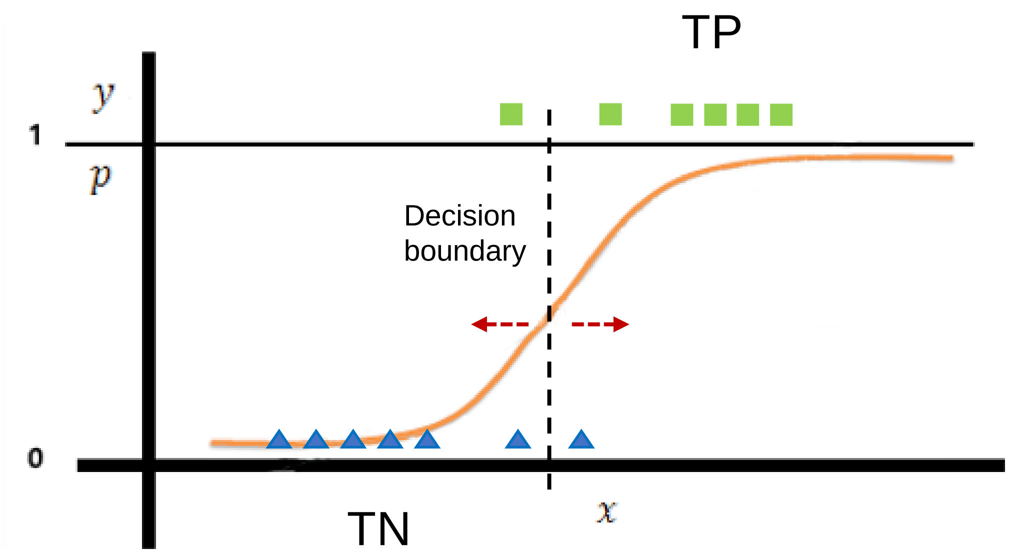
$$\text{precision} = \frac{TP}{TP + FP}$$

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

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

Performance measure (varying threshold)

- Precision-recall tradeoff



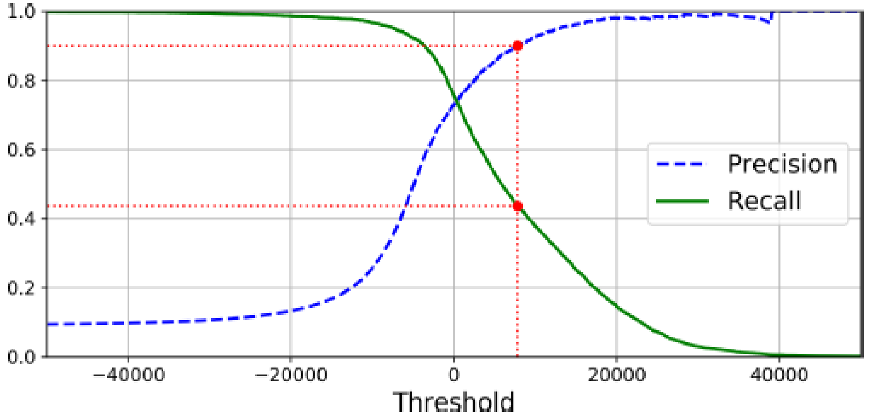
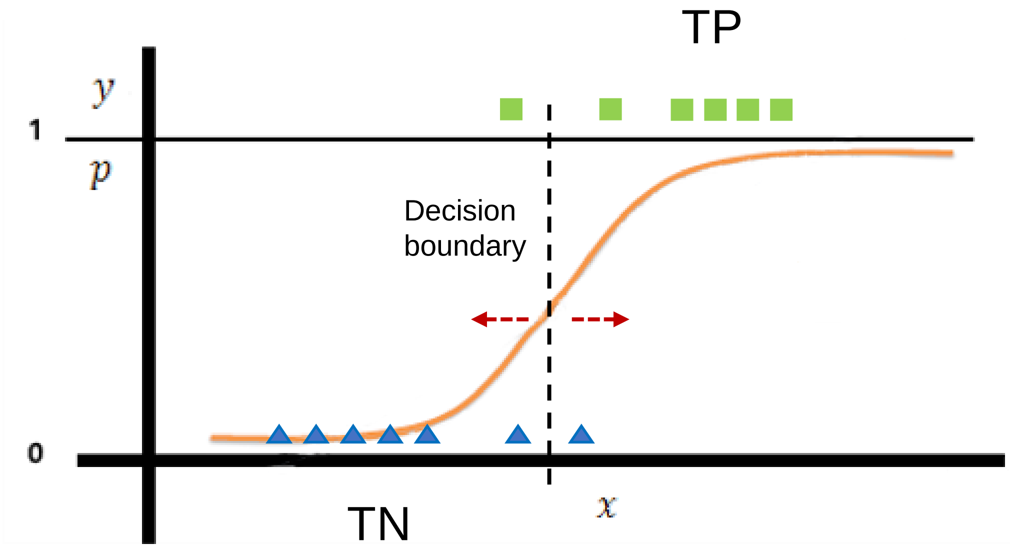


Figure 3-4. Precision and recall versus the decision threshold

Performance measure (varying threshold)

- Precision-recall tradeoff



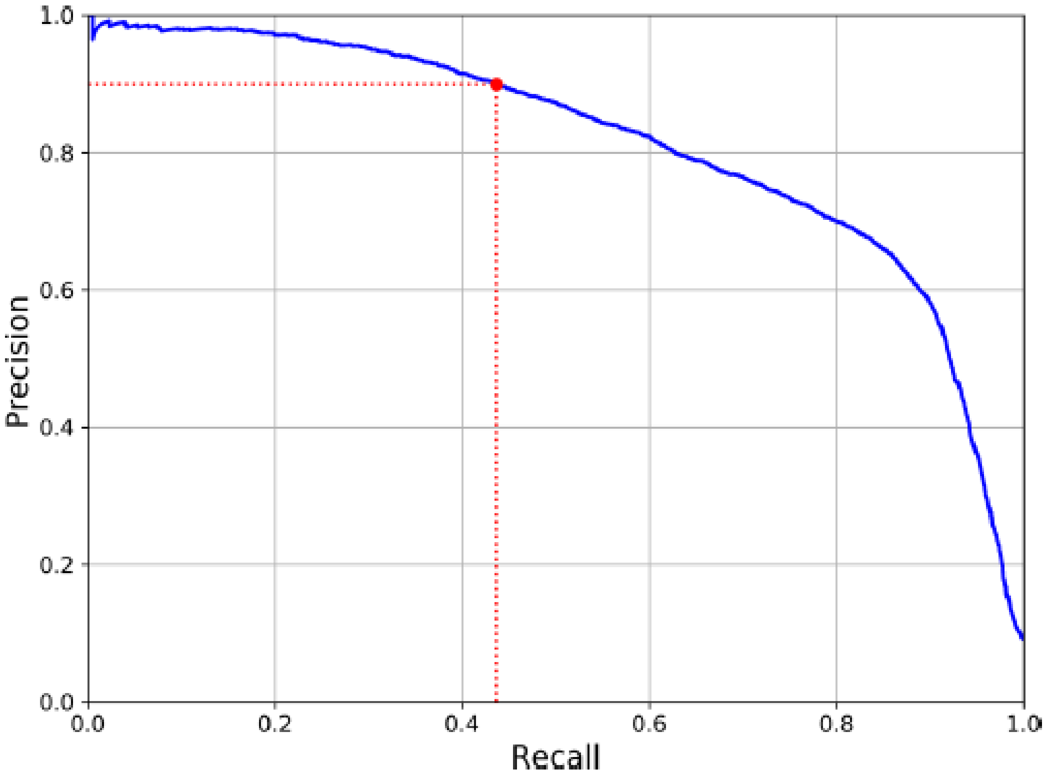
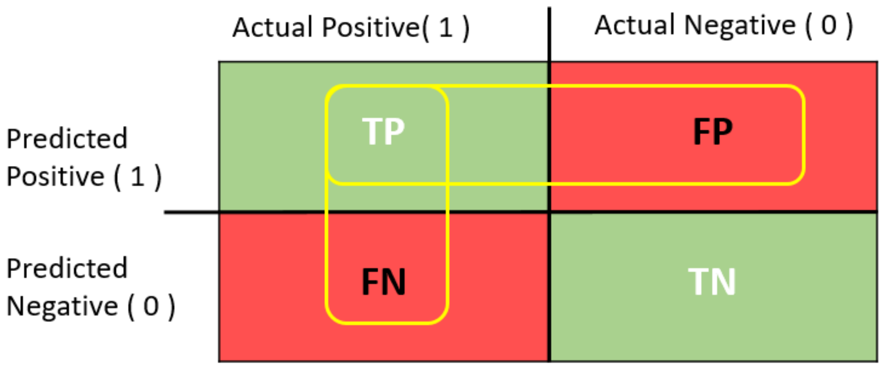


Figure 3-5. Precision versus recall

PR-ROC

Performance measure (varying threshold)

- ROC curve & AUC
  - Receiver Operating Characteristic curve
  - Aera under curve



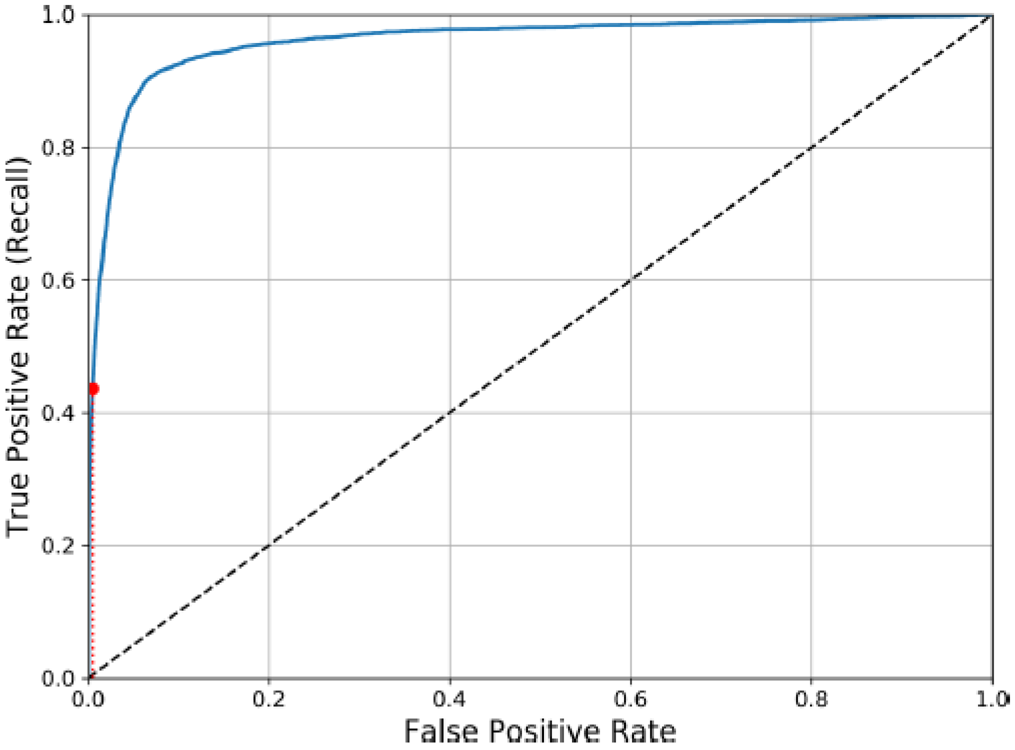


Figure 3-6. This ROC curve plots the false positive rate against the true positive rate for all possible thresholds; the red circle highlights the chosen ratio (at 43.68% recall)

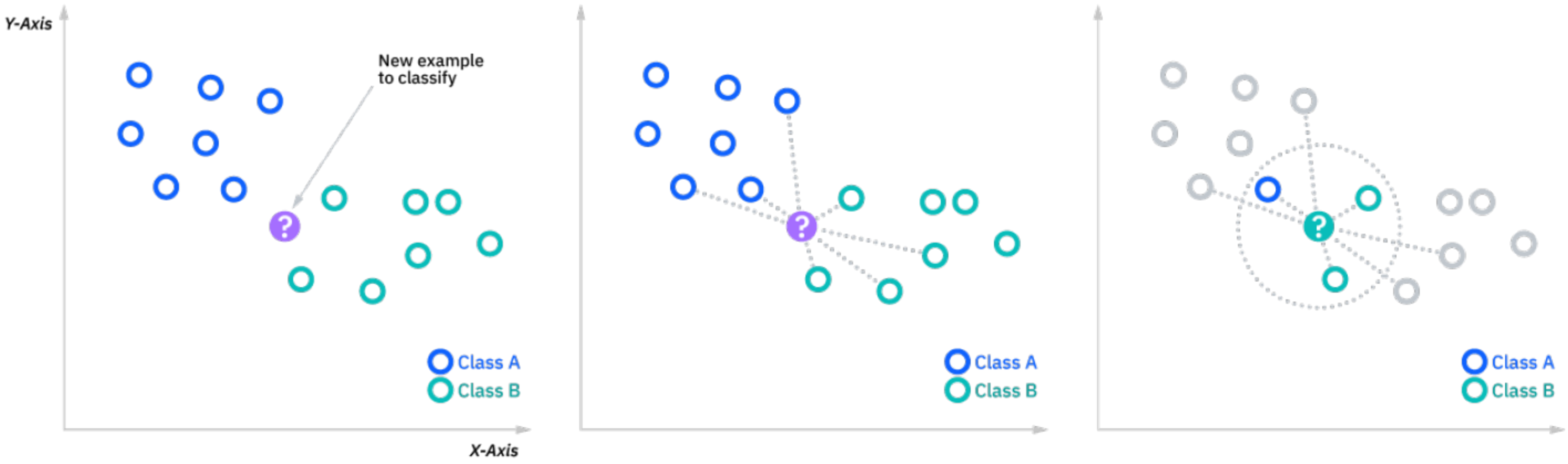
True positive rate (TPR), TP/(TP+FN)
False positive rate (FPR), FP/(FP+TN)

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)

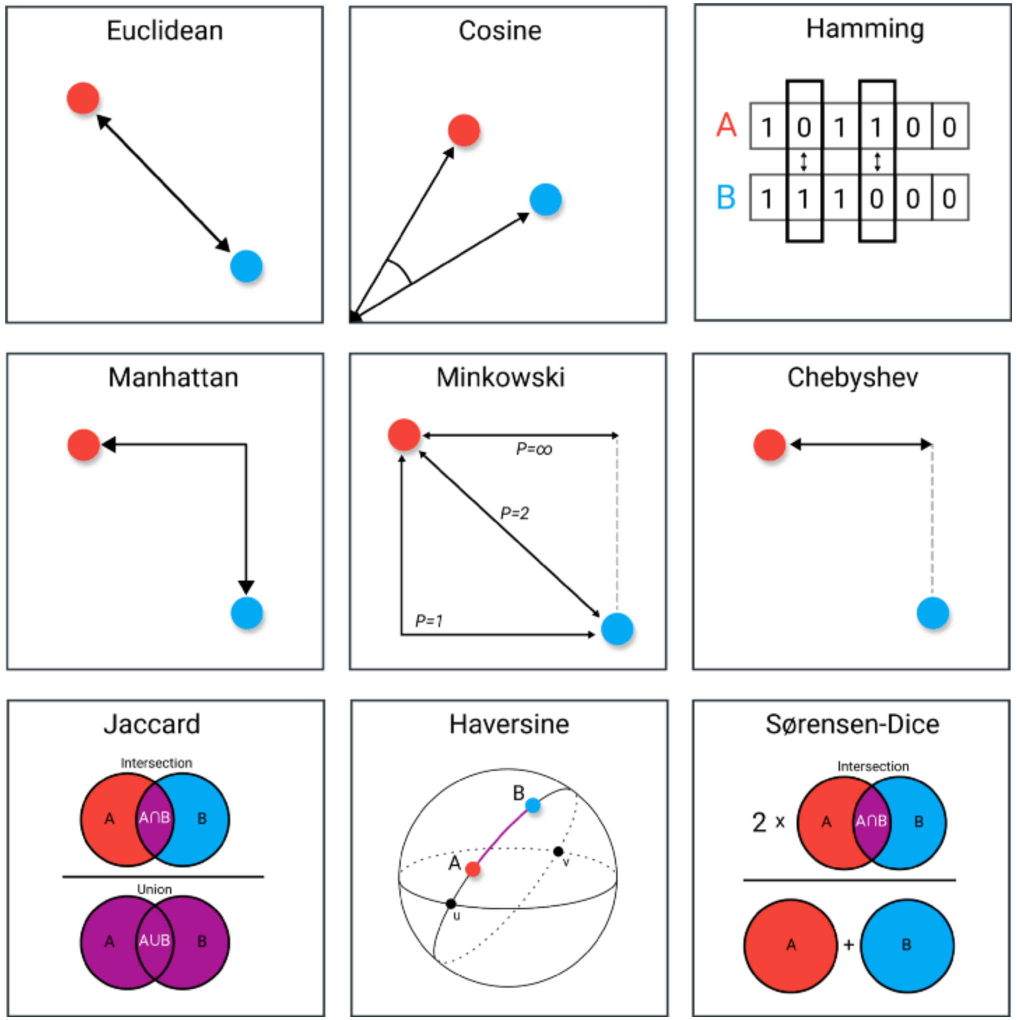
$$D(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2)}$$

  - Manhattan distance (L1 norm)

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

  - Minkowski distance (Lp norm)

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



https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

Naïve bayes

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathrm{X}) \propto P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Q: Why Naïve Bayes is called naïve?

https://www.saedsayad.com/naive_bayesian.htm

Naïve bayes

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathrm{X}) \propto P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Q: Why Naïve Bayes is called naïve?

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

Naïve bayes

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathrm{X}) \propto P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- Gaussian Naïve Bayes

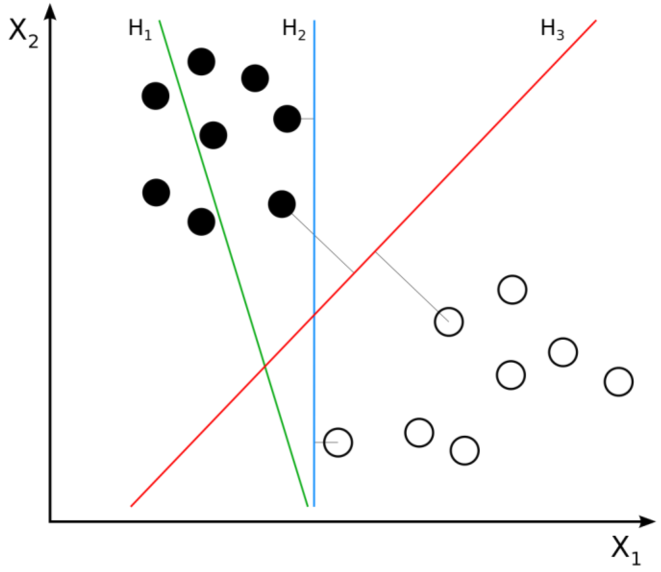
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Support vector machine

- Decision function

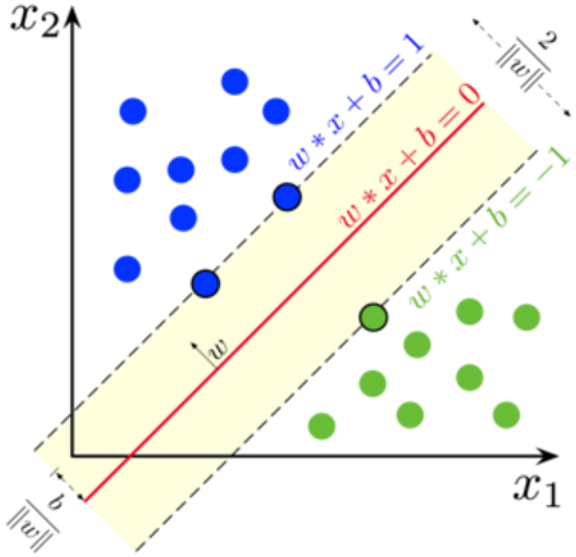
Equation 5-2. Linear SVM classifier prediction

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



Question: Which line is the best?

Support vector machine



- Large margin classification

Equation 5-3. Hard margin linear SVM classifier objective

$$\begin{aligned}
&\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} \\
&\text{subject to} \quad t^{(i)}\left(\mathbf{w}^\top\mathbf{x}^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \cdots, m
\end{aligned}$$

- 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

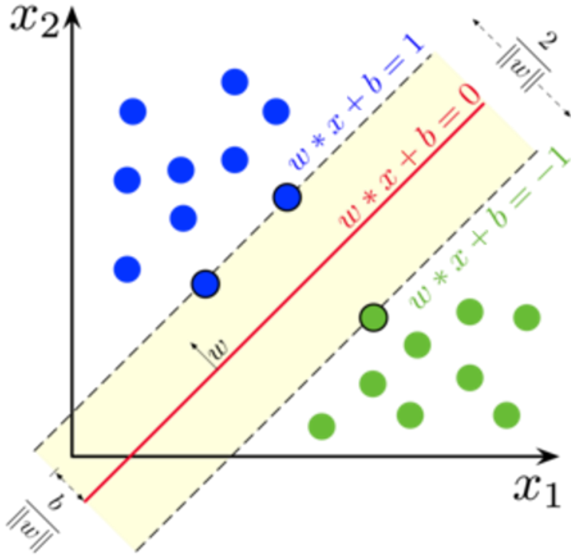
Support vector machine

- Large margin classification

Equation 5-3. Hard margin linear SVM classifier objective

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

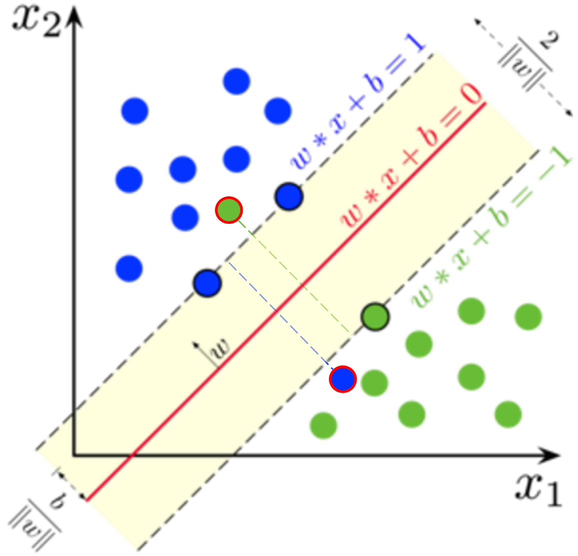
$$\text{subject to} \quad t^{(i)}\left(\mathbf{w}^\intercal \mathbf{x}^{(i)} + b\right) \geq 1 \quad \text{for } i = 1, 2, \cdots, m$$



- Soft-margin classification

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

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



Support vector machine (nonlinear)

- With polynomial kernel

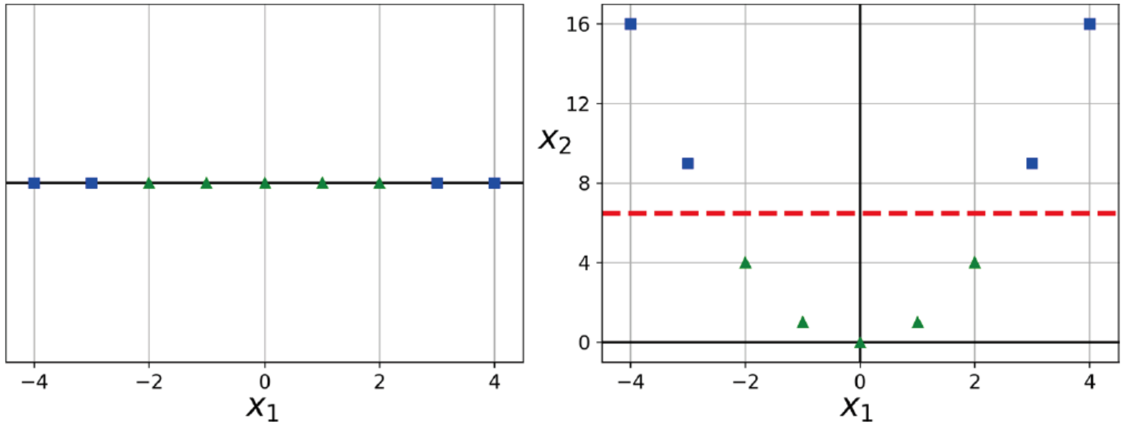


Figure 5-5. Adding features to make a dataset linearly separable

- 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

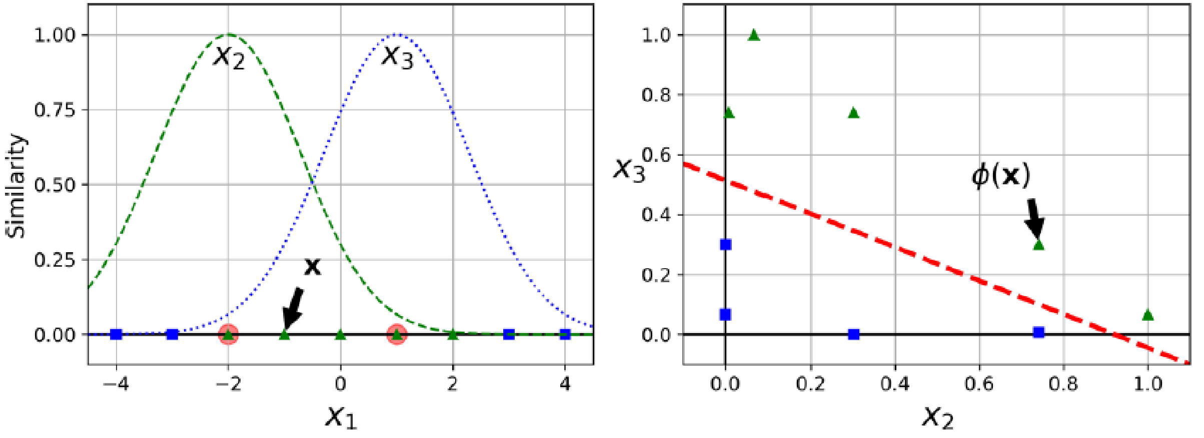


Figure 5-8. Similarity features using the Gaussian RBF

Decision trees

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

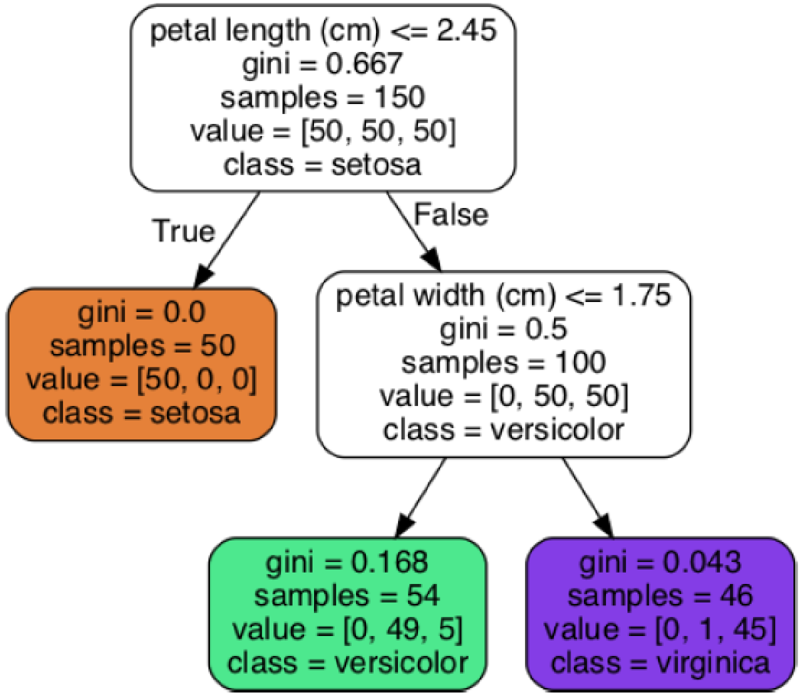
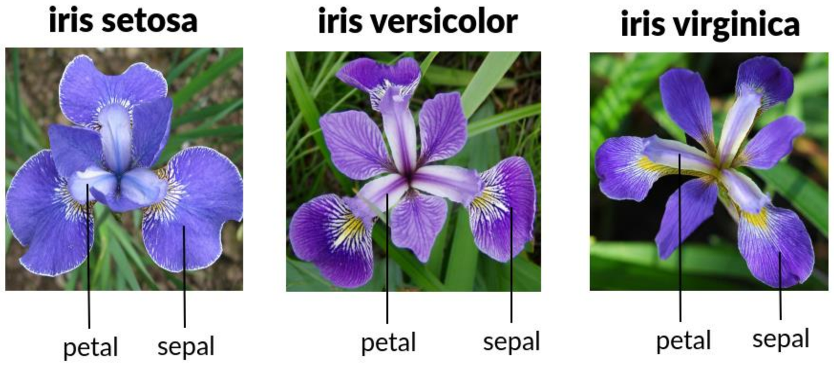


Figure 6-1. Iris Decision Tree



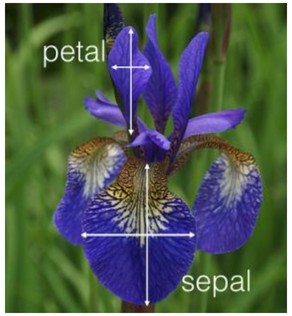


Image from internet

Decision trees

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

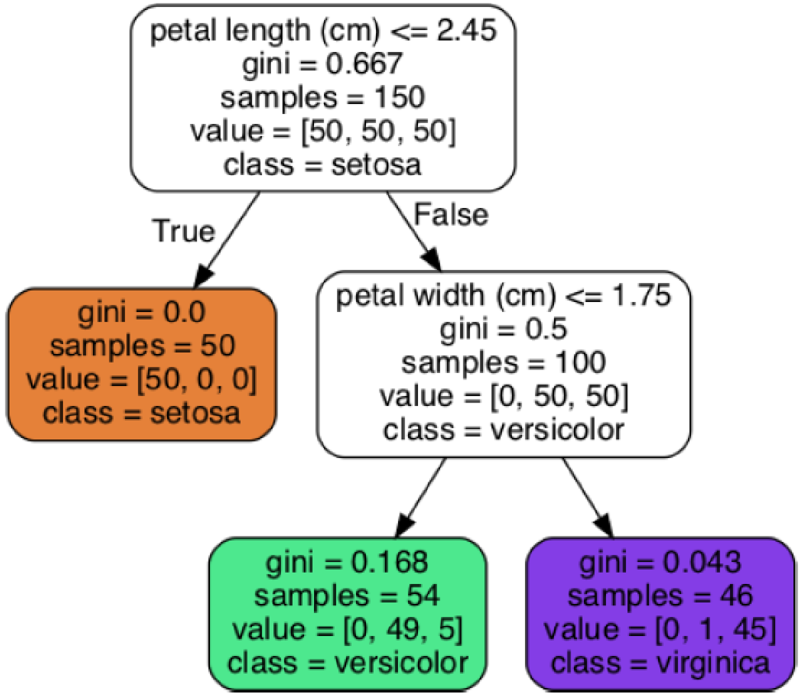


Figure 6-1. Iris Decision Tree

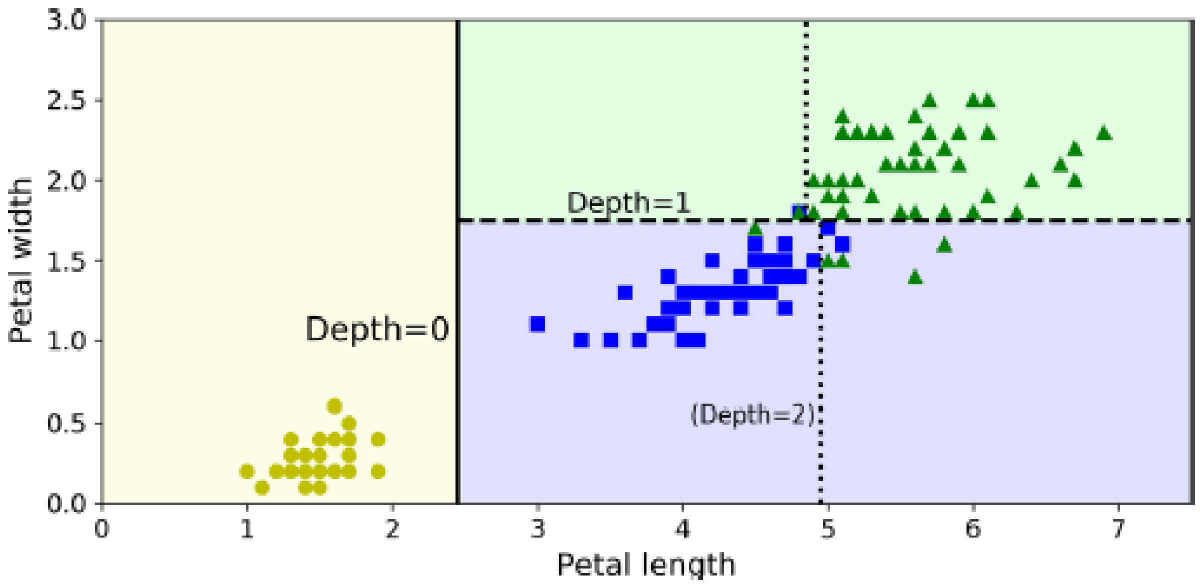


Figure 6-2. Decision Tree decision boundaries

Decision trees

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

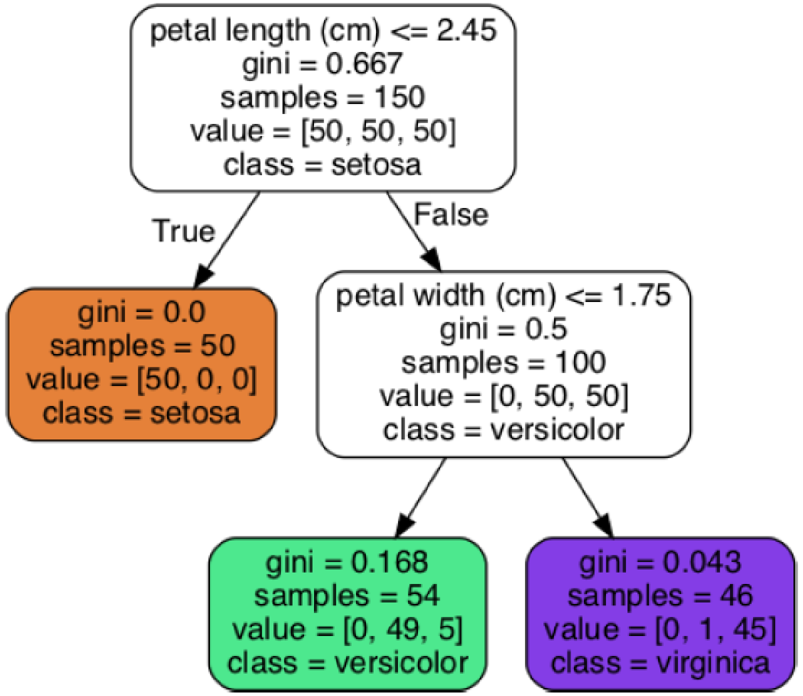


Figure 6-1. Iris Decision Tree

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

Equation 6-2. CART cost function for classification

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

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

Decision trees

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

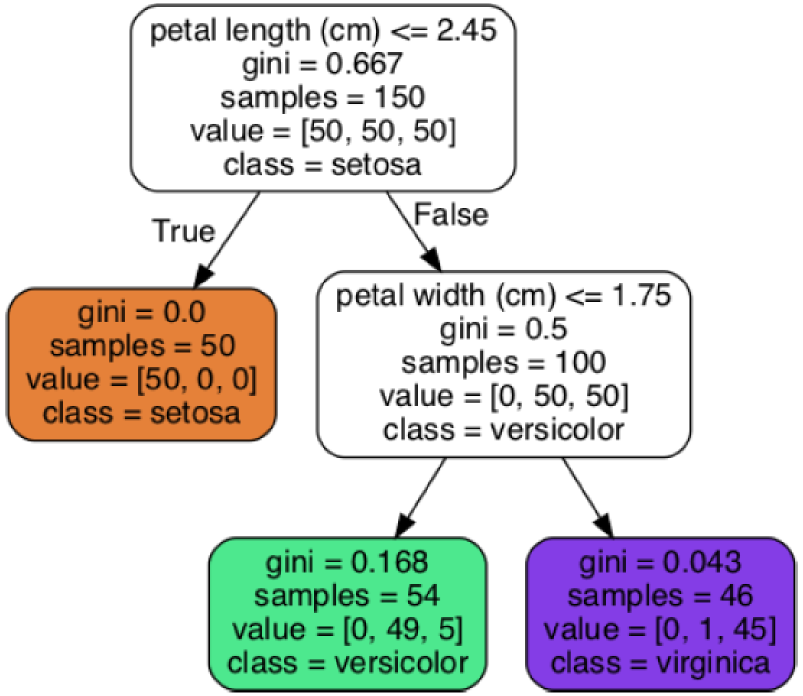


Figure 6-1. Iris Decision Tree

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

Equation 6-2. CART cost function for classification

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

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

- 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

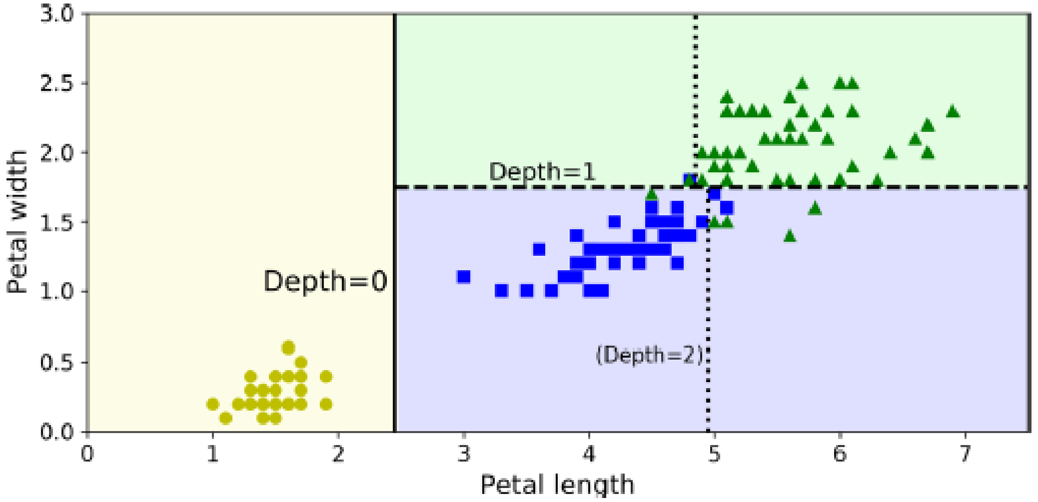


Figure 6-2. Decision Tree decision boundaries

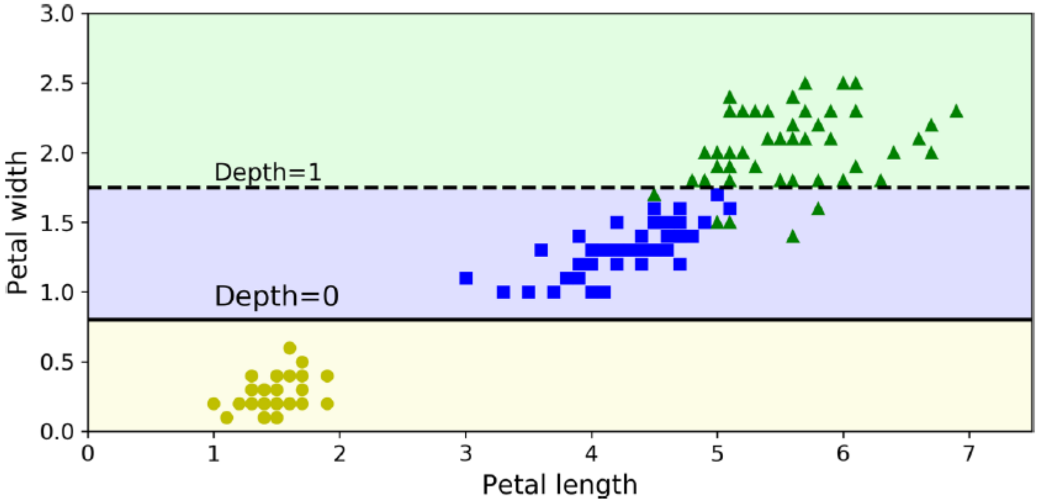


Figure 6-8. Sensitivity to training set details

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

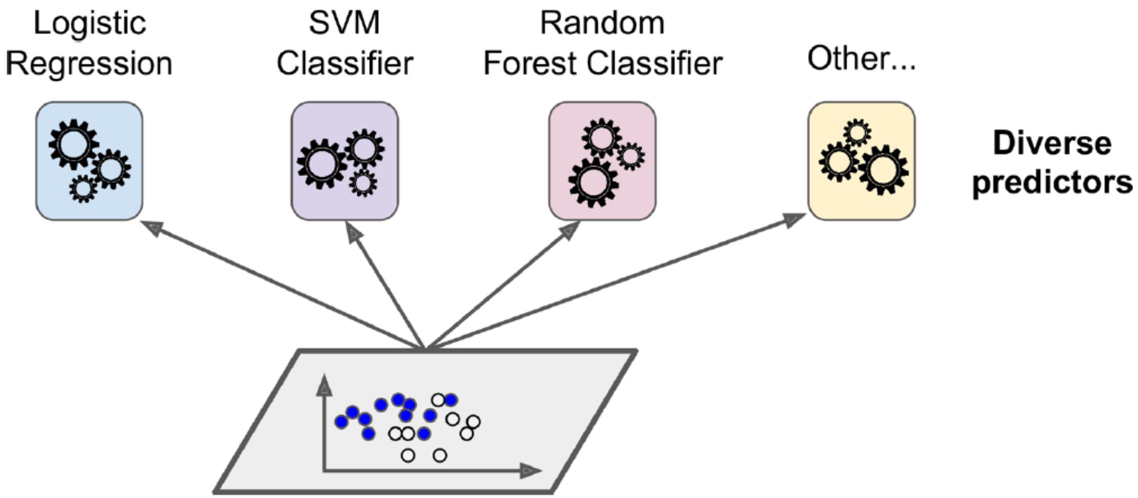


Figure 7-1. Training diverse classifiers

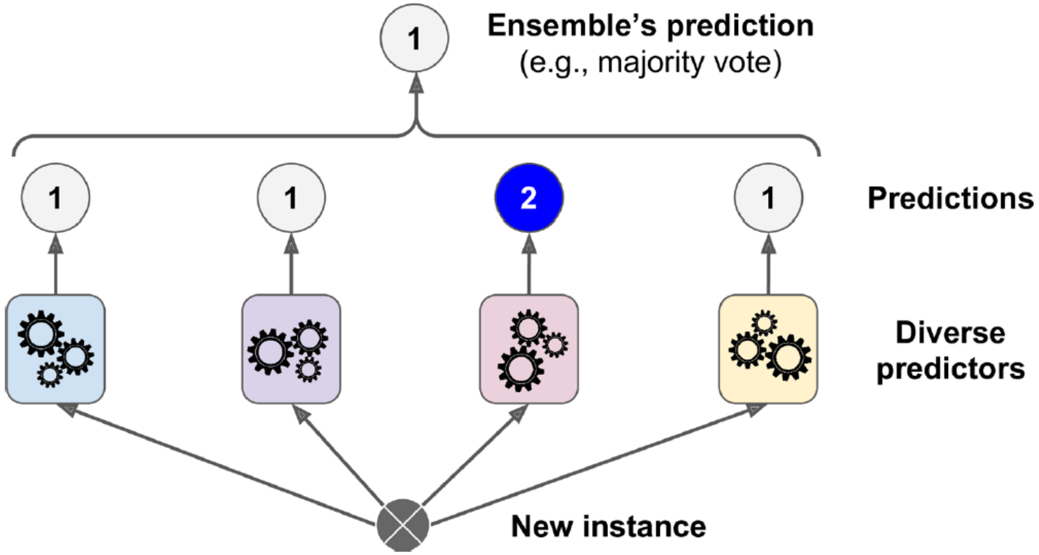
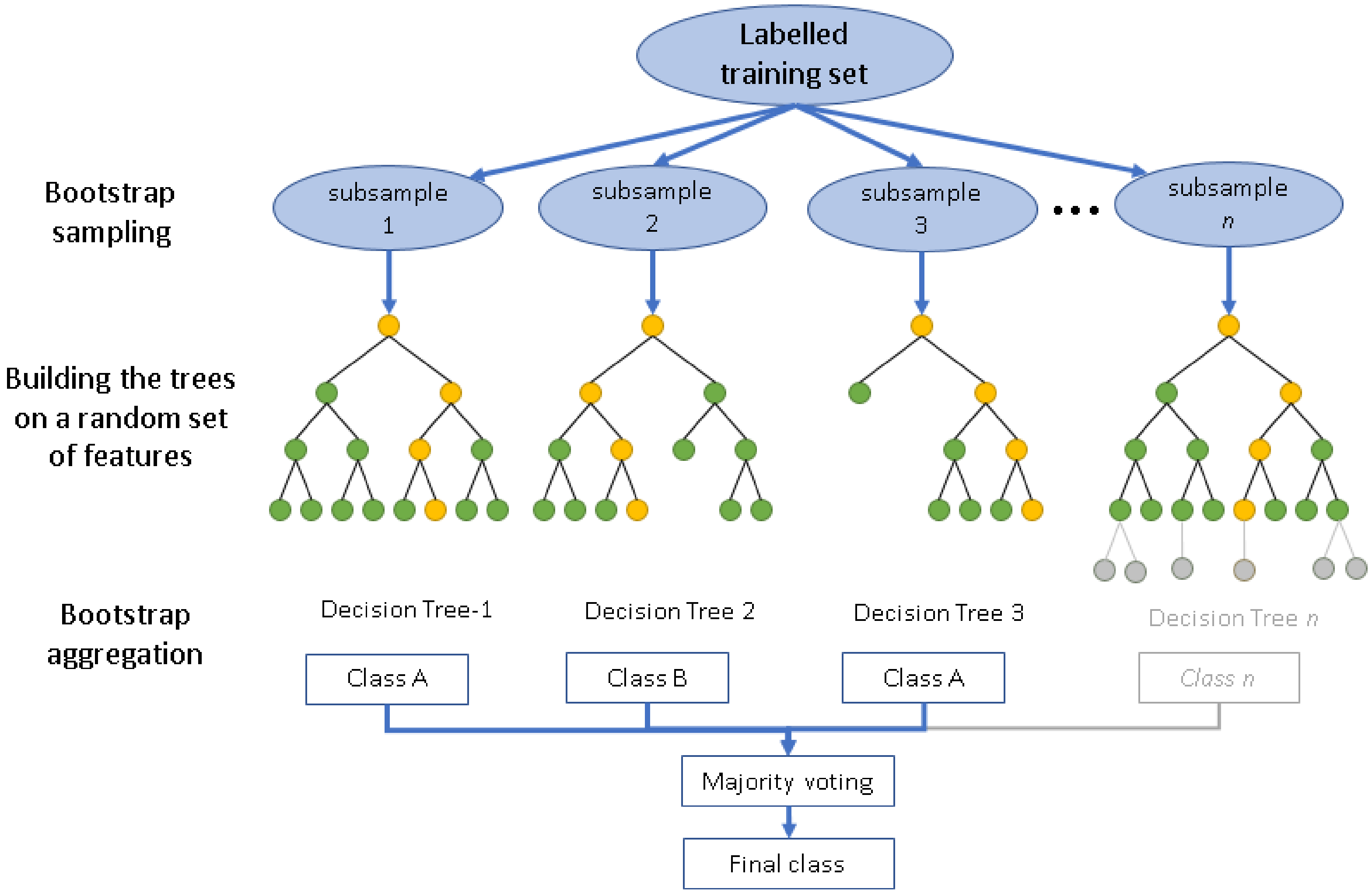


Figure 7-2. Hard voting classifier predictions

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

Random forest

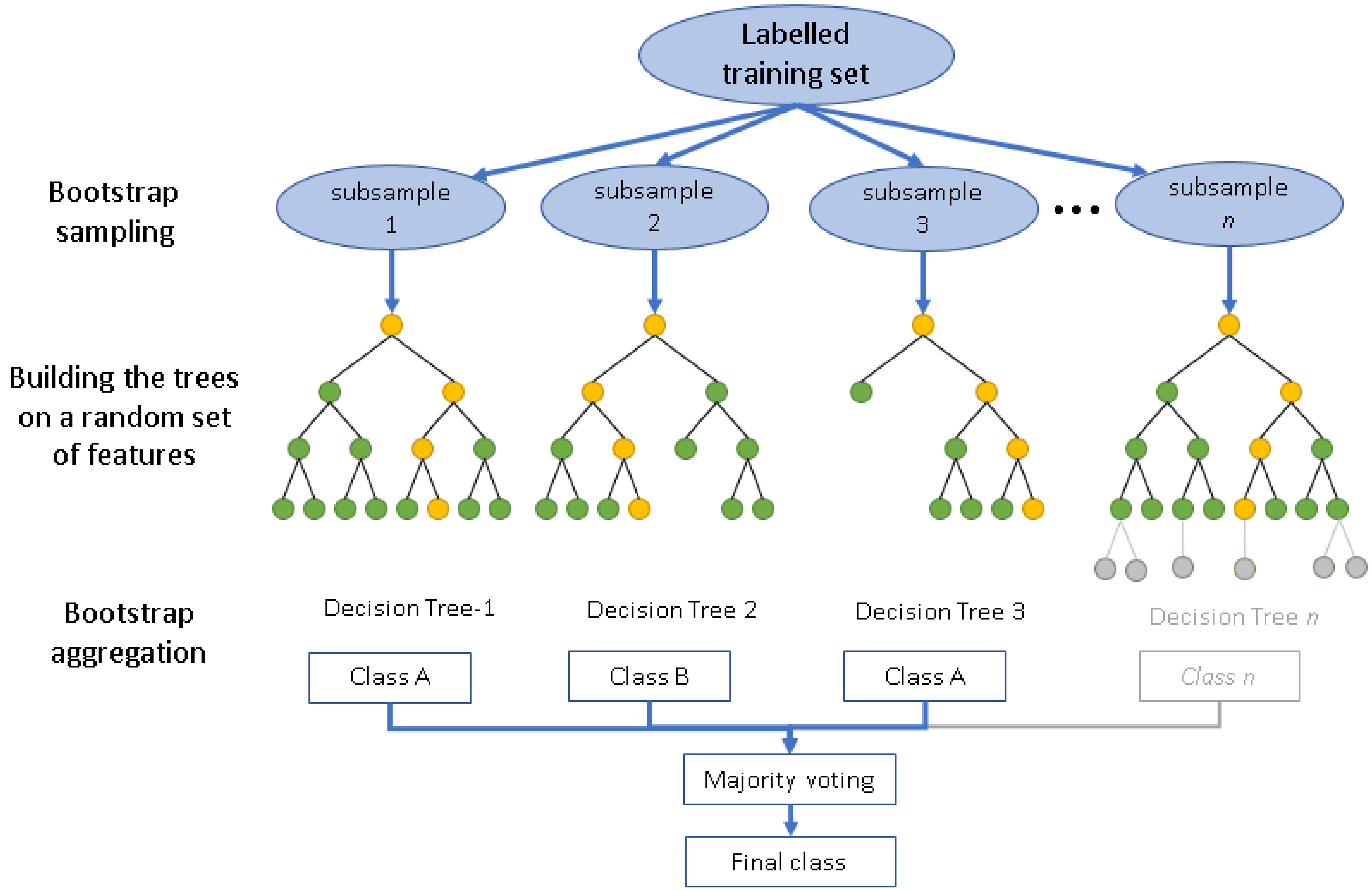
- An ensemble of decision trees



Reduce the variance of a single tree

Random forest

- An ensemble of decision trees

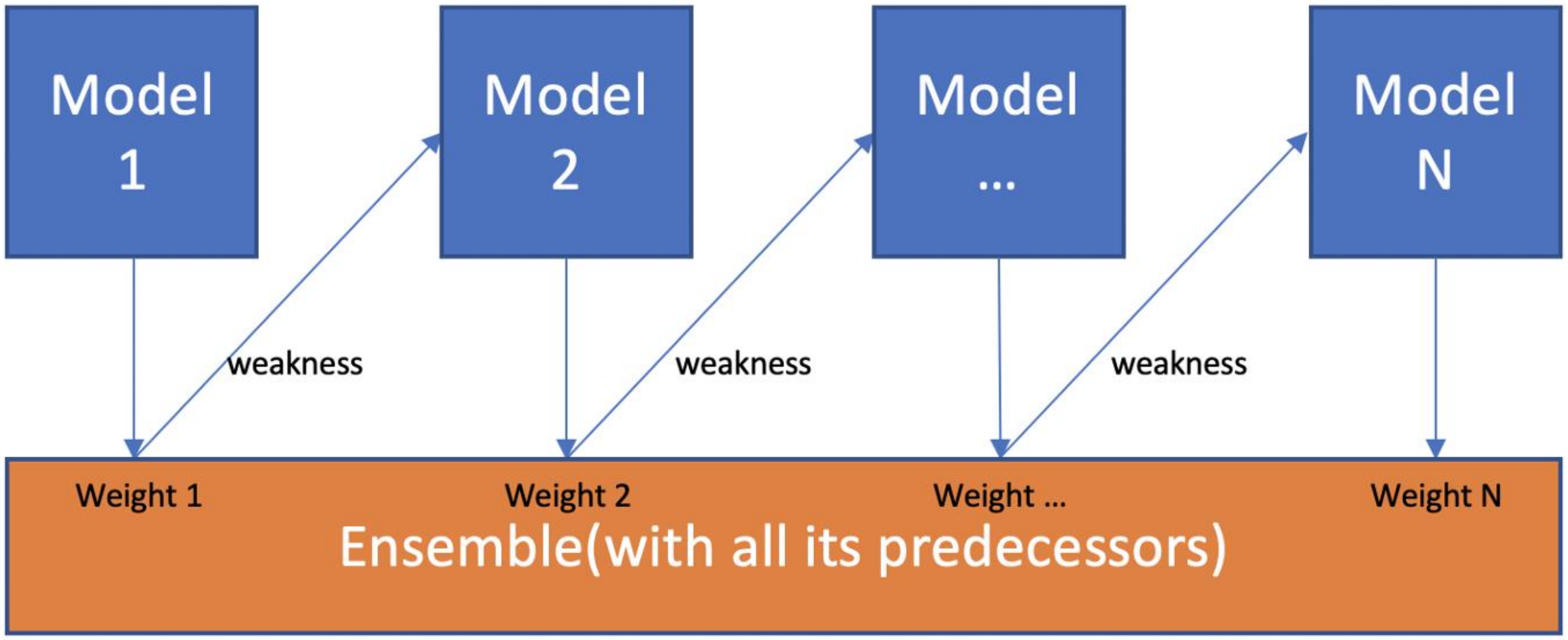


Feature importance:
how much the tree nodes that use that feature reduce impurity on average (across all trees in the forest)

Boosting method:

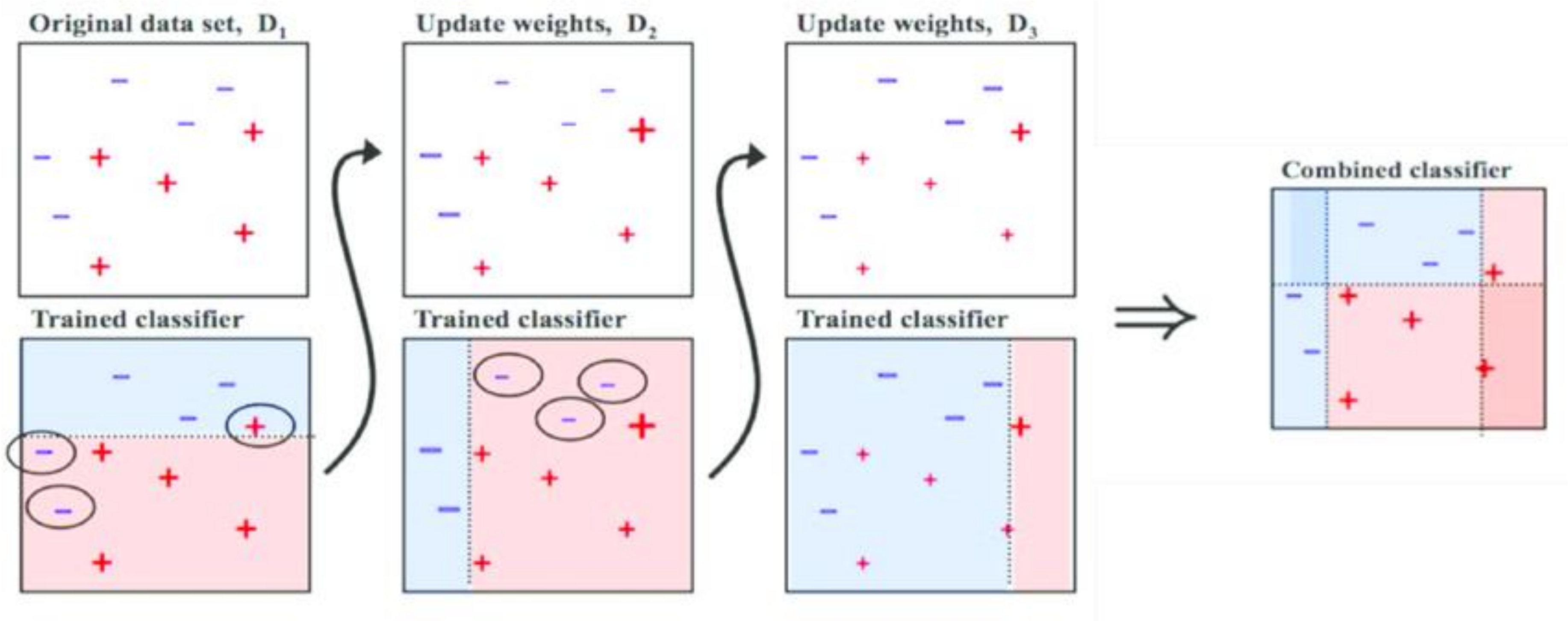
- Combine weak learner to form a strong learner
- Train predictors sequentially, each trying to correct its predecessor

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

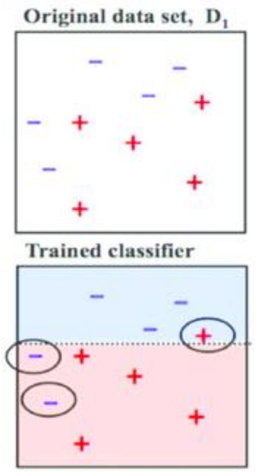


Adaboost

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



Adaboost

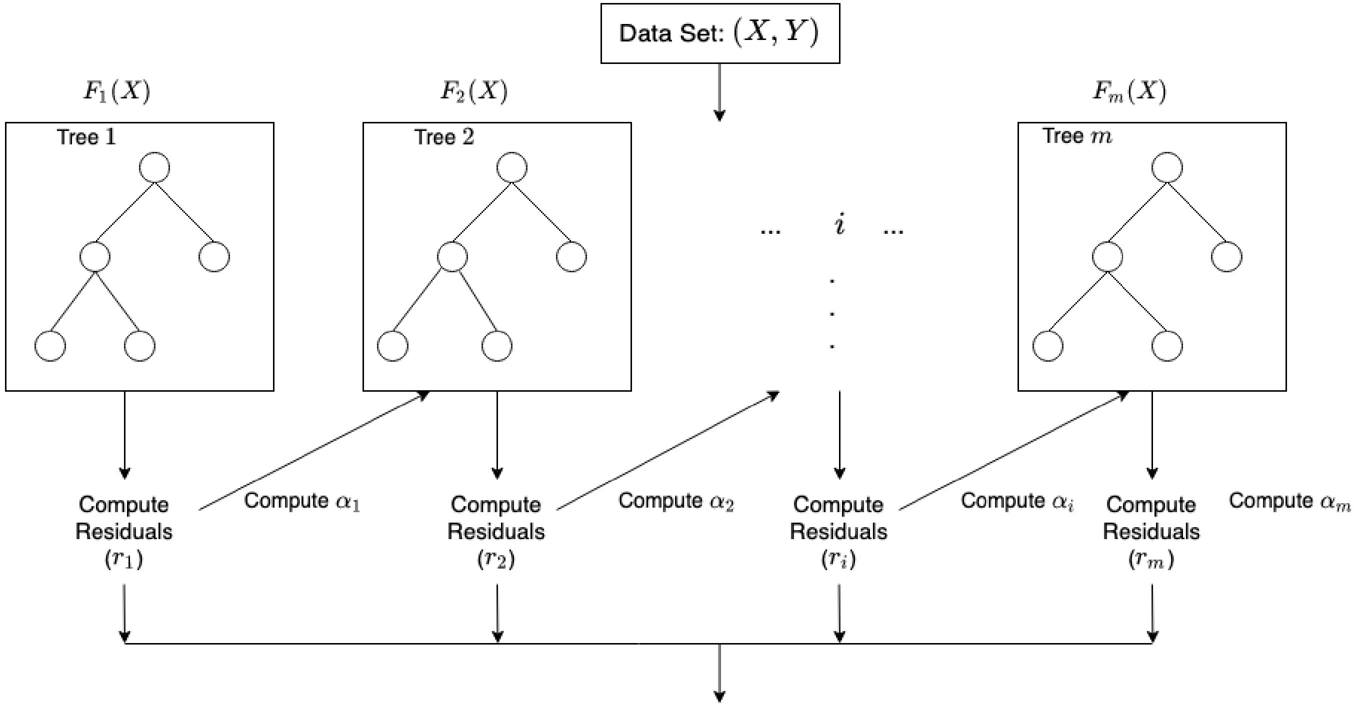


- Core algorithm:
  - ➢ Initialize with equal weights for each instance i
  - ➢ Compute weighted error rate for j^{th} predictor
    $$r_j = \frac{\sum\limits_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}}{\sum\limits_{i=1}^{m} w^{(i)}}$$
  - ➢ Compute the predictor's weight
    $$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$
  - ➢ Weight update for instance
    $$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \widehat{y_j}^{(i)} = y^{(i)} \\ w^{(i)} \exp\left(\alpha_j\right) & \text{if } \widehat{y_j}^{(i)} \neq y^{(i)} \end{cases}$$
  - ➢ Repeat until designed number of predictors is reached or perfect predictors is found

- 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



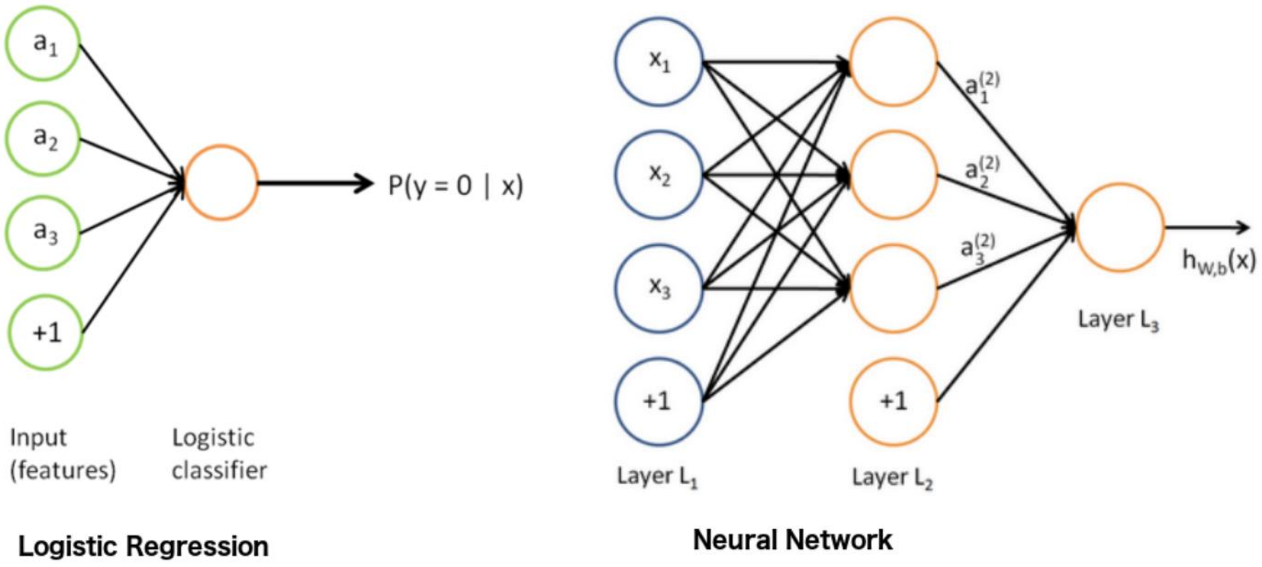
$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, 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.

https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

Neural network

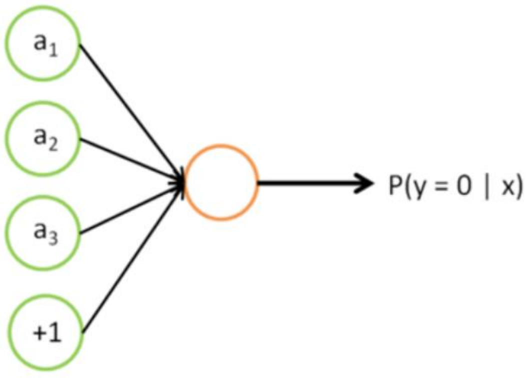
- Logistic regression can be regarded as a single layer of Neural network with sigmoid activation function



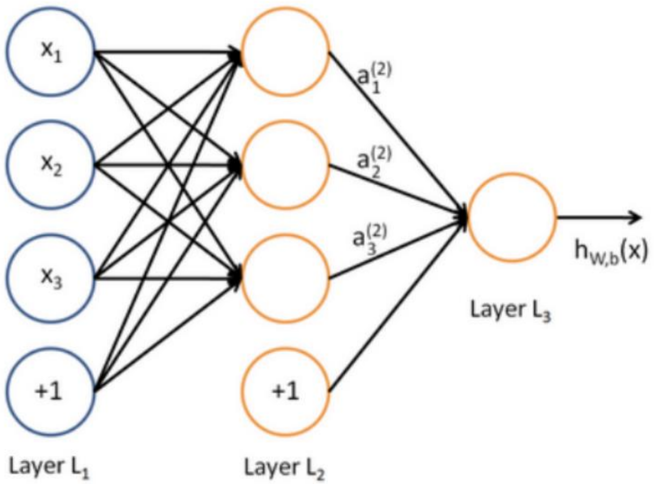
https://stats.stackexchange.com/questions/366707/a-logistic-regression-with-neural-network-mindset-vs-a-shallow-neural-network
https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

Neural network

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



Logistic Regression



Neural Network

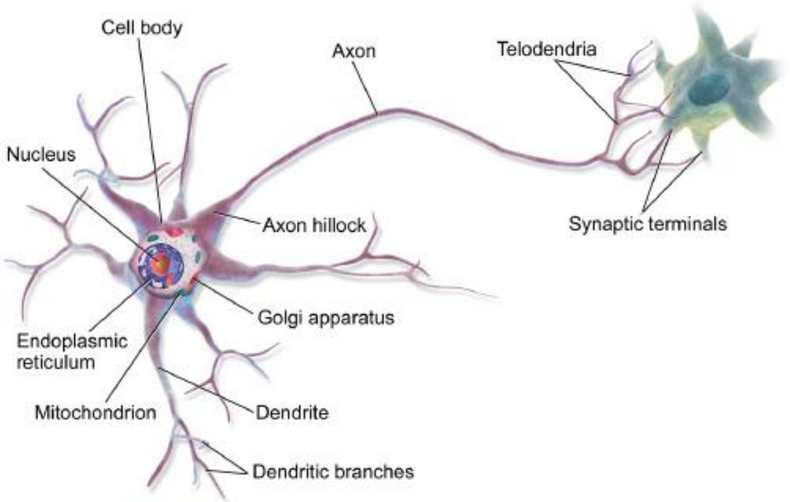
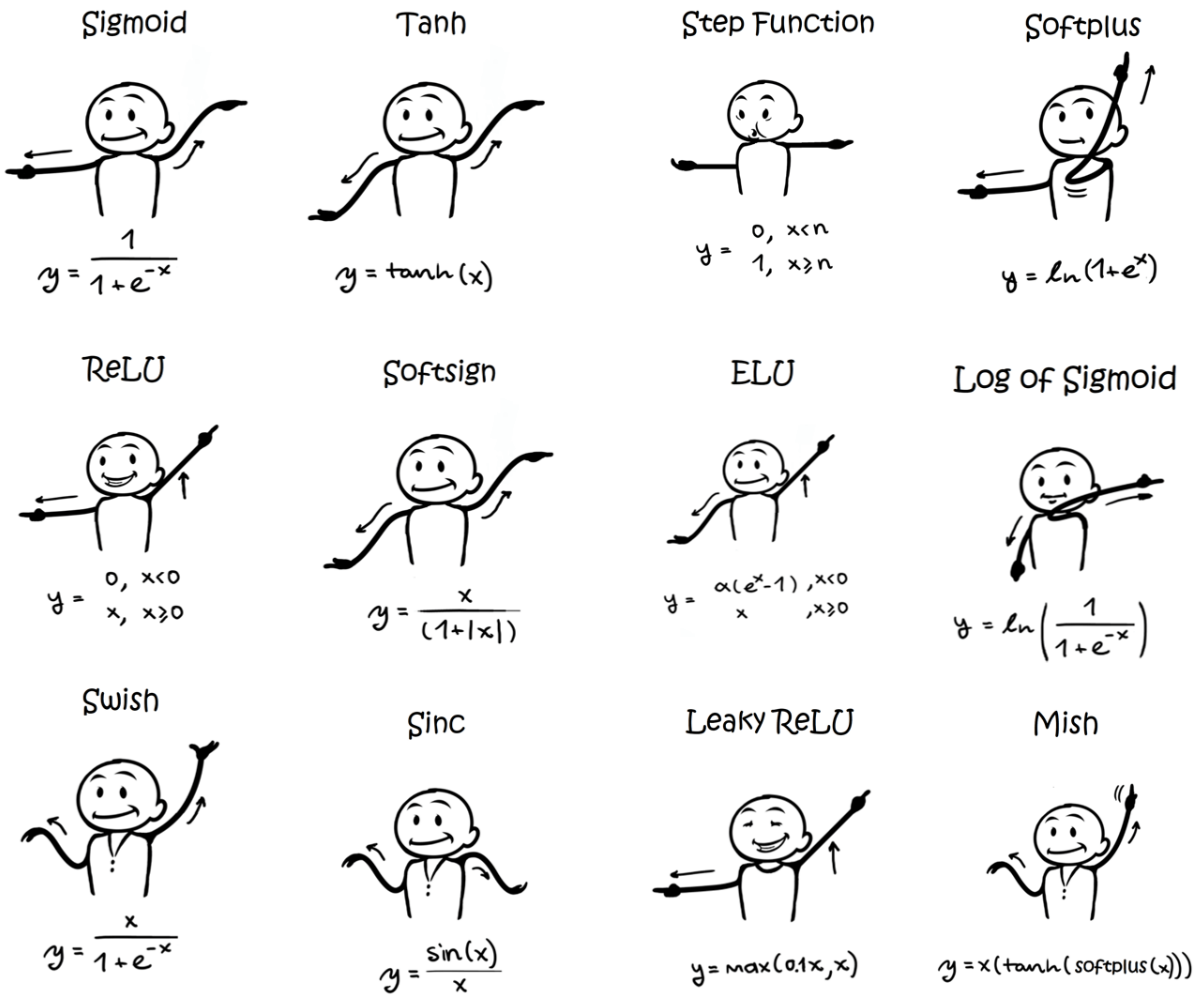


Figure 10-1. Biological neuron[4]

Neural network

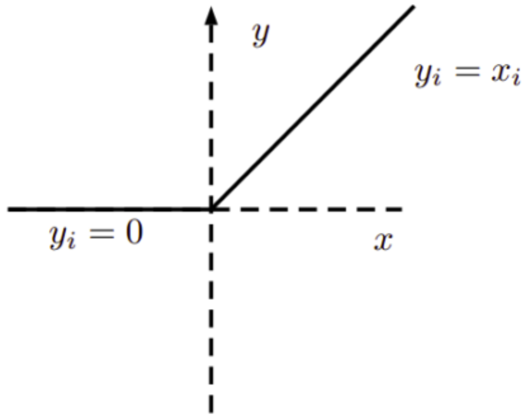
- Activation function
  - What if there is no activation function?



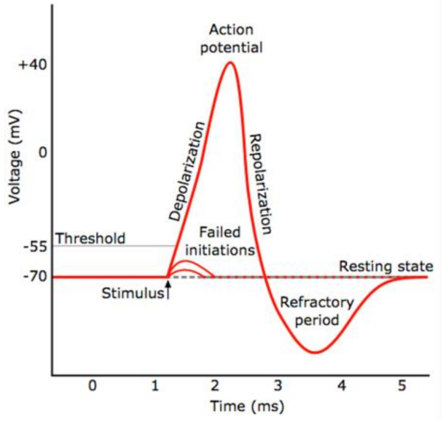
https://bioserver.net/understanding-the-spikes-of-the-neurons/
https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions/

Neural network

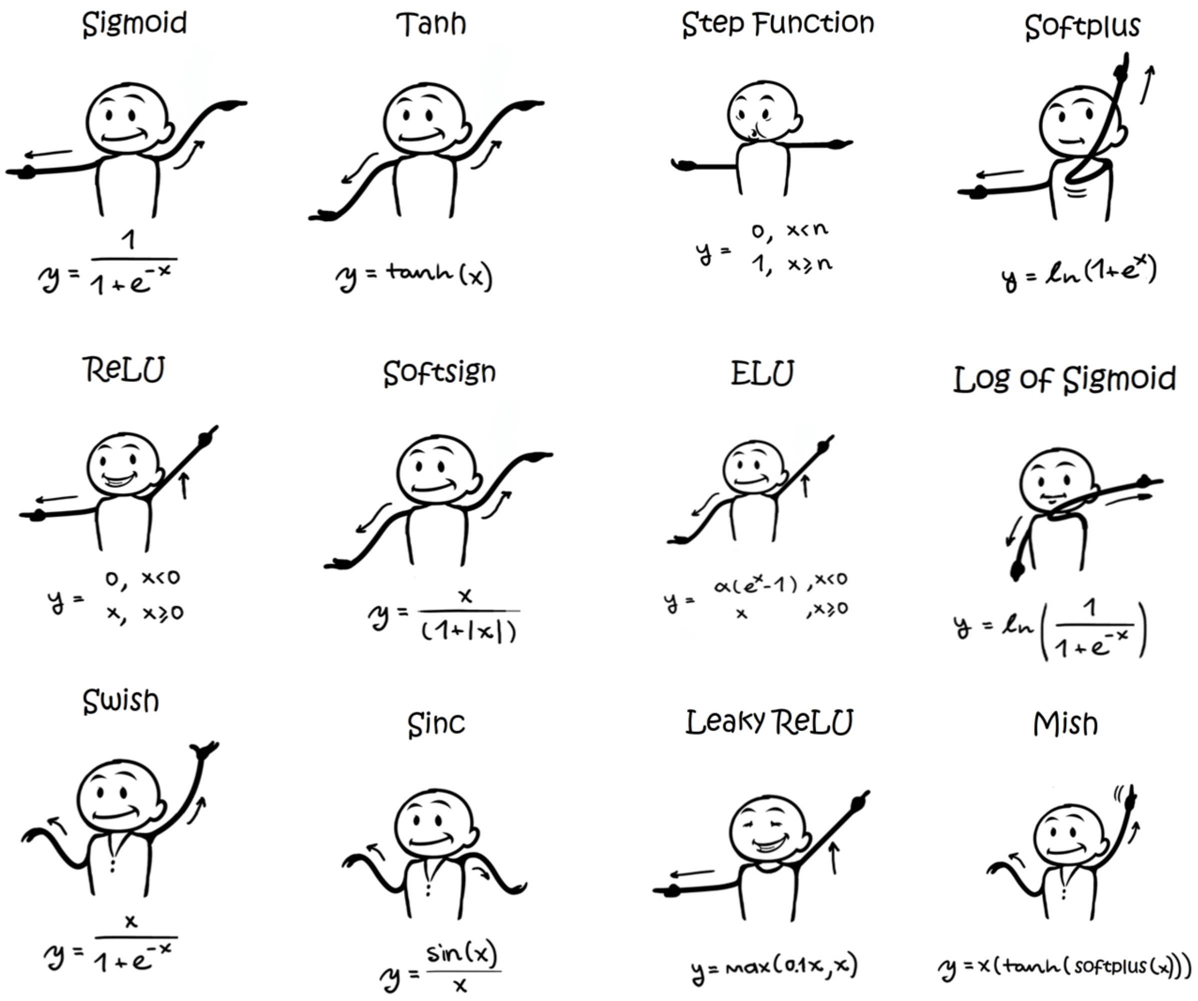
- Activation function
  - ReLU is a good default



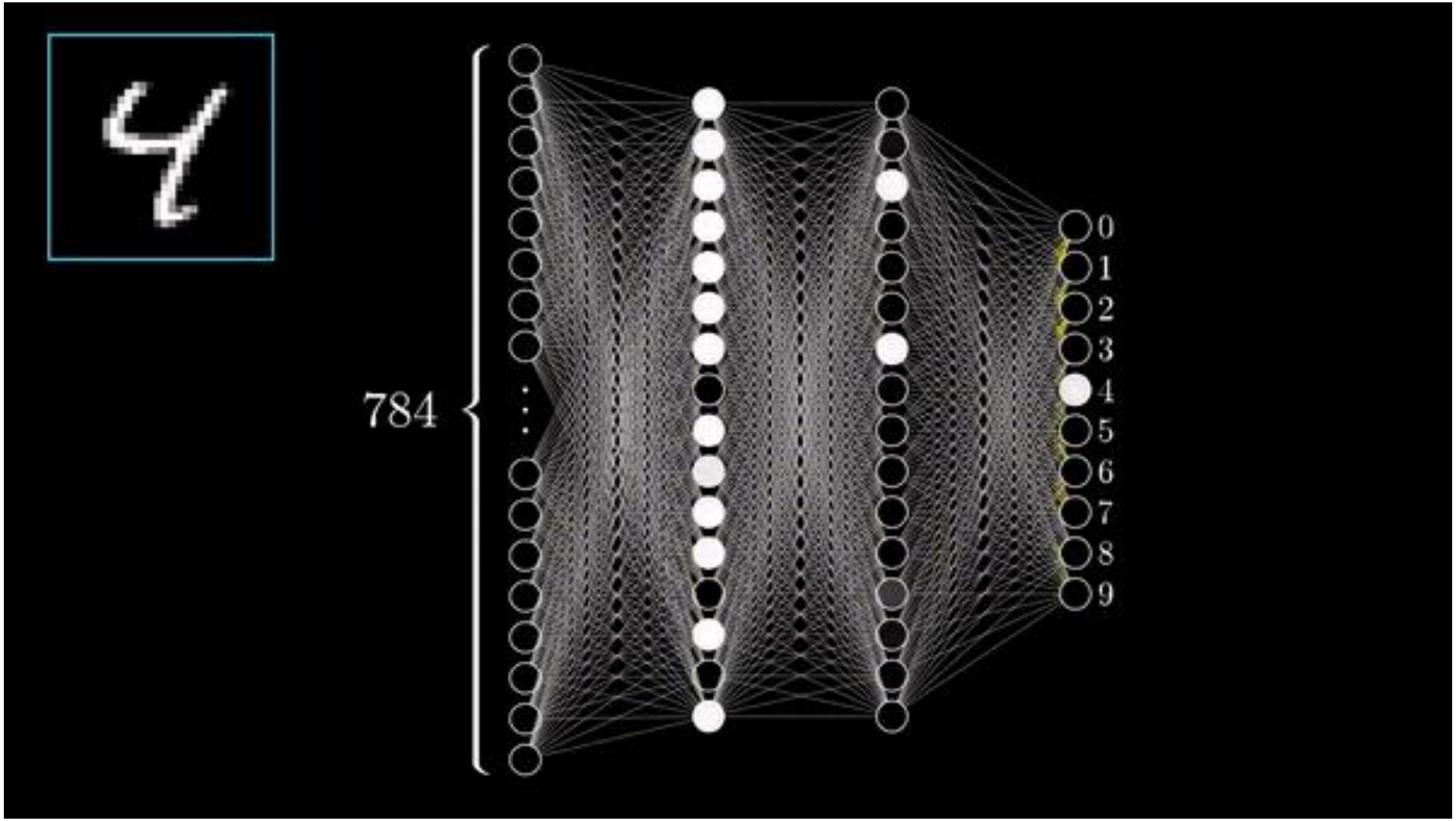
ReLU



Neuron spike



Neural network example (forward propagation)

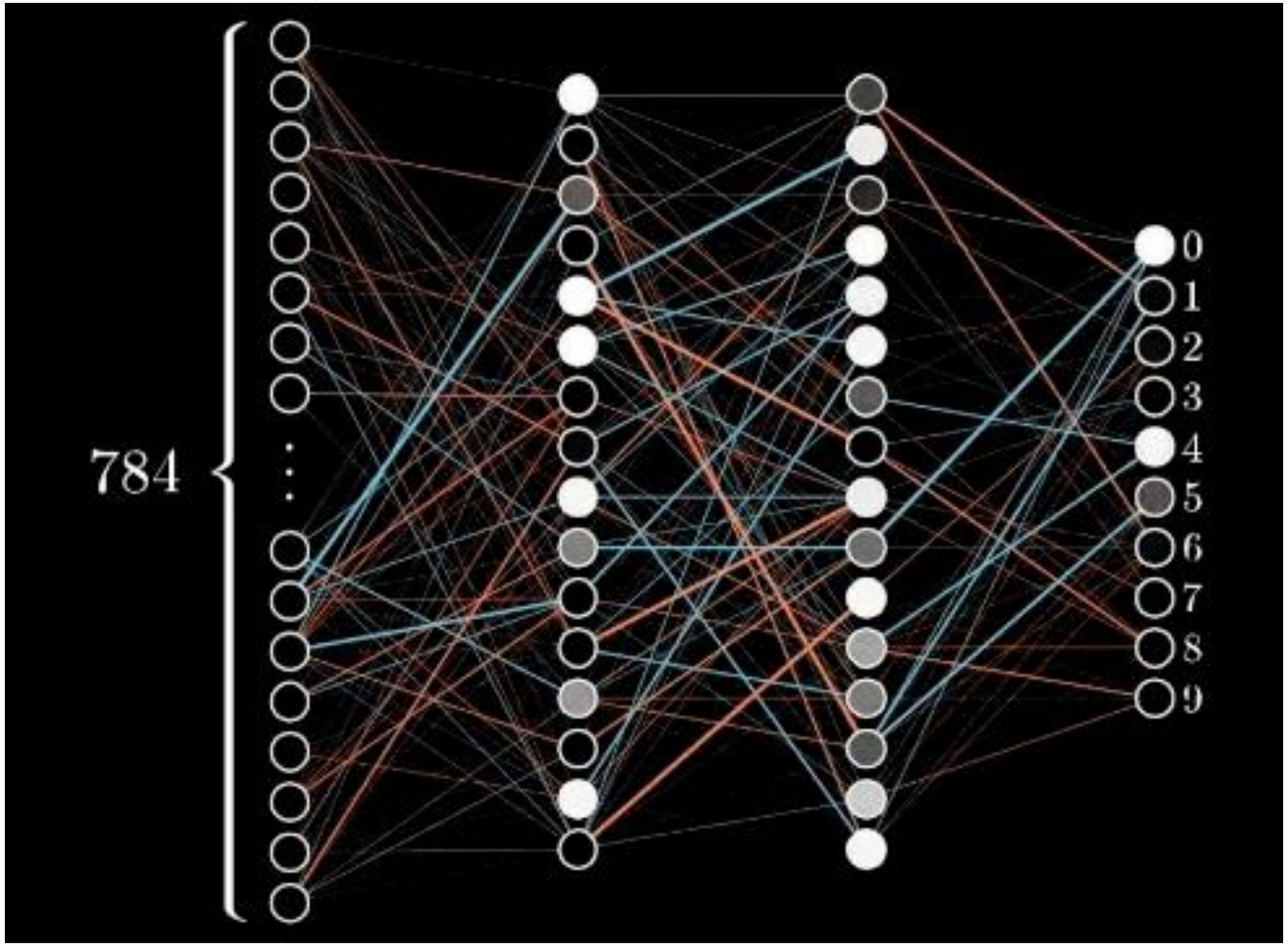


Cross Entropy Loss:

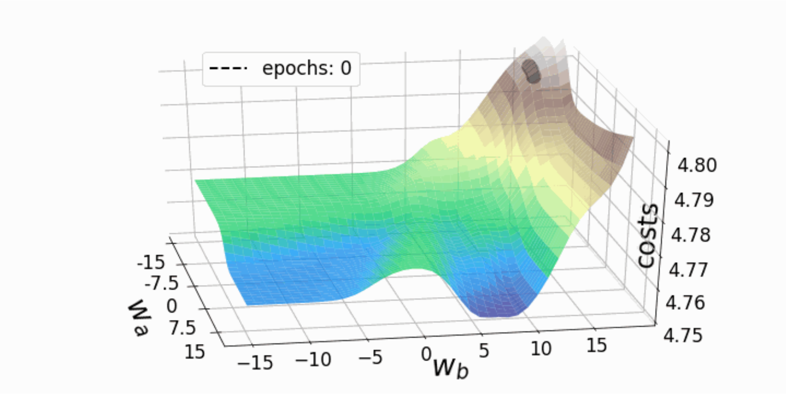
$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

Image from 3Blue1Brown

Neural network example (back propagation)



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

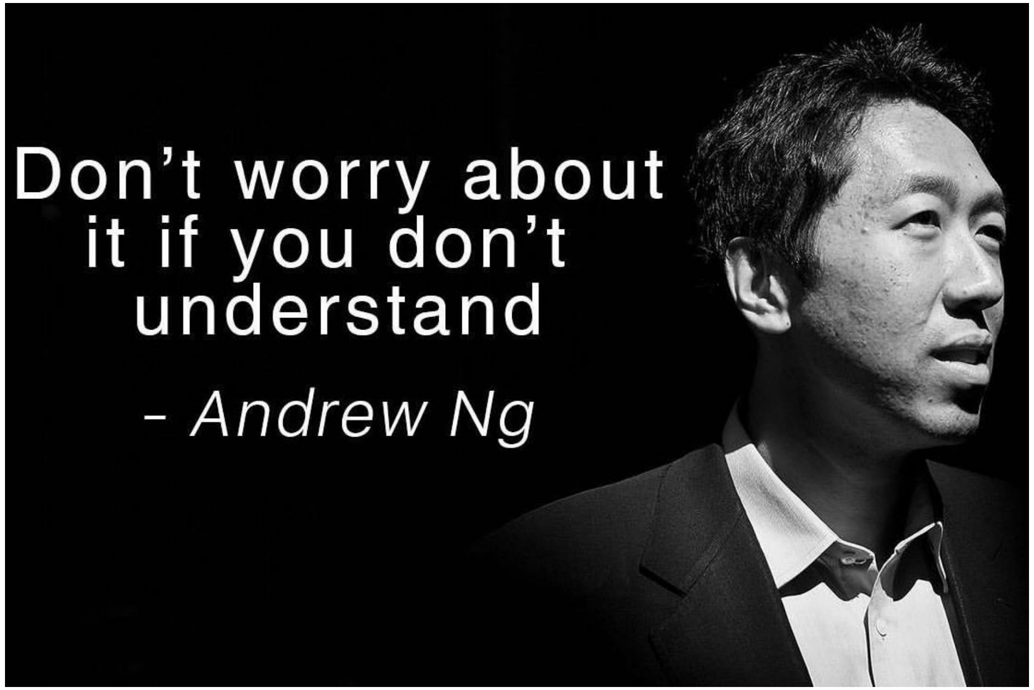


Summary

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	• 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)	• Intuitive, easy to implement • Less assumption restriction • No pre-training is needed	• 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)	• 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)	• Easy to interpret, white box • Require little data preparation • Scale well to large datasets	• Not robust to small variation • May overfit
Random forest	O(k*n*log(n)*d)	O(k*d)	O(k*td)	• Reduce overfitting, improve accuracy • Built-in feature importance	• 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!



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

Q&A