Artificial Intelligence

Machine Learning

Nguyễn Văn Diêu

HO CHI MINH CITY UNIVERSITY OF TRANSPORT

2025

Kiến thức - Kỹ năng - Sáng tạo - Hội nhập

Sứ mệnh - Tầm nhìn

Triết lý Giáo dục - Giá trị cốt lõi

Outline I

Outline II

Outline III

Outline IV

ML Definition

Traditional Programming:

Refers to any manually created program that uses input data and runs on a computer to produce the output.

Machine Learning:

Input and output data are fed to an algorithm to create a program. This program can be used to predict future outcomes.

ML Definition

Definition of **Tom Mitchell** (Machine Learning 1997):

*"A computer program is said to **learn** from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."*

Checkers learning problem:

- Task **T**: Playing checkers.
- Performance measure **P**: Percent of games won against opponents.
- Training experience **E**: Playing practice games against itself.

ML Definition

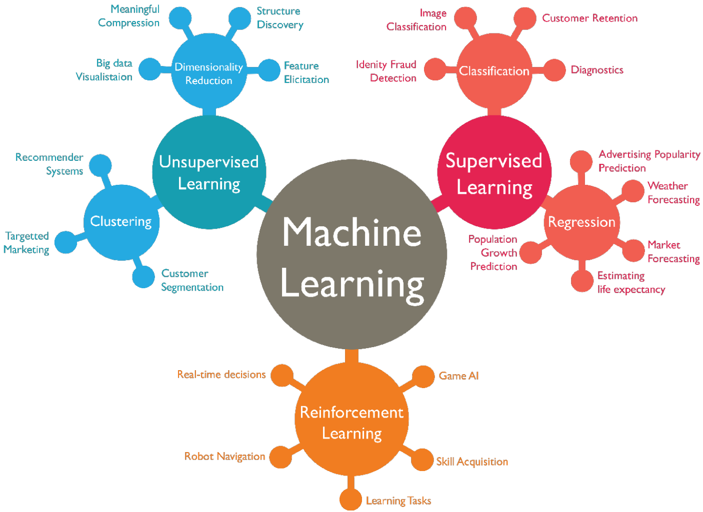
Handwriting recognition learning problem:

- Task **T**: Recognizing and classifying handwritten words within images.
- Performance measure **P**: Percent of words correctly classified.
- Training experience **E**: Database of handwritten words with given classifications.

Robot driving learning problem:

- Task **T**: Driving on public four-lane highways using vision sensors.
- Performance measure **P**: Average distance traveled before an error (as judged by human overseer).
- Training experience **E**: Sequence of images and steering commands recorded while observing a human driver. .

Type of Machine learning



Training data

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}) \dots (x^{(i)}, y^{(i)}) \dots (x^{(n)}, y^{(n)})\}$$

$$x^{(i)} \in \mathbb{R}^d \ , \ y^{(i)} \in \mathbb{R} : \text{label.}$$

$$\begin{array}{c|c|c}
\text{sample } x^{(1)} \rightarrow & (x_1^{(1)} \dots x_j^{(1)} \dots x_d^{(1)}) & y^{(1)} \leftarrow \text{label} \\
\hline
\text{sample } x^{(i)} \rightarrow & (x_1^{(i)} \dots x_j^{(i)} \dots x_d^{(i)}) & y^{(i)} \leftarrow \text{label} \\
\hline
\dots & \dots & \dots \\
\hline
\text{sample } x^{(n)} \rightarrow & (x_1^{(n)} \dots x_j^{(n)} \dots x_d^{(n)}) & y^{(n)} \leftarrow \text{label}
\end{array}$$

fruit	length	width	weight	label
fruit 1	165	38	172	Banana
fruit 2	218	39	230	Banana
fruit 3	76	80	145	Orange
fruit 4	145	35	150	Banana
fruit 5	90	88	160	Orange
...				
fruit n	...	...	...	...

Supervised vs. Unsupervised

fruit	length	width	weight	label
fruit 1	165	38	172	Banana
fruit 2	218	39	230	Banana
fruit 3	76	80	145	Orange
fruit 4	145	35	150	Banana
fruit 5	90	88	160	Orange
...				
fruit n	...	...	...	...

1. **Unsupervised learning**

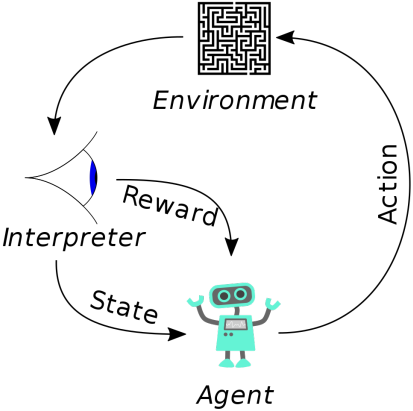
   Learning a model from unlabeled data.

2. **Supervised learning**

   Learning a model from labeled data.

Reinforcement Learning

3. **Reinforcement learning**
   It's concerned with how **intelligent agents** ought to take actions in an **environment** in order to maximize the **cumulative reward**.



ML Workflow

1. **Get Data**

   This process depends on project and data type.

2. **Clean, Prepare & Manipulate Data**

   Real-world data often has unorganized, missing, or noisy elements. Need to clean, prepare, and manipulate the data.

   After getting the data, need to convert the data sets into valid formats for ML platform.

   Finally, split data into training and test data sets. The typical default is a 70/30 split between training and test sets.

3. **Train Model**

   Train data set connects to an algorithm, this mathematical modeling to learn and develop predictions.

ML Workflow

4. **Test Model**

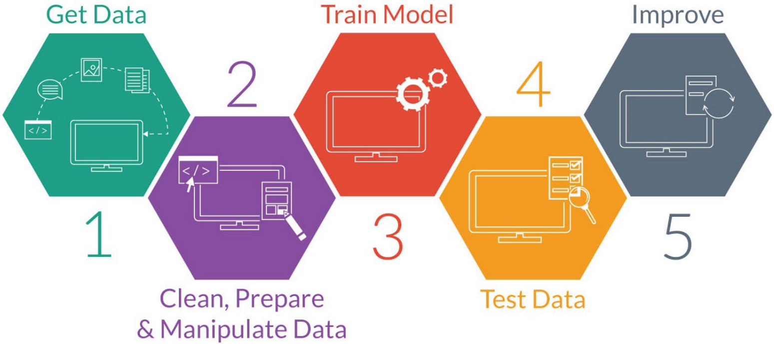
   it's time to validate your trained model. Using the test data, check the model's accuracy.

5. **Improve**

   Refine model and improve accuracy:

   - Review model's results. Are there other data elements worth adding to your model to make it more accurate?
   - Reconsider algorithm choice. A different algorithm may perform better.
   - Adjust the parameters of chosen algorithm to improve performance.

ML Workflow


Get Data
1
2
Clean, Prepare
& Manipulate Data
Train Model
3
4
Test Data
Improve
5

Linear Regression: history

- A very popular technique.
- Rooted in Statistics.
- Method of Least Squares used as early as 1795 by Gauss.
- Re-invented in 1805 by Legendre.
- Frequently applied in astronomy to study the large scale of the universe.
- Still a very useful tool today.

Training data

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}) \dots (x^{(i)}, y^{(i)}) \dots (x^{(n)}, y^{(n)})\}$$

$$x^{(i)} \in \mathbb{R}^d \text{ , } y^{(i)} \in \mathbb{R} : \text{label.}$$

sample $x^{(1)} \rightarrow$	$(x_1^{(1)} \dots x_j^{(1)} \dots x_d^{(1)})$	$y^{(1)} \leftarrow$ label
sample $x^{(i)} \rightarrow$	$(x_1^{(i)} \dots x_j^{(i)} \dots x_d^{(i)})$	$y^{(i)} \leftarrow$ label
...	...	...
sample $x^{(n)} \rightarrow$	$(x_1^{(n)} \dots x_j^{(n)} \dots x_d^{(n)})$	$y^{(n)} \leftarrow$ label

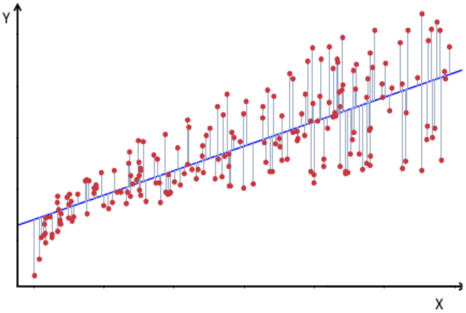
Task Learn a regression function

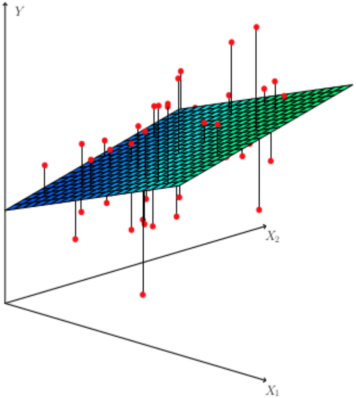
$$f : \mathbb{R}^d \rightarrow \mathbb{R}$$
$$f(x) = y$$

Linear Regression

A regression model is said to be linear if it is represented by a linear function.

Linear Regression





$d = 1$ **Line in** $\mathbb{R}^2$ $d = 2$ **Hyperplane in** $\mathbb{R}^3$

Linear Regression

Linear Regression Model

$$\hat{y} = f(x) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j \qquad \textbf{with } \theta_j \in \mathbb{R},\ j \in \{1, 2, ..., d\}$$

θ: parameters or coefficients or weights.

Learning the linear model $\rightarrow$ learning the θ

Estimation with Least squares

Least square loss

$$loss(y^{(i)}, f(x^{(i)})) = (y^{(i)} - f(x^{(i)}))^2 = (y^{(i)} - \hat{y}^{(i)})^2$$

Minimize the loss over all samples

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Univariate Linear Regression

Model with one feature $d = 1$

$$\hat{y} = f(x) = \theta_0 + \theta_1 x$$

Loss function

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 \quad , \qquad \mathcal{L}(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2$$

Minimize $\mathcal{L}(\theta_0, \theta_1) \rightarrow$ **find** θ_0 **,** θ_1

$$\underset{\{\theta_0,\theta_1\}}{argmin}\ \mathcal{L}(\theta_0, \theta_1) = \underset{\{\theta_0,\theta_1\}}{argmin}\ \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2$$

Find θ_0 **,** θ_1 $\qquad \dfrac{\partial \mathcal{L}}{\partial \theta_0} = 0 \qquad \dfrac{\partial \mathcal{L}}{\partial \theta_1} = 0$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = 2 \times \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \times \frac{\partial}{\partial \theta_0}(y^{(i)} - \theta_0 - \theta_1 x^{(i)})$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \times (-1) = 0$$

$$\boxed{\theta_0 = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} - \theta_1 \frac{1}{n}\sum_{i=1}^{n} x^{(i)}}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = 2 \times \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \times \frac{\partial}{\partial \theta_1}(y^{(i)} - \theta_0 - \theta_1 x^{(i)})$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \times (-x^{(i)}) = 0$$

$$\theta_1 \sum_{i=1}^{n} x^{(i)^2} = \sum_{i=1}^{n} x^{(i)} y^{(i)} - \sum_{i=1}^{n} \theta_0 x^{(i)}$$

Plugging θ_0 in θ_1

$$\boxed{\theta_1 = \frac{\sum_{i=1}^{n} x^{(i)} y^{(i)} - \frac{1}{n}\sum_{i=1}^{n} x^{(i)} \sum_{i=1}^{n} y^{(i)}}{\sum_{i=1}^{n} x^{(i)^2} - \frac{1}{n}\sum_{i=1}^{n} x^{(i)} \sum_{i=1}^{n} x^{(i)}}}$$

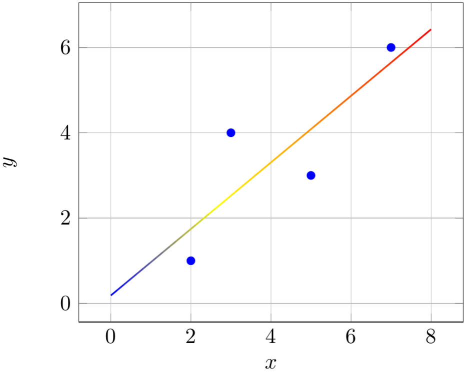
e.g.

– **Training set**

$$\mathcal{D} = \{(2,1),(3,4),(5,3),(7,6)\}$$

– **Linear Regression Model**

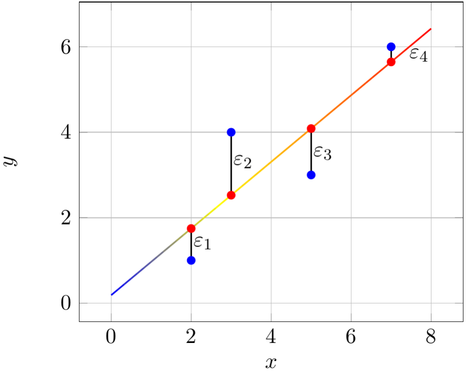
$$f(\theta, x) = \theta_0 + \theta_1 x$$



e.g.

$$\epsilon_i = (y_i, f(\theta, x_i))$$

$$\begin{aligned}\mathcal{L}_i(\theta_0, \theta_1) &= {\epsilon_i}^2 \\ &= (y^{(i)} - f(\theta, x^{(i)}))^2 \\ &= (y^{(i)} - (\theta_0 + \theta_1 x^{(i)}))^2\end{aligned}$$



Multivariate Linear Regression

Model with many feature $d > 1$

$$f(x) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$$

Minimize

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} (y^{(i)} - \theta_0 - \sum_{j=1}^{d} \theta_j x_j^{(i)})^2$$

Matrix with n sample, d feature

X Matrix $n \times (d+1)$

y Label vector

Θ Weights vector

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_j^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(i)} & \cdots & x_j^{(i)} & \cdots & x_d^{(i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_j^{(n)} & \cdots & x_d^{(n)} \end{pmatrix}$$

$$y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(n)} \end{pmatrix} \qquad \Theta = \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_j \\ \vdots \\ \theta_d \end{pmatrix}$$

Normal Equation

$$\mathcal{L}(\Theta) = \frac{1}{2n}\left\|(y - X\Theta)\right\|^2$$

$$\mathcal{L}(\Theta) = \frac{1}{2n}(y - X\Theta)^T(y - X\Theta)$$

$$\frac{\partial \mathcal{L}}{\partial \Theta} = -\frac{1}{n}X^T(y - X\Theta)$$

We have

$$\frac{\partial^2 \mathcal{L}}{\partial \Theta} = \frac{1}{n}X^T X \;:\; \textit{positive}$$

So that is a minimum when

$$X^T(y - X\Theta) = 0$$

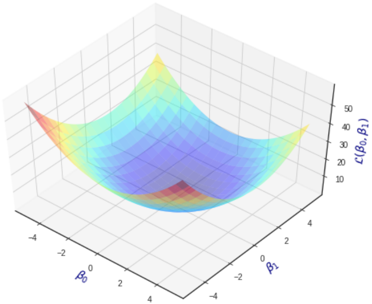
The unique solution

$$\Theta = (X^T X)^{-1} X^T y$$

$$\mathcal{D} \;=\; \{\;(x^{(1)}, y^{(1)}) \ldots (x^{(i)}, y^{(i)}) \ldots (x^{(n)}, y^{(n)})\;\}$$
$$x^{(i)} \in \mathbb{R}^d \;,\; y^{(i)} \in \mathbb{R}.$$

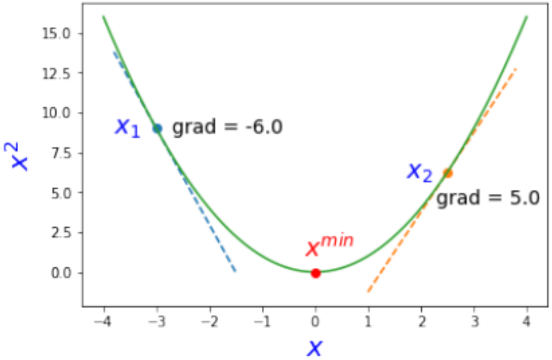
Loss Function $\mathcal{L}(\cdot)$

- n
- $x^{(i)} \;\in\; \mathbb{R}^d$
- $\mathcal{L} \;:\; k$ order



Gradient Descent

$$f(x) = x^2 \quad , \quad f'(x) = 2x$$



Taylor Series Expansion

1. $f(x+\epsilon) = f(x) + \epsilon f^{'}(x) + \frac{\epsilon^2}{2!} f^{''}(x) + \ldots$
2. $f(x+\epsilon) \approx f(x) + \epsilon f^{'}(x)$
3. $\eta > 0 \,,\ \epsilon = -\eta f^{'}(x)$
4. $f(x - \eta f^{'}(x)) \approx f(x) - \eta f^{'}(x)^2$

$$f(\ x - \eta f^{'}(x)\ ) \leq f(x)$$

Gradient descent algo.

1. x_0
2. $x \leftarrow x_0$
3. η : learning rate
4. ϵ : error convergence
5. while $|f^{'}(x)| \geq \epsilon$ do

$$x \leftarrow x - \eta f^{'}(x)$$

Batch Gradient Descent

1. θ_0
2. $\theta \leftarrow \theta_0$
3. η : learning rate
4. ϵ : error convergence
5. while $\|\nabla \mathcal{L}(\theta)\|_2 \geq \epsilon$ do

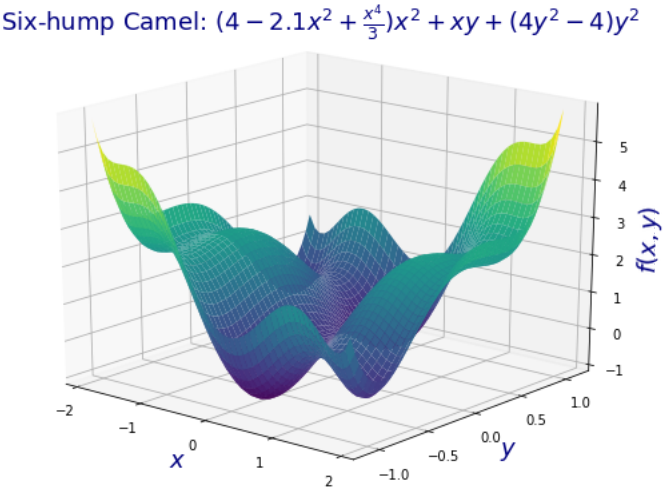
   $$\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta)$$

Stochastic Gradient Descent

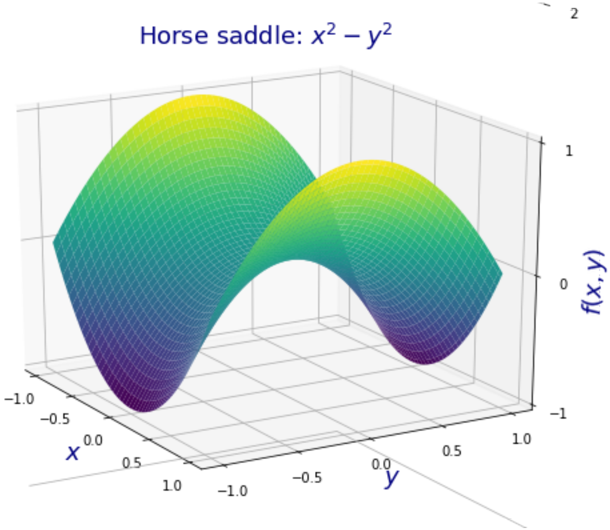
$$\mathcal{L}(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(\theta^T X^{(i)} - y^{(i)})^2$$

$$\mathcal{L}(\theta, X^{(k)}) = \tfrac{1}{2}\,(\theta^T X^{(k)} - y^{(k)})^2$$

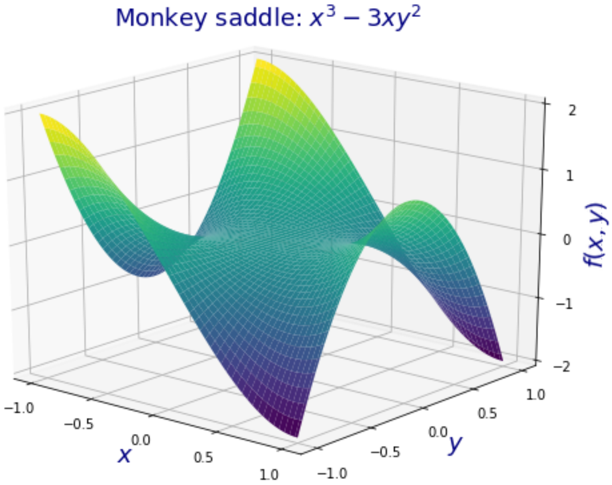
Non-Convex Function



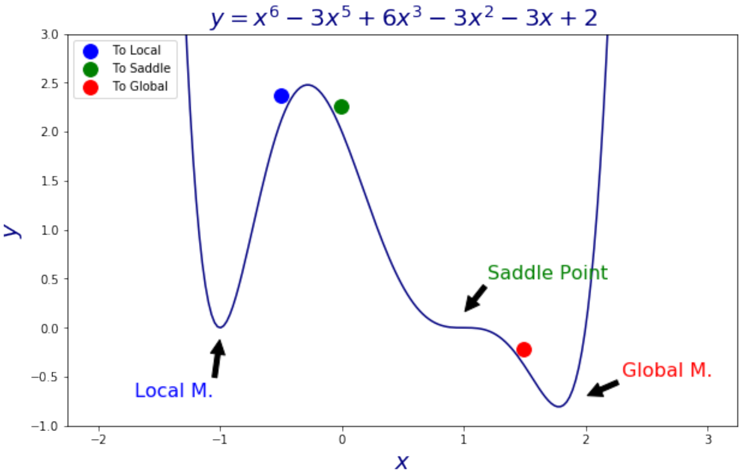
Non-Convex Function



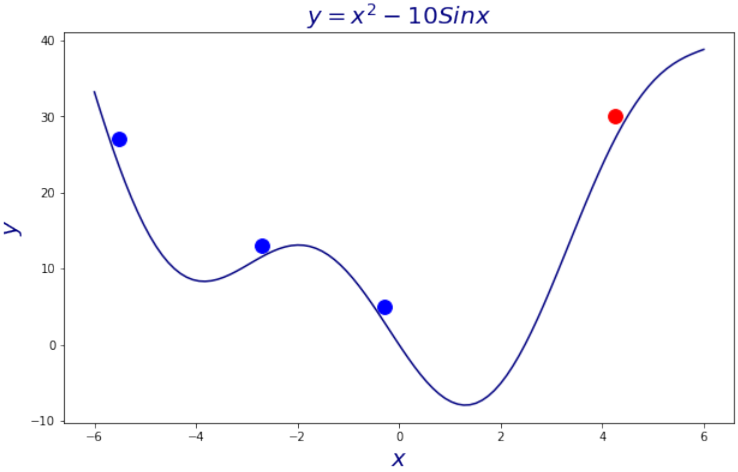
Non-Convex Function



Non-Convex Function



Momentum



Momentum Algorithm

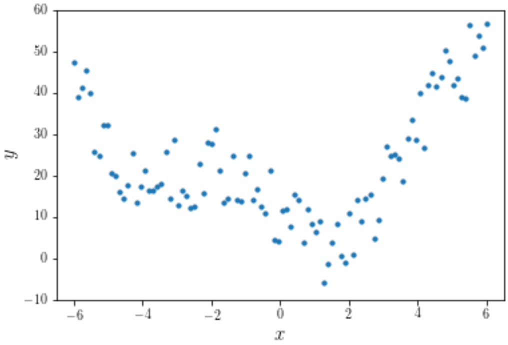
1. $\theta_0 \ ; \ \theta \leftarrow \theta_0$
2. $\eta \ ; \ \epsilon$
3. $v_0 \ ; \ v \leftarrow v_0$
4. γ : momentum
5. while $\|\nabla \mathcal{L}(\theta)\|_2 \geq \epsilon$ do

$$v \leftarrow \gamma v + \eta \nabla \mathcal{L}(\theta)$$
$$\theta \leftarrow \theta - v$$

Polynomial Regression

Sample with $d = 1$

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}) \dots (x^{(i)}, y^{(i)}) \dots (x^{(n)}, y^{(n)})\}$$



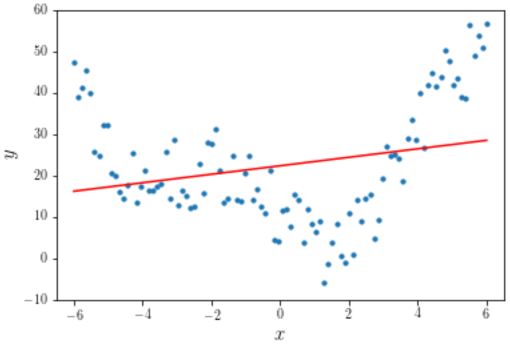
Samples

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}) \ldots (x^{(i)}, y^{(i)}) \ldots (x^{(n)}, y^{(n)})\}$$

$$x^{(i)} \in \mathbb{R} \ , \ \ y^{(i)} \in \mathbb{R}$$

Simple Linear Regression

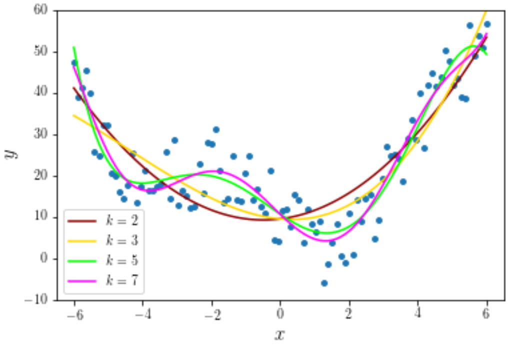
$$f(x) = \theta_0 + \theta_1 x$$



Polynomial Regression

Polynominal k-order

$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k$$



Polynomial Regression

Solving the problem Polynomial Regression is a model used when the response variable is non-linear.

$$f(x) = \theta_0 + \sum_{i=1}^{k} \theta_i x^i$$

Convert to Multivariate Linear Regression

with $d = k$ feature.

$$x_i = x^i$$

$$f(x) = \theta_0 + \sum_{i=1}^{k} \theta_i x_i$$

Multivariate Polynomial Regression

So far we have

- Simple Linear Regression $f(x) = \theta_0 + \theta_1 x$
- Multivariate Linear Regression $f(x) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i$
- Polynomial Regression $f(x) = \theta_0 + \sum_{i=1}^{k} \theta_i x_i^i$

Multivariate Polynomial Regression

- **Second Order**

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_{11} x_1^2 + \theta_{22} x_2^2 + \theta_{12} x_1 x_2$$

- **General Order** Big issue.

To sole this issue, it can mapped to a higher order space of independent variables called as the feature space (eg. Kernel method ...)

Logistic Regression for Classification

Logistic model (Logit model): Probability of a class label in dataset.

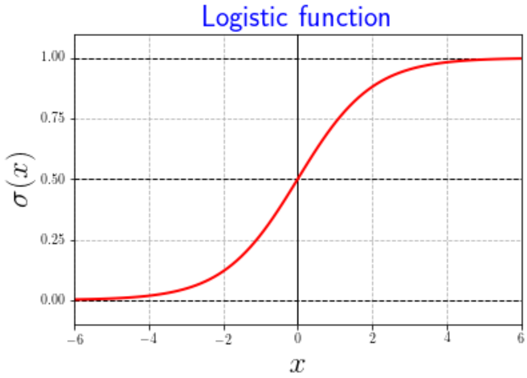
Logistic function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$x \to +\infty \ , \ \sigma(x) \to 1$

$x \to -\infty \ , \ \sigma(x) \to 0$

– *Continuous, has a first derivative.*



Probability return

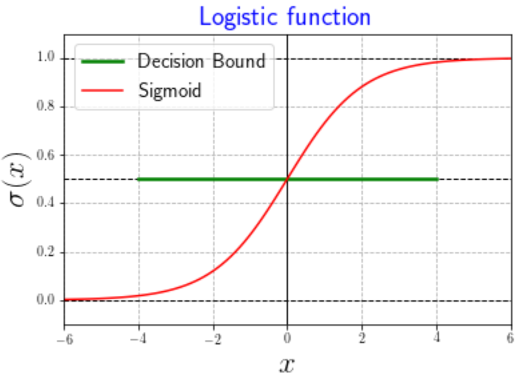
Logistic return probability score between 0 and 1

$$Prob = \sigma(x) = \frac{1}{1 + e^{-x}}$$

$Prob \in (0, 1)$

$Prob \geq 0.5 \; , \; class = 1$

$Prob < 0.5 \; , \; class = 0$



Logistic Regression Model

$$\mathcal{D} = \left\{ \left(x^{(i)}, y^{(i)} \right) \right\}_1^n, \quad x^{(i)} \in \mathbb{R}^d, \quad y^{(i)} \in \{0, 1\}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \in \mathbb{R}^{d+1} \qquad x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

Model

$$f(x) = \sigma(\theta.x) = \frac{1}{1 + e^{-\theta.x}}$$

dot product $\theta.x \equiv \sum_j \theta_j x_j$

Logistic Regression Model

e.g.

$x = (x_1, x_2)$, $y = \{0, 1\}$. imagine we know θ.

$z = \theta.x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

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

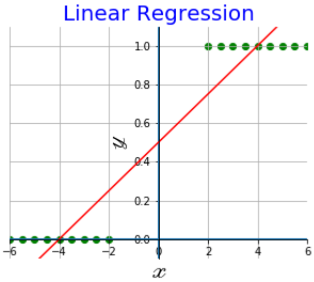
Decision boundary = .5

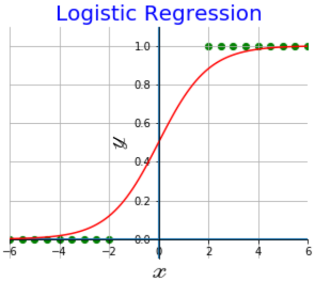
e.g. $Prob(class = 1)$ *return .4, only have 40% chance "class 1" or this observation as "class 0".*

Linear Regression Vs. Logistic Regression

- Linear Regression: *Continuous output.*
- Logistic Regression: *Constant output.*
- Linear Regression: *Using Ordinary Least Squares (OLS).*
- Logistic Regression: *Using Maximum Likelihood Estimation (MLE).*

Consider dataset which two class $\{0, 1\}$





Logistic Regression Model

Model

$$f(x) = \sigma(\theta.x) = \frac{1}{1 + e^{-\theta.x}}$$

$f(x) \in (0, 1)$

$f(x)$: Probability.

Probability return by Model

$Prob(y = 1 \mid x; \theta) = f(x)$

$Prob(y = 0 \mid x; \theta) = 1 - f(x)$

Model base on probability

$$Prob(y \mid x; \theta) = f(x)^y \, (1 - f(x))^{1-y}$$

The Likelihood

One sample i

$$Prob(y^{(i)} \mid x^{(i)}; \theta) = f(x^{(i)})^{y^{(i)}} \; (1 - f(x^{(i)}))^{1-y^{(i)}}$$

Loss function $Loss(\theta)$ for all sample.

n training samples were generated independently.

Likelihood

$$Loss(\theta) = \prod_{i=1}^{n} Prob(y^{(i)} \mid x^{(i)}; \theta)$$

$$Loss(\theta) = \prod_{i=1}^{n} f(x^{(i)})^{y^{(i)}} \; (1 - f(x^{(i)}))^{1-y^{(i)}}$$

Logarithm of Likelihood

- Logarithm turns a product into a sum.
- It avoid the issue of small number(typically for probability).

$$
\begin{aligned}
\mathcal{L}(\theta) &= log\ Loss(\theta) \\
&= log \prod_{i=1}^{n} f(x^{(i)})^{y^{(i)}} \ (1 - f(x^{(i)}))^{1-y^{(i)}} \\
&= \sum_{i=1}^{n} log \left\{ f(x^{(i)})^{y^{(i)}} \ (1 - f(x^{(i)}))^{1-y^{(i)}} \right\} \\
&= \sum_{i=1}^{n} \left\{ log f(x^{(i)})^{y^{(i)}} \ + \ log(1 - f(x^{(i)}))^{1-y^{(i)}} \right\} \\
&= \sum_{i=1}^{n} \left\{ y^{(i)} \ log f(x^{(i)}) \ + \ (1 - y^{(i)}) \ log(1 - f(x^{(i)})) \right\}
\end{aligned}
$$

Maximization into a Minimization

Maximize the Likelihood

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \left\{ y^{(i)}\ log f(x^{(i)}) \ + \ (1 - y^{(i)})\ log(1 - f(x^{\ (i)})) \right\}$$

– Negative log-likelihood (NLL).

– Use gradient descent algo.

So we minimize the negative $\mathcal{L}(\theta)$ with

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \left\{ y^{(i)}\ log f(x^{(i)}) \ + \ (1 - y^{(i)})\ log(1 - f(x^{(i)})) \right\}$$

Minimize the negative Likelihood

$$\boxed{\mathcal{L}(\theta) = \sum_{i=1}^{n} \left\{ -y^{(i)} \; log f(x^{(i)}) \; - \; (1 - y^{(i)}) \; log(1 - f(x^{(i)})) \right\}}$$

$$\frac{\partial}{\partial \theta_j}\mathcal{L}(\theta) = \sum_{i=1}^{n} \left\{ -y^{(i)} \frac{1}{\sigma(\theta.x^{(i)})} + (1 - y^{(i)}) \frac{1}{1 - \sigma(\theta.x^{(i)})} \right\} \times \frac{\partial}{\partial \theta_j}\sigma(\theta.x^{(i)})$$

since $\frac{\partial}{\partial \theta_j}\sigma(\theta.x^{(i)}) = \sigma(\theta.x^{(i)})(1 - \sigma(\theta.x^{(i)}))\frac{\partial}{\partial \theta_j}\theta.x^{(i)}$, so

$$= \sum_{i=1}^{n} \left\{ \sigma(\theta.x^{(i)}) - y^{(i)} \right\} \frac{\partial}{\partial \theta_j}\theta.x^{(i)}$$

Since $\theta.x^{(i)} = \theta_0 + \sum_{j=1}^{d} \theta_j x_j^{(i)} \implies \frac{\partial}{\partial \theta_j}\theta.x^{(i)} = x_j^{(i)}$, so

$$= \sum_{i=1}^{n} (f(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Types of Logistic Regression

- **Binary Logistic Regression** Binary class, e.g. Spam or Not Spam, Cancer or No Cancer.
- **Multinomial Logistic Regression** Many class, e.g. predicting the type of Wine.
- **Ordinal Logistic Regression** Many ordinal class, e.g. restaurant or product rating from 1 to 5.

Classification

$$\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_1^n, \quad x^{(i)} \in \mathbb{R}^d \quad , \quad y^{(i)} : \textbf{discrete}.$$

Binary Classification

$$y^{(i)} \in \{0, 1\} \; or \; y^{(i)} \in \{-1, 1\}$$

e.g.

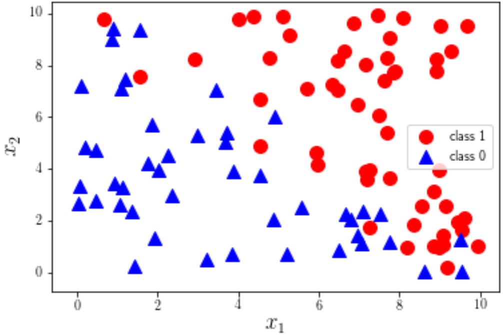
$$\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_1^n, \; x^{(i)} \in \mathbb{R}^2, \; y^{(i)} \in \{0, 1\}$$

x_1	x_2	y
8.625	0.058	0
3.828	0.723	0
7.150	3.899	1
6.477	8.198	1
1.922	1.331	0

e.g. Binary Classification

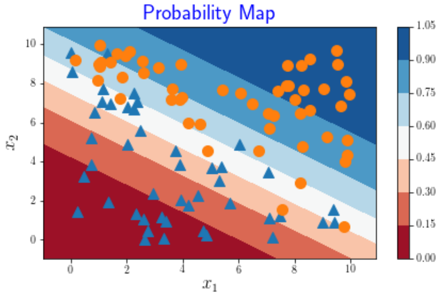
e.g. chart of dataset above

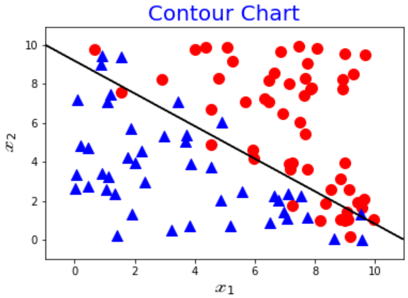
$$\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_1^n,\ x^{(i)} \in \mathbb{R}^2,\ \ y^{(i)} \in \{0, 1\}$$



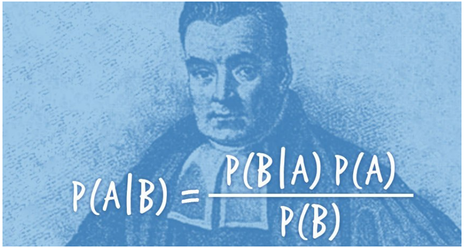
e.g. Logistic Regression

- This model predict probability of two class:
- Contour chart base on probability.





Bayes's Theorem



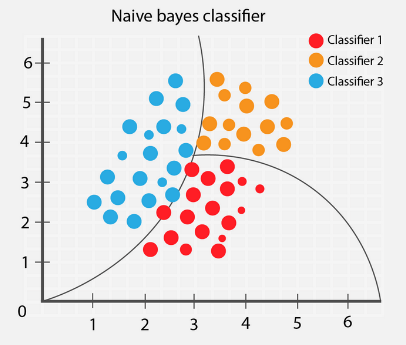
$p(A)$: *Prior*
$p(A|B)$: *Posterior*
$p(B|A)$: *Likelihood*
$p(B)$: *Evidence*

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

$$\boxed{P(class/data) = \frac{P(data/class) \times P(class)}{P(data)}}$$

Naïve Bayes Classifier

$$P(class/data) = \frac{P(data/class) \times P(class)}{P(data)}$$



Review: Joint Probability

Tossing two coins: Independent

- A: Means the first coin lands face up
- B: Means the second coin lands face up
  - $p(A) = p(B) = 0.5$
  - $p(A \ and \ B) = p(A) \ p(B) = 0.25$
  - $p(B|A) = p(B)$

Events are not independent

- A: Mean it rains today
- B: Means it rains tomorrow
  - It rained today, it more likely rain tomorrow
  - $p(B|A) > p(B)$
  - $p(A \ and \ B) = p(A) \ p(B|A)$

e.g. Cookie problem

– *Suppose there are two bowls of cookies*
  + Bowl 1:
    30 vanilla
    10 chocolate
  + Bowl 2:
    20 vanilla
    20 chocolate

– *Now suppose you choose*
  + One of the bowls at random
  + Without looking, select a cookie at random

This is a conditional probability

$$p(Bowl\ 1 \mid vanilla)$$

$p(vanila \mid Bowl\ 1) = 3/4$
$\neq p(Bowl\ 1 \mid vanilla)$

Bayes's Theorem

Any events A and B

– $p(A \text{ and } B) = p(B \text{ and } A)$

– $p(A \text{ and } B) = p(A)\ p(B|A)$

– $p(B \text{ and } A) = p(B)\ p(A|B)$

$\Longrightarrow\ p(B)\ p(A|B) = p(A)\ p(B|A)$

Bayes's Theorem

$$\boxed{p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}}$$

Cookie problem

+ $B1$: Hypothesis of cookie came from Bowl 1
+ V: Vanilla cookie

$$p(B1|V) = \frac{p(V|B1)\ p(B1)}{p(V)}$$

e.g. Cookie problem

$$p(B1|V) = \frac{p(V|B1)\ p(B1)}{p(V)}$$

$p(B1)$: Probability chose Bowl 1

$$p(B1) = 1/2$$

$p(V|B1)$: Probability vanilla cookie from Bowl 1

$$p(V|B1) = 3/4$$

$p(V)$: Probability vanilla cookie from either bowl

$$p(V) = 5/8$$

$$p(B1|V) = \frac{(3/4)\ (1/2)}{(5/8)} = 3/5$$

e.g. Elderly Fall and Death

- Elderly person is died: 10%
- Elderly people falling: 5%
- All elderly people die, they had fall: 7%

Probability that elderly people die when they fall?

$$P(Die|Fall) = \frac{P(Fall|Die) \times P(Die)}{P(Fall)}$$

$P(Die) = 0.10$
$P(Fall) = 0.05$
$P(Fall|Die) = 0.07$

$$P(Die|Fall) = \frac{0.07 \times 0.10}{0.05}$$

$$P(Die|Fall) = \mathbf{0.14}$$

- If an elderly person falls
- There is a 14% probability that they will die from the fall

e.g. Email and Spam Detection

- Email receive is spam: 2%
- Spam detector accuracy: 99%
- When an email is not spam, it will mark it as spam: 0.1%

Probability that fact spam email in spam folder?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(B) = P(B|A) \times P(A) \ + \ P(B|not\ A) \times P(not\ A)$$

e.g. Email and Spam Detection

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(B) = P(B|A) \times P(A) \; + \; P(B|not\ A) \times P(not\ A)$$

$$\begin{aligned}
P(A|B) &= P(Spam|Detected) = ? \\
P(B|A) &= P(Detected|Spam) = 0.99 \\
P(A) &= P(Spam) = 0.02 \\
P(not\ A) &= 1 - P(Spam) = 0.98 \\
P(B|not\ A) &= P(Detected|not\ Spam) = 0.001
\end{aligned}$$

$$P(Spam|Detected) = \frac{0.99 \times 0.02}{0.99 \times 0.02 + 0.001 \times 0.98} = 0.952$$

– *Probability fact spam email in spam folder, is* **95.2%**.

e.g. Liars and Lie Detectors

- Lie Detector test persons: *if positive result* $\Longrightarrow$ *they are lying*.
- People are tested:
  - + Telling the truth: 98%
  - + Liars: 2%
- Liar people is tested: positive result 72%
- When the machine says they are *not lying*: this is true 97%

Probability that they are indeed lying?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(B) = P(B|A) \times P(A) \; + \; P(B|not\ A) \times P(not\ A)$$

e.g. Liars and Lie Detectors

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(B) = P(B|A) \times P(A) \; + \; P(B|not\ A) \times P(not\ A)$$

$$\begin{aligned}
P(A|B) &= P(Lying|Positive) = ? \\
P(B|A) &= P(Positive|Lying) = 0.72 \\
P(A) &= P(Lying) = 0.02 \\
P(not\ A) &= 1 \text{ - } P(Lying) = 0.98 \\
P(not\ B|not\ A) &= P(not\ Positive|not\ Lying) = 0.097 \\
P(B|not\ A) &= 1 \text{ - } P(not\ B|not\ A) = 1 \text{ - } 0.097 = 0.03
\end{aligned}$$

$$P(Lying|Positive) = \frac{0.72 \times 0.02}{0.72 \times 0.02 + 0.03 \times 0.98} = 0.328$$

– *Probability fact lying when positive test result, is* **32.8%**.

e.g. Medical test

– People have a certain genetic defect: 1% – Testing to genetic defect (true positives): 90% – Testing have false positives: 9.6%

Probability genetic defect when get a positive test result?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B|A) \times P(A) + P(B|not\ A) \times P(not\ A)}$$

$$\begin{aligned}
P(A|B) &= P(GeneticDefect|Positive) = ?\\
P(B|A) &= P(Positive|GeneticDefect) = 0.9\\
P(A) &= P(GeneticDefect) = 0.01\\
P(not\ A) &= 1 \text{ - } P(GeneticDefect) = 0.99\\
P(B|not\ A) &= P(Positive|not\ GeneticDefect) = 0.096
\end{aligned}$$

$$P(GeneticDefect|Positive) = \frac{0.9 \times 0.01}{0.9 \times 0.01 + 0.096 \times 0.99} = 0.0865$$

– *Probability faulty gene on positive result, is* **8.65%**.

e.g. Breast Cancer test

– Women over 50 have breast cancer: 1% – Women who have breast cancer, had positive result test: 90% – Women will have false positives: 8%

Probability woman has cancer if she has a positive result?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B|A) \times P(A) + P(B|not\ A) \times P(not\ A)}$$

$$\begin{aligned}
P(A|B) &= P(Cancer|Positive) = ? \\
P(B|A) &= P(Positive|Cancer) = 0.9 \\
P(A) &= P(Cancer) = 0.01 \\
P(not\ A) &= 1 \text{ - } P(Cancer) = 0.99 \\
P(B|not\ A) &= P(Positive|not\ Cancer) = 0.08
\end{aligned}$$

$$P(Cancer|Positive) = \frac{0.9 \times 0.01}{0.9 \times 0.01 + 0.08 \times 0.99} = 0.10$$

– *Probability cancer, given a positive test result, is* **10%**.

Naïve Bayes Classifier

Given dataset $D = \{ \ (x^{(i)}, y^{(i)}) \ \}_{i=1}^{n}$, $x^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in C$

given (x, y) , $x \in \mathbb{R}^d$, find $y \in C$, with maximum $p(y|x)$

$$p(y|x) = \frac{p(y) \ p(x|y)}{p(x)}$$

- $p(y)$: Prior probability of class y in dataset D
  (we have C class)
- $p(y|x)$: Posterior probability of class y given **one** evidence
  $x = (x_1, x_2, ..., x_d)$
- $p(x|y)$: Likelihood which is the probability of evidence given
  class $y \in C$
- $p(x)$: Prior probability of **one** evidence in D
  $x = (x_1, x_2, ..., x_d)$

Naïve Bayes Model

$$p(y|x_1, x_2, ..., x_d) = \frac{p(y)\ p(x_1, x_2, ..., x_d|y)}{p(x_1, x_2, ..., x_d)}$$

$(x_1, x_2, ..., x_d)$ are stochastically independent, given y:

$$p(x_1, x_2, ..., x_d|y) = p(x_1|y)\ p(x_2|y)\ ...\ p(x_d|y)$$

$$p(y|x_1, x_2, ..., x_d) = \frac{p(y) \prod_{i=1}^{d} p(x_i|y)}{p(x_1, x_2, ..., x_d)}$$

$p(x_1, x_2, ..., x_d)$ is constant given the Dataset,

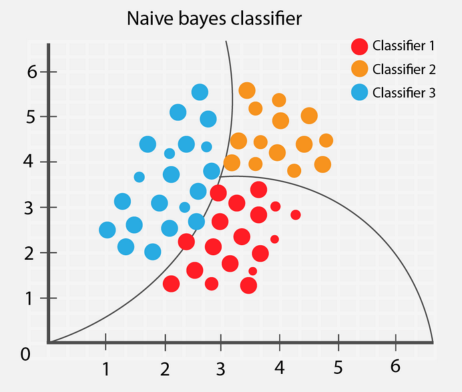
$$p(y|x_1, x_2, ..., x_d) \ \propto \ p(y) \prod_{i=1}^{d} p(x_i|y)$$

Algorithm

$$\hat{y} = \underset{y\ \in\ C}{argmax}\ p(y) \prod_{i=1}^{d} p(x_i|y)$$

Naïve Bayes Classifier algorithm

$$\hat{y} = \underset{y \in C}{argmax}\ p(y) \prod_{i=1}^{d} p(x_i|y)$$



e.g.

Outlook	Temperature	Huminity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Give a new instance x :

Outlook	Temperature	Huminity	Windy	Play
sunny	*cool*	*high*	*true*	**?**

y = **yes**

$p(yes) = \frac{9}{14}$
$p(sunny|yes) = \frac{2}{9}$
$p(cool|yes) = \frac{3}{9}$
$p(high|yes) = \frac{3}{9}$
$p(true|yes) = \frac{3}{9}$
$p(y = yes) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}$
$= 0.00529$

y = **no**

$p(no) = \frac{5}{14}$
$p(sunny|no) = \frac{3}{5}$
$p(cool|no) = \frac{1}{5}$
$p(high|no) = \frac{4}{5}$
$p(true|no) = \frac{3}{5}$
$p(y = no) = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}$
$= 0.02057$

$p(y = no) > p(y = yes)$. Predict result:

Outlook	Temperature	Huminity	Windy	Play
sunny	*cool*	*high*	*true*	**no**

Probability Distribution

– A probability distribution is a statistical function that describes all the possible values that a random variable can take within a given range.

– Probability distribution depends on factors:

Mean expected value of the distribution.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance describes the spread of observation from the mean.

Population: $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x-\mu)^2$ **Sample:** $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x-\mu)^2$

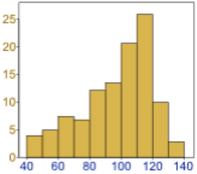
Standard deviation describes the normalized spread of observations from the mean.

σ

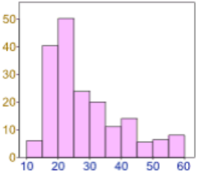
Probability Distribution

Data can be "distributed" (spread out) in different ways

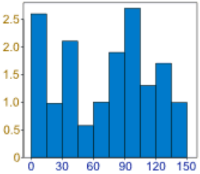
It can be spread out more on the left



Or more on the right



Or it can be all jumbled up



Normal Distribution

Many cases, data tends to be around a central value

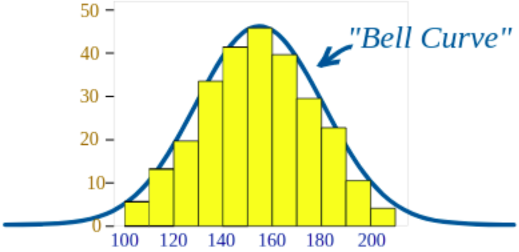
σ : Standard deviation of x

μ : Mean of x

$\pi \approx 3.14159...$

$e \approx 2.71828...$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



Normal Distribution

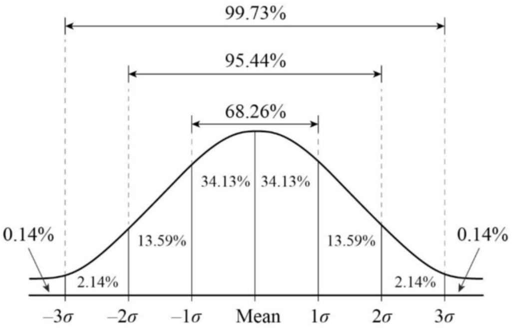
Many things closely follow a Normal Distribution:

- Heights of people
- Size of things produced by machines
- Errors in measurements
- Blood pressure
- Marks on a test

Normal Distribution

The 68 - 95 - 99.7 Rule

- Mean μ
- Standard deviation σ
- Approximately 68% observations fall within σ of the mean μ.
- Approximately 95% observations fall within 2σ of σ.
- Approximately 99.7% observations fall within 3σ of σ.



Gaussian Naïve Bayes

If features are continuous values, the likelihood of the features is assumed to be Gaussian:

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

Algorithm

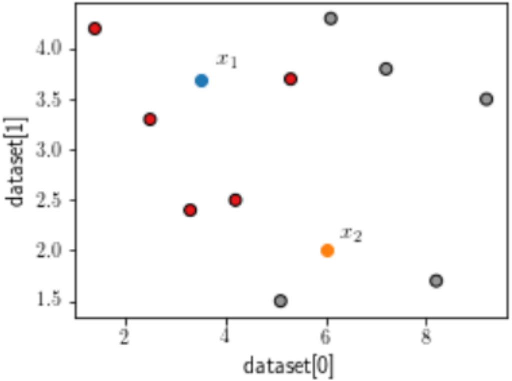
$$\boxed{\hat{y} = \underset{y \in C}{argmax}\ p(y) \prod_{i=1}^{d} p(x_i|y)}$$

Very easy!

e.g. Gaussian Naïve Bayes Classifier

dataset:

$$
\begin{aligned}
&(3.3\ ,\ 2.4\ ,\ 0),\\
&(2.5\ ,\ 3.3\ ,\ 0),\\
&(4.2\ ,\ 2.5\ ,\ 0),\\
&(1.4\ ,\ 4.2\ ,\ 0),\\
&(5.3\ ,\ 3.7\ ,\ 0),\\
&(6.1\ ,\ 4.3\ ,\ 1),\\
&(9.2\ ,\ 3.5\ ,\ 1),\\
&(7.2\ ,\ 3.8\ ,\ 1),\\
&(5.1\ ,\ 1.5\ ,\ 1),\\
&(8.2\ ,\ 1.7\ ,\ 1)
\end{aligned}
$$



$$x_1 = (3.5\ ,\ 3.7)$$
$$x_2 = (6\ ,\ 2)$$

	$y=0$	$y=1$
feature 1		
μ	3.34	7.16
σ	1.50	1.63
feature 2		
μ	3.22	2.96
σ	0.77	1.28

$x_1 = (3.5\ ,\ 3.7)$

$y = 0$

$$p(x_1|y) = p(y) \times p(x_{10} = 3.5|y) \times p(x_{11} = 3.7|y) = 0.5 \times 0.26 \times 0.43 = 0.056$$

$y = 1$

$$p(x_1|y) = p(y) \times p(x_{10} = 3.5|y) \times p(x_{11} = 3.7|y) = 0.5 \times 0.02 \times 0.26 = 0.026$$

So, predict Class of x_1 is $y = 0$

Multinomial Naïve Bayes

Used in text classification

$\{(D, y)\}$, D: Document , y Class (Category of D).

$\theta_y = (\theta_{y1}, ..., \theta_{yd})$ The distribution vector for class y
d : number of feature = size of the vocabulary.

$\theta_{yi} = P(x_i \mid y)$ The probability of feature i appearing in a sample belonging to class y.

θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha d}$$

$N_{yi} = \sum_{x \in T} x_i$: number of times feature i appears in a sample of class y in the training set T.
$N_y = \sum_{i=1}^{d} N_{yi}$: total count of all features for class y.
$\alpha \geqslant 0$ prevents zero probabilities.
Setting $\alpha = 1$: Laplace smoothing, while $\alpha < 1$ Lidstone smoothing.

e.g. Text classifier

No	Doc	Class
1	All Attended, All Exercises	A+
2	Attended, Exercises	A
3	No Attended	E
4	No Exercises	E
test	Attended, No Exercises	?

e.g. Text classifier

Feature = { All, Attended, Exercises, No }

$$\hat{y} = \underset{y \in C}{argmax}\ p(y) \prod_{i=1}^{d} p(x_i|y)$$

P(A+) = 1/4   P(A) = 1/4   P(E) = 1/2

P(Attended | A+)	=	(1 + 1) / (4 + 1*4)	=	2/8
P(No | A+)	=	(0 + 1) / (4 + 1*4)	=	1/8
P(Excercises | A+)	=	(1 + 1) / (4 + 1*4)	=	2/8
P(Attended | A)	=	(1 + 1) / (2 + 1*4)	=	2/6
P(No | A)	=	(0 + 1) / (2 + 1*4)	=	1/6
P(Excercises | A)	=	(1 + 1) / (2 + 1*4)	=	2/6
P(Attended | E)	=	(1 + 1) / (4 + 1*4)	=	2/8
P(No | E)	=	(2 + 1) / (4 + 1*4)	=	3/8
P(Excercises | E)	=	(1 + 1) / (4 + 1*4)	=	2/8

e.g. Text classifier

$$\hat{y} = \underset{y \in C}{argmax}\ p(y) \prod_{i=1}^{d} p(x_i|y)$$

$$P(A+ \mid test) = 1/4 * 2/8 * 1/8 * 2/8 \approx 0.002$$

$$P(A \mid test) = 1/4 * 2/6 * 1/6 * 2/6 \approx 0.005$$

$$P(E \mid test) = 1/2 * 2/8 * 3/8 * 2/8 \approx 0.01 \quad \surd$$

Attended, No Exercises : E class.

Bernoulli Naïve Bayes

Multivariate Bernoulli distributions

$$D = \{\ (x, y)\ \}\ ,\ \ x \in \mathbb{R}^d\ ,\ y \in C$$

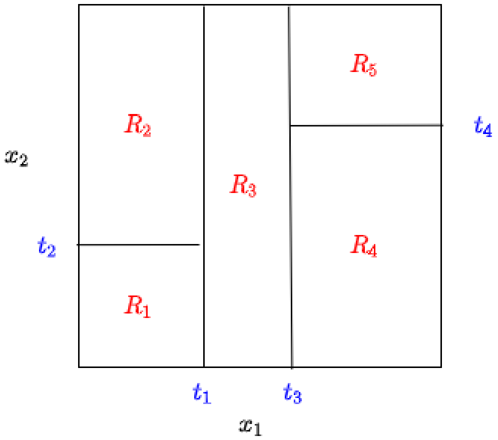
x_i binary-valued (Bernoulli, boolean) variable.

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)$$

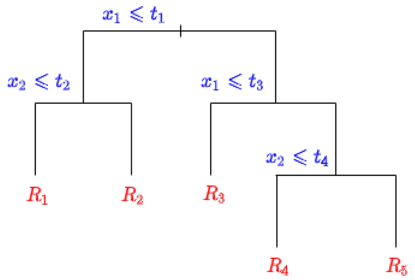
Decision Tree

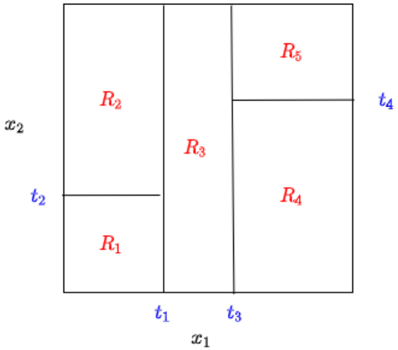
e.g. split $x(x_1, x_2)$ to 5 region



Decision Tree

$\{x(x_1, x_2), y\}$





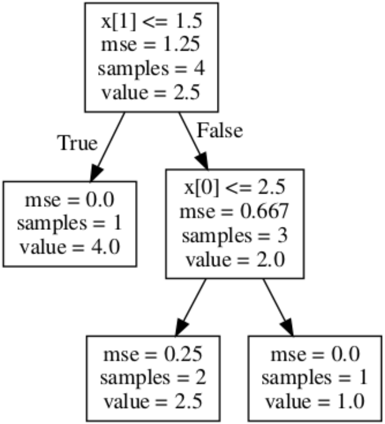
$$\hat{f}(x) = \sum_{m=1}^{5} c_m I\{\ (x_1, x_2) \in R_m\ \}$$

Dataset

$x = \{\ (2,4), (3,3), (1,2), (4,1)\ \}$
$y = \{\ 3, 1, 2, 4\ \}$

Predict

$(2.3, 2) \Rightarrow \hat{y} = 2.5$
$(2.5, 1) \Rightarrow \hat{y} = 4.0$



Dataset

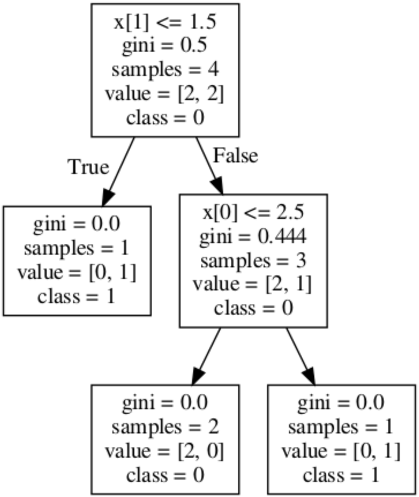
$x = \{\ (2,4),(3,3),(1,2),(0,1)\ \}$

$y = \{\ 0,1,0,1\ \}$

Predict

$(3.2, 1.4) \Rightarrow \hat{y} = 1$

$(2.3, 2.4) \Rightarrow \hat{y} = 0$



Decision Tree Model

- Dataset: $\{(x_i, y_i)\}_1^n$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ $(or\ y_i \in\ Category)$
- Partition: M regions $R_1, R_2, ..., R_M$
- Model response as a constant c_m in each region:
- I: Indicator function

$$f(x) = \sum_{m=1}^{M} c_m I\{\ x \in R_m\ \}$$

- Data at node R with n samples
- Candidate split (j, s) : feature j , threshold s

$$R(j,s) \to R_1(j,s)\ +\ R_2(j,s)$$

Left $\quad R_1(j,s) = \{\ (x,y) \mid x_j \leqslant s\ \}$

Right $\quad R_2(j,s) = \{\ (x,y) \mid x_j > s\ \}$

Loss Function

- $H()$: **Loss Function**
- $G()$: **Information Gain**

$$\boxed{G(R(j,s)) = \frac{n_1}{n}H(R_1(j,s)) + \frac{n_2}{n}H(R_2(j,s))}$$

$$\boxed{(j,s)^* = \underset{(j,s)}{argmin}\ G(R(j,s))}$$

Recurse

$R_1(j,s)$, $R_2(j,s)$

Until maximum allowable depth is reached

Decision Tree use for both classification and regression tasks.

CART algorithm

Loss function base on Mean Squared Error (MSE or $\mathscr{L}_2$ error)

$$MSE(.) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

The best $f(x_i)$ is average of y_i on region R

$$c_k = ave(y_i \mid x_i \in R_k)$$

$$H(R_k) = MSE(R_k) = \frac{1}{n}\sum_{y \in R_k}(y - c_k)^2$$

$$G(R(j,s)) = \left\{ \frac{n_1}{n}\frac{1}{n_1}\sum_{y \in R_1}(y - c_1)^2 + \frac{n_2}{n}\frac{1}{n_2}\sum_{y \in R_2}(y - c_2)^2 \right\}$$

Regression Decision Tree algo.

CART algorithm

$$G(R(j,s)) = \frac{1}{n}\Big\{ \sum_{y\in R_1} (y - c_1)^2 \; + \; \sum_{y\in R_2} (y - c_2)^2 \Big\}$$

$$(j,s)^* = \underset{(j,s)}{argmin} \left\{ \sum_{y\in R_1(j,s)} (y - c_1)^2 + \sum_{y\in R_2(j,s)} (y - c_2)^2 \right\}$$

Regression Decision Tree algo.

CART(R, $stop!$)

1. $list\ W = \{\ \}$
2. $for\ all\ j:\ feature\ x_j$
   - $sort\ Domain\ \{x_j\}$
   - $for\ all\ t_k \in Domain\ \{x_j\}$
     - $choose\ s:\ s = \frac{(t_k + t_{k+1})}{2}$
     - $w(j,s) = \sum_{y \in R_1(j,s)} (y - c_1)^2 + \sum_{y \in R_2(j,s)} (y - c_2)^2$
     - $add\ w(j,s)\ to\ list\ W$
3. $w(j,s) = min\{W\}$
4. **CART**($R_1(j,s)$, $stop!$) , **CART**($R_2(j,s)$, $stop!$)

Regression Tree Algorithm

Problem

- How large should we grow the tree?
- Very large tree might overfit the data.
- Small tree might not show the important structure.
- Optimal tree size, we can choose from the dataset.
- Real domain x !. We have to partition it.

Classification Decision Tree algo.

Loss function base on proportion p_{mk} of class k in region R_m with N_m observations:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) = \frac{n_k}{N_m}$$

- **Gini**

$$H(R_m) = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2$$

- **Entropy**

$$H(R_m) = -\sum_{k=1}^{K} p_{mk} \, log \, p_{mk}$$

- **Misclassification**

$$H(R_m) = 1 - max(p_{mk})$$

CART algorithm

gini

$$(j,s)^* = \underset{(j,\ s)}{argmin} \left\{ \frac{n_1}{n} H(R_1) \ + \ \frac{n_2}{n} H(R_2) \right\}$$

$$(j,s)^* = \underset{(j,\ s)}{argmin} \left\{ \frac{n_1}{n} (1 - \sum_{k=1}^{K} p_{1k}^2) \ + \ \frac{n_2}{n} (1 - \sum_{k=1}^{K} p_{2k}^2) \right\}$$

$$(j,s)^* = \underset{(j,\ s)}{argmin} \left\{ 1 - (\frac{n_1}{n} \sum_{k=1}^{K} p_{1k}^2 \ + \ \frac{n_2}{n} \sum_{k=1}^{K} p_{2k}^2) \right\}$$

$$(j,s)^* = \underset{(j,\ s)}{argmax} \left\{ n_1 \sum_{k=1}^{K} p_{1k}^2 \ + \ n_2 \sum_{k=1}^{K} p_{2k}^2 \right\}$$

CART algorithm (gini)

CART($R,\ stop!$)

1. $list\ W = \{\ \}$
2. $for\ all\ j:\ feature\ x_j$
   - $sort\ Domain\ \{x_j\}$
   - $for\ all\ t_k \in Domain\ \{x_j\}$
     - $choose\ s:\ s = \frac{(t_k + t_{k+1})}{2}$
     - $w(j,s) = n_1 \sum_{k=1}^{K} p_{1k}^2 + n_2 \sum_{k=1}^{K} p_{2k}^2$
     - $add\ w(j,s)\ to\ list\ W$
3. $w(j,s) = max\{W\}$
4. **CART**($R_1(j,s),\ stop!$) , **CART**($R_2(j,s),\ stop!$)

e.g. Classification Decision Tree

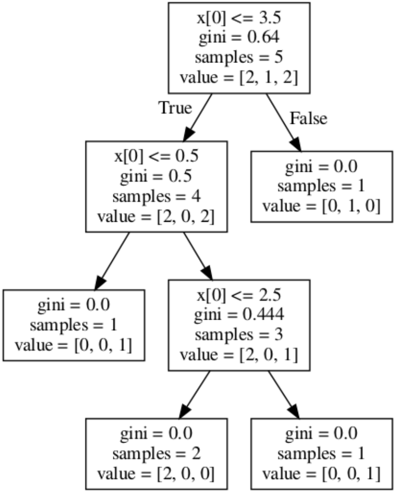
Dataset

$$x = \{\ 3,\ 1,\ 0,\ 4,\ 2\ \}$$
$$y = \{\ 2,\ 0,\ 2,\ 1, 0\ \}$$

Predict

$$value = [n_0, n_1, n_2]$$
$$number\ of\ class$$

$$(x = 4.5) \Rightarrow \hat{y} = 1$$
$$(x = 1.3) \Rightarrow \hat{y} = 0$$



Equation of a Line

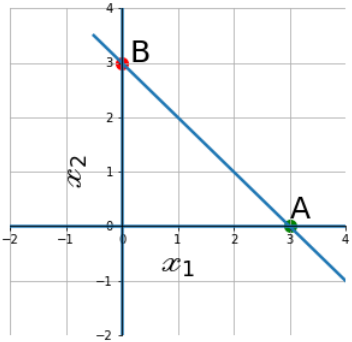
$in\ \mathbb{R}^2$ Line from 2 points.

$$\frac{x_1 - x_1^A}{x_1^A - x_1^B} = \frac{x_2 - x_2^A}{x_2^A - x_2^B}$$

$$\iff \quad \frac{x_1 - 3}{3 - 0} = \frac{x_2 - 0}{0 - 3}$$

$$\iff \quad x_2 = -x_1 + 3$$

$$\iff \quad x_1 + x_2 - 3 = 0$$



$A(x_1^A, x_2^A)\ ,\ B(x_1^B, x_2^B)$

General Line *(dot product)*

$$\overline{w} \cdot \overline{x} + b = 0$$

with $\overline{w} = (w_1, w_2)\ ,\ \overline{x} = (x_1, x_2)$

Normal and Direction vector

$in\ \mathbb{R}^2$

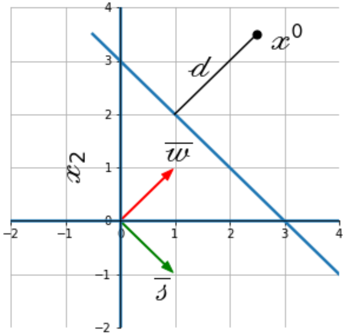
$$\overline{w} \cdot \overline{x} + b = 0$$

$\overline{w} = (w_1, w_2) : \ Normal\ vector.$
$\overline{s} = (w_2, -w_1) : \ Direction\ vector.$

Norm of $\overline{w}$ $\quad \|\overline{w}\| = \sqrt{w_1^2 + w_2^2}$

$d : \ distance\ from\ x^0\ to\ line.$

$$d = \frac{|\overline{w} \cdot \overline{x^0} + b|}{\|\overline{w}\|}$$



$$\Rightarrow distance\ from\ (0, 0)\ to\ line\ = \frac{|b|}{|\overline{w}|}$$

$\overline{u} \cdot \overline{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n \qquad (algebraic)$
$\overline{u} \cdot \overline{v} = \|\overline{u}\| \|\overline{v}\| cos(\alpha) \qquad (geometric)$

Distance between Parallel Lines

$in\ \mathbb{R}^2$

$$\ell_1 : \overline{w} \cdot \overline{x} + b_1 = 0$$
$$\ell_2 : \overline{w} \cdot \overline{x} + b_2 = 0$$

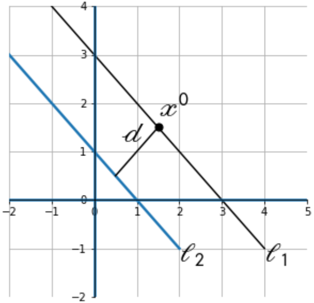
$\ell_1\ parallel\ with\ \ell_2$

$x_0 \in \ell_1 \Rightarrow \overline{w} \cdot \overline{x^0} + b_1 = 0$

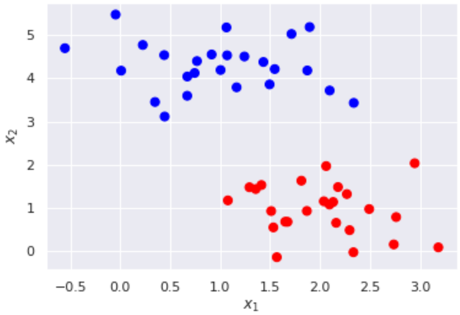
$d:\ distance\ from\ x^0\ to\ \ell_2$

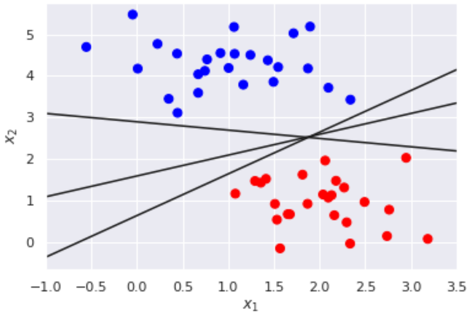
$$d = \frac{|\overline{w} \cdot \overline{x^0} + b_2|}{\|\overline{w}\|} = \frac{|b_2 - b_1|}{\|\overline{w}\|}$$

$$d = \frac{|b_2 - b_1|}{\|\overline{w}\|}\ , \quad unit\ vector\ \hat{w}, \quad \hat{w} = \frac{\overline{w}}{\|w\|}$$



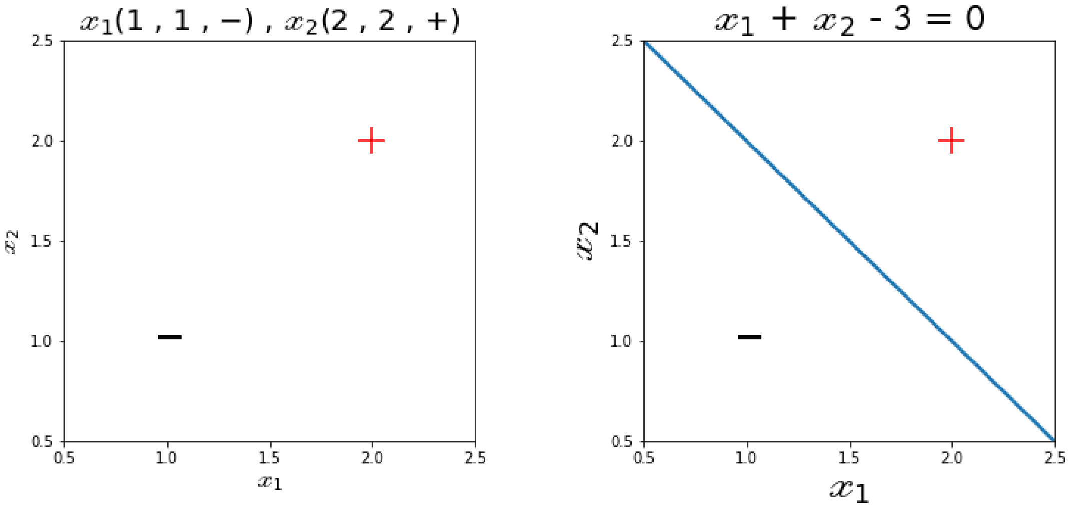
SVM Overview





- dataset from sklearn. 2 class.
- many lines can separate 2 class.

SVM Overview



SVM, Linear, Binary class

$$\mathcal{D} = \Big\{(x_i, y_i)\Big\}_1^n \ , \ x_i \in \mathbb{R}^d \ , \ y_i \in \{-1, 1\}$$

Key idea: find widest separating "street"

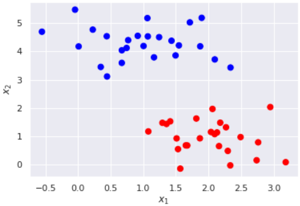
$h^0 \ : \quad w \cdot x + b = 0$

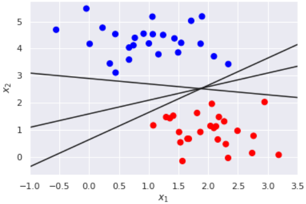
$h^+ : \quad w \cdot x + b = +1$

$h^- : \quad w \cdot x + b = -1$

Model

$$f(x) = sign(w \cdot x + b)$$





SVM, Linear, binary classifiers

$\forall(x_i, y_i)$

$h^+ : \quad w \cdot x_i + b - 1 \geqslant 0 \quad , \quad y_i = +1$

$h^- : \quad w \cdot x_i + b + 1 \leqslant 0 \quad , \quad y_i = -1$

$\Rightarrow$

$w \cdot x_i + b \geqslant +1 \quad , \quad y_i = +1$

$w \cdot x_i + b \leqslant -1 \quad , \quad y_i = -1$

$\Rightarrow$

1^{st} *constrain*

$$y_i(w \cdot x_i + b) \geqslant 1$$

Distance from h^+ to h^-

$d = \frac{|(b-1)-(b+1)|}{\|w\|} = \dfrac{2}{\|w\|}$

2^{nd} *constrain*

Maximize $\dfrac{2}{\|w\|} \Rightarrow$ minimize $\frac{1}{2}\|w\|^2$

SVM, Optimization with constraints

$$\underset{w,b}{argmin} \quad \frac{1}{2}\|w\|^2$$

$$s.t. \qquad y_i(w \cdot x_i + b) - 1 \geqslant 0 \quad , \quad \forall (x_i, y_i) \in \mathcal{D}$$

Lagrange Multipliers

$$\begin{aligned}\mathcal{L}(w, b, \alpha) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i + b) - 1] \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \; y_i \; w \cdot x_i - \sum_{i=1}^{n} \alpha_i \; y_i \; b + \sum_{i=1}^{n} \alpha_i \quad (1)\end{aligned}$$

- Lagrange multiplier α_i
- Satisfies the Karush–Kuhn–Tucker (KKT)

$$\begin{aligned}&\alpha_i [y_i(w \cdot x_i + b) - 1] = 0 \\ &\alpha_i \geqslant 0\end{aligned}$$

- Minimize $\mathcal{L}$ respect to w , b
- and maximize $\mathcal{L}$ respect to α

SVM, Optimization with constraints

Minimize $\mathcal{L}$ respect to w , b

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (2)\ , \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad (3)$$

and maximize $\mathcal{L}$ respect to α use (1) and substitute by (2), (3):

$$\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \ y_i \ w \cdot x_i - \sum_{i=1}^{n} \alpha_i \ y_i \ b + \sum_{i=1}^{n} \alpha_i$$

$$\begin{aligned}
\mathcal{L}_{dual} &= \frac{1}{2} w \cdot w - w \cdot \left(\sum_{i=1}^{n} \alpha_i \ y_i \ x_i \right) - b \sum_{i=1}^{n} \alpha_i \ y_i + \sum_{i=1}^{n} \alpha_i \\
&= -\frac{1}{2} w \cdot w + \sum_{i=1}^{n} \alpha_i \\
&= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \ \alpha_j \ y_i \ y_j \ x_i \cdot x_j
\end{aligned}$$

Dual problem

$$\underset{\alpha}{argmax} \ \ \mathcal{L}_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \ \alpha_j \ y_i \ y_j \ x_i \cdot x_j \tag{4}$$

$$s.t. \qquad \sum_{i=1}^{n} \alpha_i \ y_i = 0 \ \ , \qquad \alpha_i \geqslant 0$$

$w, \ b$

KKT: $\qquad \alpha_i[y_i(w \cdot x_i + b) - 1] = 0$

Two case:

1. $\alpha_i = 0$, or
2. $y_i(w \cdot x_i + b) - 1 = 0 \quad \Rightarrow \quad y_i(w \cdot x_i + b) = 1$

$\Rightarrow \alpha > 0 \ \ then \ \ y_i(w \cdot x_i + b) = 1$

and x_i must be a support vector

if a point is not a support vector, then $\alpha_i = 0$

SVM Linear Classifier

$$w = \sum_{\alpha_i > 0} \alpha_i y_i x_i$$

b_i per support vector:

$\alpha_i(y_i(w \cdot x_i + b_i) - 1) = 0$

$y_i(w \cdot x_i + b_i) = 1$

$$b_i = \frac{1}{y_i} - w \cdot x_i$$

$$b_i = y_i - w \cdot x_i$$

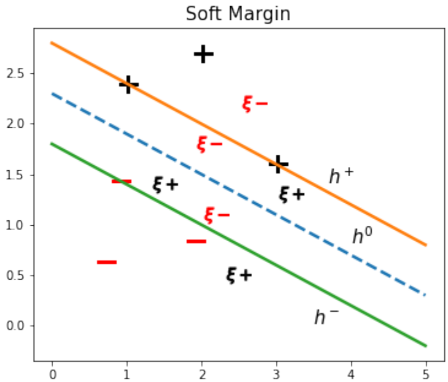
$$b = \underset{\alpha_i > 0}{avg}\{b_i\}$$

Linear Classifier

$$\hat{y} = \ sign(w \cdot x + b)$$

Soft Margin

- x_i away from hyperlane
- slack variables ξ_i for x_i
- $\xi_i \geqslant 0$
- $\xi_i = 0$, x_i at least $\frac{1}{\|w\|}$ away from hyperlane
- $0 < \xi_i < 1$ x_i within the margin
- $\xi_i \geqslant 1$ x_i appears on the wrong side



$$y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$$

Soft Margin

$$\underset{w,b,\xi_i}{argmin} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \qquad y_i(w \cdot x_i + b) \geqslant 1 - \xi_i$$

$$\xi_i \geqslant 0 \;, \quad \forall x_i \in \mathcal{D}$$

$C > 0$

– large C $\sim$ small margin,

– small C $\sim$ large margin.

Soft Margin, Lagrange Multipliers

$$\underset{\alpha}{argmax} \;\; \mathcal{L}_{dual} \;=\; \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \; \alpha_j \; y_i \; y_j \; x_i \cdot x_j \tag{4}$$

$$s.t. \qquad \sum_{i=1}^{n} \alpha_i \; y_i = 0$$

$$0 \leqslant \alpha_i \leqslant C$$

w vector and b

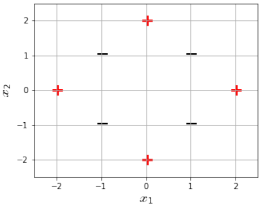
$$w = \sum_{\alpha_i > 0} \alpha_i \; y_i \; x_i$$

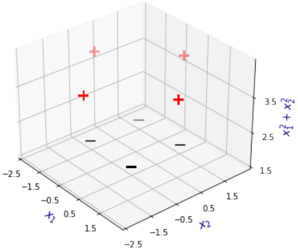
$$b = \underset{\alpha_i > 0}{avg} \{b_i\}$$

Linear Classifier

$$\hat{y} = \; sign(w \cdot x + b)$$

Kernel Trick





e.g. Simple Kernel Trick: $\mathcal{R}^2 \rightarrow \mathcal{R}^3$

$$x(x_1, x_2)$$

$$\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$$

Kernel Trick, Lagrange Multipliers

$$\underset{\alpha}{argmax} \ \mathcal{L}_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \ \alpha_j \ y_i \ y_j \ K(x_i, x_j) \qquad (5)$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i \ y_i = 0$$

$$0 \leqslant \alpha_i \leqslant C$$

w vector and b

$$w = \sum_{\alpha_i > 0} \alpha_i \ y_i \ \phi(x_i)$$

$$b = \underset{\alpha_i > 0}{avg} \{b_i\}$$

Kernel SVM Classifier

$$\hat{y} = \ sign(w \cdot \phi(x) + b)$$

Dual Solution: Gradient Ascent

$(5) \Rightarrow$

$$\mathcal{L}(\alpha_k) = \alpha_k - \tfrac{1}{2}\alpha_k^2 \ y_k^2 \ K(x_k \cdot x_k) - \alpha_k \ y_k \sum_{\substack{i=1 \\ i \neq k}}^{n} \alpha_i \ y_i \ K(x_i \cdot x_k)$$

$$\nabla \mathcal{L}(\alpha) = \left(\frac{\partial \mathcal{J}(\alpha)}{\partial \alpha_1}, \frac{\partial \mathcal{J}(\alpha)}{\partial \alpha_2}, \ldots, \frac{\partial \mathcal{J}(\alpha)}{\partial \alpha_n} \right)$$

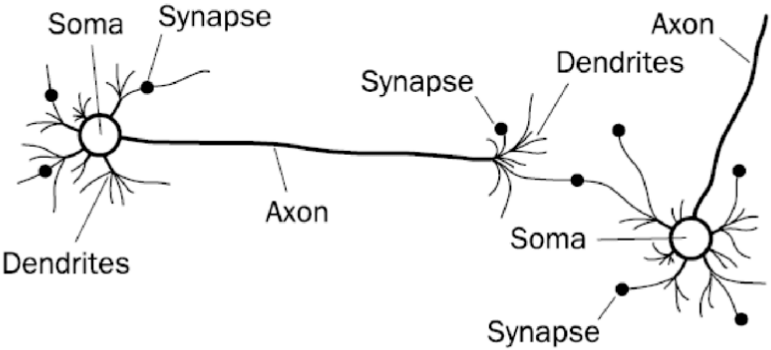
$$\frac{\partial \mathcal{L}}{\partial \alpha_k} = 1 - y_k \left(\sum_{i=1}^{n} \alpha_i \ y_i \ \widetilde{K}(x_i \cdot x_k) \right)$$

Gradient ascent

$$\alpha_{t+1} = \alpha_t + \eta \nabla \mathcal{J}(\alpha_t)$$

This algorithm can demo by scikit-learn.

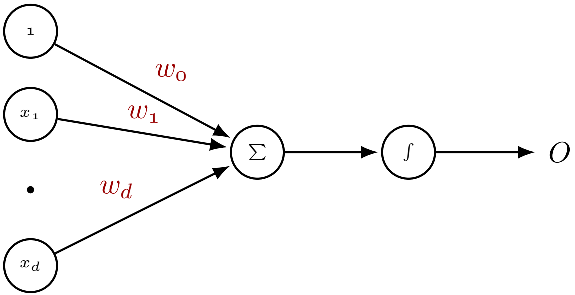
Biological Neuron



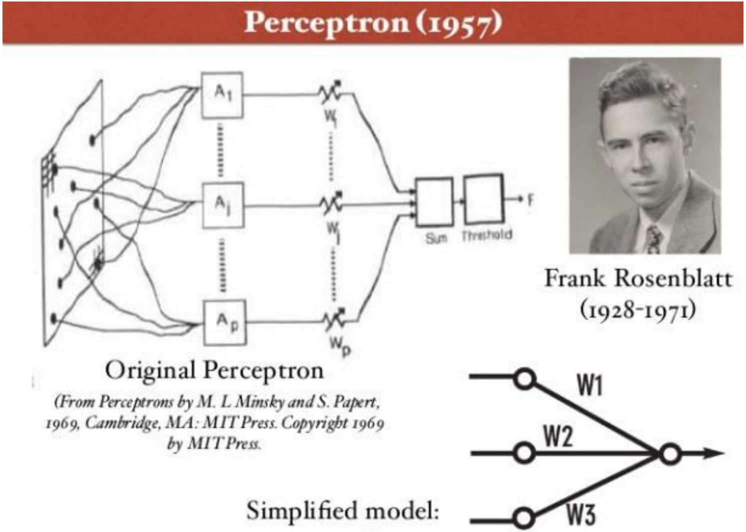
Biological Neuron vs. Artificial Neuron

Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

Artificial Neuron

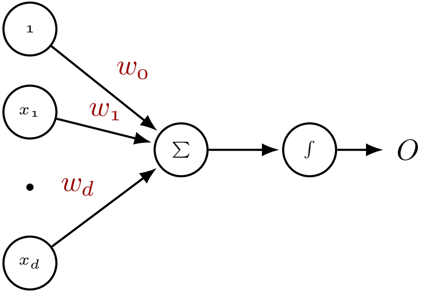


Rosenblatt's Perceptron



Rosenblatt's Perceptron

$$\mathcal{D} = \left\{ \left(x^{(i)}, y^{(i)} \right) \right\}_1^n, \quad x^{(i)} \in \mathbb{R}^d \quad , \quad y^{(i)} \in \{\mathscr{C}_1, \mathscr{C}_2\} \equiv \{0, 1\}$$



Input: $x = [+1, x_1, x_2, ..., x_d]^T$ **Weight**: $w = [w_0, w_1, w_2, ..., w_d]^T$

$$\hat{y}^{(i)} = f(x^{(i)}) = \begin{cases} 1 & \textit{if } \sum_{j=0}^{d} w_j x_j^{(i)} > 0 \\ 0 & \textit{otherwise} \end{cases}$$

Perceptron Training Rule

Initialization: Initial weight vector w and learning rate η, set to some small value (e.g., 0.1)

$$w = random \quad , \quad \eta \in (0, 1]$$

Loop for *n epoch* and *not Convergence*

for each training $(x^{(i)}, y^{(i)})$ in dataset $\mathcal{D}$

Activation function:

$$sign(\sum_{j=0}^{d} w_j x_j^{(i)}) = \begin{cases} 1 & if \ \sum_{j=0}^{d} w_j x_j^{(i)} > 0 \\ 0 & otherwise \end{cases}$$

Class response: $\hat{y}^{(i)} = sign(\sum_{j=0}^{d} w_j x_j^{(i)})$

Modify weight vector: $w = w + \eta(y^{(i)} - \hat{y}^{(i)})x^{(i)}$

End

Perceptron Training Rule

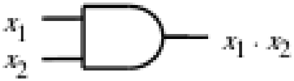
$$\boxed{w = w + \eta(y^{(i)} - \hat{y}^{(i)})x^{(i)}}$$

– This learning procedure can be proven to convergen to a weight vector that correctly classifies all training samples (Minsky and Papert 1969).

– If the data are not linearly separable, convergence is not assured.

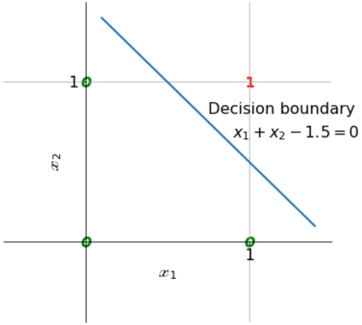
Perceptron Training Rule

AND gate



How to learn?

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



Perceptron learning AND gate

Initial Weight vector: $w = (-0.5, 0.4, 1)$, Learning rate: $\eta = 0.1$

$(1, x)$	w	$w^T x$	$\hat{y}$	y	$w + \eta(y - \hat{y})x$
$(1, 0, 0)$	$(-0.5, 0.4, 1)$	-0.5	0	0	*not update* w
$(1, 0, 1)$	$(-0.5, 0.4, 1)$	0.5	1	0	$(-0.5, 0.4, 1) + 0.1(0 - 1)(1, 0, 1)$
$(1, 1, 0)$	$(-0.6, 0.4, 0.9)$	-0.2	0	0	*not update* w
$(1, 1, 1)$	$(-0.6, 0.4, 0.9)$	0.7	1	1	*not update* w
$(1, 0, 0)$	$(-0.6, 0.4, 0.9)$	-0.6	0	0	*not update* w
$(1, 0, 1)$	$(-0.6, 0.4, 0.9)$	0.3	1	0	$(-0.6, 0.4, 0.9) + 0.1(0 - 1)(1, 0, 1)$
$(1, 1, 0)$	$(-0.7, 0.4, 0.8)$	-0.3	0	0	*not update* w
$(1, 1, 1)$	$(-0.7, 0.4, 0.8)$	0.5	1	1	*not update* w
$(1, 0, 0)$	$(-0.7, 0.4, 0.8)$	-0.7	0	0	*not update* w
$(1, 0, 1)$	$(-0.7, 0.4, 0.8)$	0.1	1	0	$(-0.7, 0.4, 0.8) + 0.1(0 - 1)(1, 0, 1)$
$(1, 1, 0)$	$(-0.8, 0.4, 0.7)$	-0.4	0	0	*not update* w
$(1, 1, 1)$	$(-0.8, 0.4, 0.7)$	0.3	1	1	*not update* w

Perceptron learning AND gate

$(1, x)$	w	$w^T x$	$\hat{y}$	y	$w + \eta(y - \hat{y})x$
$(1, 0, 0)$	$(-0.8, 0.4, 0.7)$	-0.8	0	0	*not update* w
$(1, 0, 1)$	$(-0.8, 0.4, 0.7)$	-0.1	0	0	*not update* w
$(1, 1, 0)$	$(-0.8, 0.4, 0.7)$	-0.4	0	0	*not update* w
$(1, 1, 1)$	$(-0.8, 0.4, 0.7)$	0.3	1	1	*not update* w

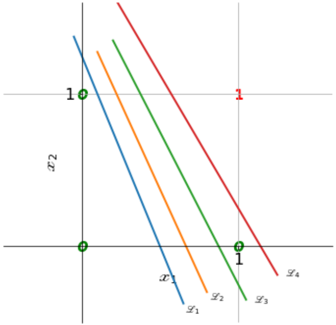
Learning result $\mathscr{L}_4$

$$\mathscr{L}_1 : 0.4x_1 + x_2 - 0.5 = 0$$

$$\mathscr{L}_2 : 0.4x_1 + 0.9x_2 - 0.6 = 0$$

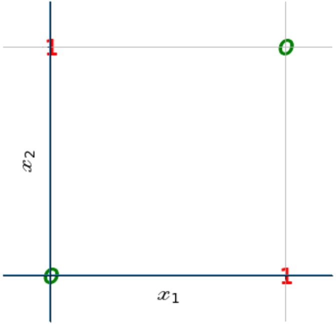
$$\mathscr{L}_3 : 0.4x_1 + 0.8x_2 - 0.7 = 0$$

$$\mathscr{L}_4 : 0.4x_1 + 0.7x_2 - 0.8 = 0$$



Perceptron XOR problem

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Two class $\{0, 1\}$ cannot be separated by a single linear line.
XOR gate cannot be solved by Rosenblatt's perceptron.

Perceptron XOR problem

Solution 1

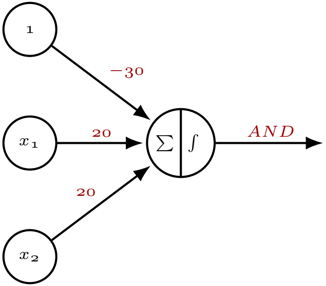
x_1	x_2	y	$\overline{x}_1 x_2$	$x_1 \overline{x}_2$	$\overline{x}_1 x_2 + x_1 \overline{x}_2$
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

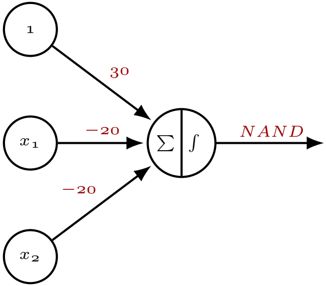
Solution 2

x_1	x_2	y	$\overline{x_1 x_2}$	$x_1 + x_2$	$\overline{x_1 x_2}(x_1 + x_2)$
0	0	0	1	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0

Perceptron XOR problem

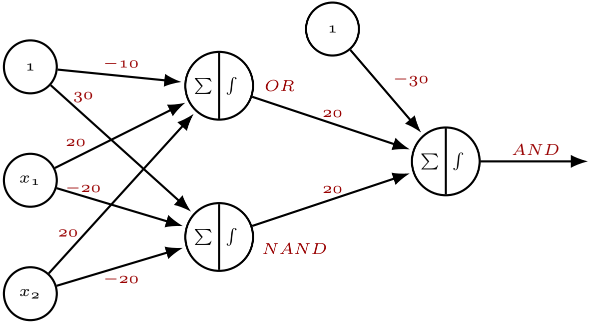
Perceptron for Solution 2





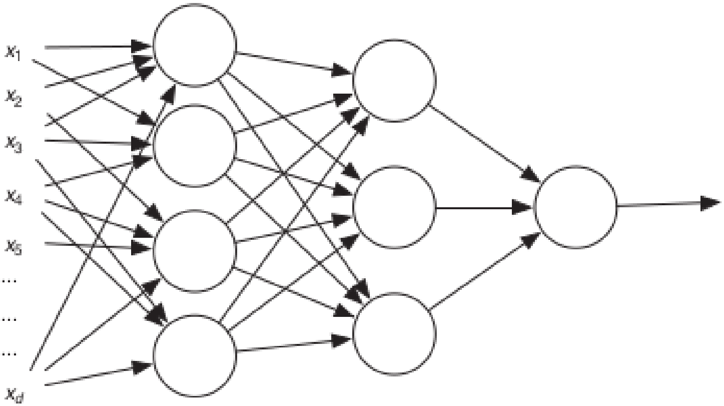
Perceptron XOR problem

Connect three gate

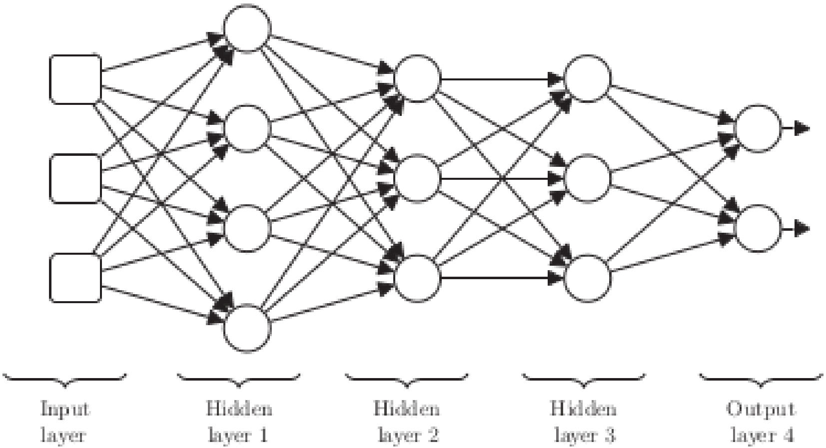


XOR as a combination of 3 basic perceptrons

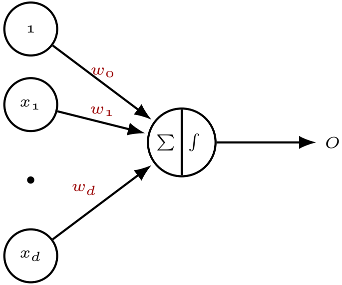
Multilayer Networks



Multilayer Networks



Activation function

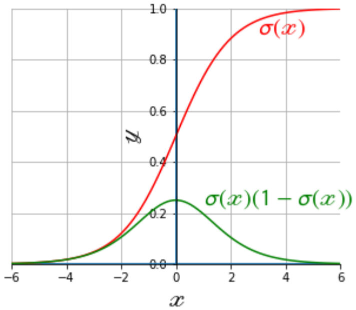


$$f(x^{(i)}) = g\left(\sum_{j=0}^{d} w_j x_j^{(i)}\right)$$

Activate function g

step (sign), sigmoid, tanh, relu, softmax, ...

Sigmoid activation function

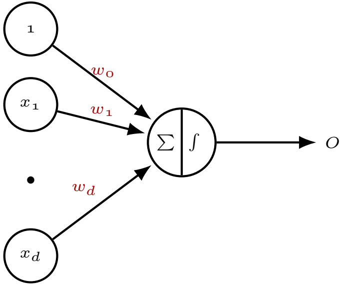


– $e = 2.71828...$ Continuous functions, differentiable, use for Gradient desent.

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}} \qquad \boxed{\frac{\partial \sigma}{\partial x} = \sigma(x)(1-\sigma(x))}$$

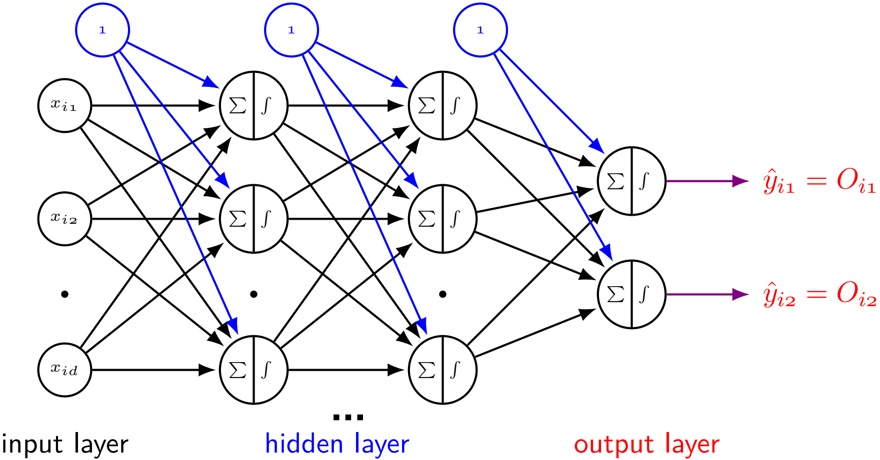
– Useful for tiny change in the weight.

Perceptron (one neuron)



$Net = \sum_{j=0}^{d} w_j x_j$; $O = g(Net)$; g: Activate Function

Neural Network (many neurons)



Loss function

- Network consider $\mathcal{K}$ outputs.
- $y_k \triangleq t_k$: Target $k^{th} \in \mathcal{K}$ of example $(x, y) \in \mathcal{D}$
- $\hat{y}_k \triangleq o_k$: Output $k^{th} \in \mathcal{K}$ of networks of $(x, y) \in \mathcal{D}$

Loss function:

Sum the errors over all of the network output units

$$
\begin{aligned}
Loss(w) \triangleq E(w) &= \frac{1}{2} \sum_{\mathcal{D}} \sum_{\mathcal{K}} (y_k - \hat{y}_k)^2 \\
&= \frac{1}{2} \sum_{\mathcal{D}} \sum_{\mathcal{K}} (y_k - o_k)^2 \\
&= \frac{1}{2} \sum_{\mathcal{D}} \sum_{\mathcal{K}} (t_k - o_k)^2
\end{aligned}
$$

Employs gradient descent algorithm to minimize $E(w)$.

Loss function for one example

– One example $e = (x, y)$.

– After forward network, we have k out put $o_k \triangleq \hat{y}_k$.

– Error all k out put o_k throught w denote by $E_e(w)$:

$$\boxed{E_e(w) = \frac{1}{2}\sum_k (y_k - o_k)^2}$$

Employs Gradient descent

The Backprop weight update rule is based on the gradient descent method w (abandon index e)

$$\boxed{w_{ij} = w_{ij} - \eta\,\frac{\partial E(w)}{\partial w_{ij}}}$$

η: learning rate.

Backpropagation algorithm

Notations:

- x_{ij}: the input from node i to unit j.
- w_{ij}: the weight associated with x_{ij}.
- $z_j = \sum w_{ij}x_{ij}$, weighted sum of inputs for unit j.
- o_j: output computed by unit j.
- σ: the sigmoid function.
- $\mathcal{K}$: the set of units in the final layer of the network.
- $succ(j)$: the set of units whose immediate inputs from the output of unit j.

Chain Rule

Chain rule to write:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}}$$

since,

$$\frac{\partial z_j}{\partial w_{ij}} = \left(\sum w_{ij} x_{ij}\right)' = (w_{ij} x_{ij})' + \left(\sum_{k \neq i} w_{kj} x_{kj}\right)' = x_{ij} + 0$$

so,

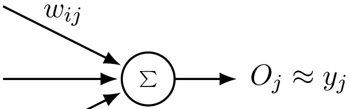
$$\boxed{\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} x_{ij}}$$

Consider two cases: the case where unit j is an output for the network, and the case where j is an internal unit.

Two cases in calculating $\frac{\partial E}{\partial z_j}$

Backpropagation algorithm

– **Case 1: Neuron j is an output neuron**



$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} x_{ij}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial z_j}$$

$$\frac{\partial E}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_k (y_k - o_k)^2 \quad , \quad (k \in \mathcal{K})$$

$\frac{\partial}{\partial o_j} (y_k - o_k)^2 = 0$ for all output units k except when $k = j$.

$$\begin{aligned}\frac{\partial E}{\partial o_j} &= \frac{\partial}{\partial o_j} \frac{1}{2}(y_j - o_j)^2 \\ &= \frac{1}{2} 2(y_j - o_j)\frac{\partial (y_j - o_j)}{\partial o_j} \\ &= -(y_j - o_j)\end{aligned}$$

Backpropagation algorithm

Since $o_j = \sigma(z_j)$

$\frac{\partial o_j}{\partial z_j} = \sigma(z_j)(1 - \sigma(z_j))$. therefore,

$\frac{\partial o_j}{\partial z_j} = \frac{\partial \sigma(z_j)}{\partial z_j} = o_j(1 - o_j)$

substituting, we obtain:

$$\frac{\partial E}{\partial z_j} = -(y_j - o_j)\ o_j\ (1 - o_j)$$

so

$$w_{ij} = w_{ij} + \eta\ (y_j - o_j)\ o_j\ (1 - o_j)\ x_{ij}$$

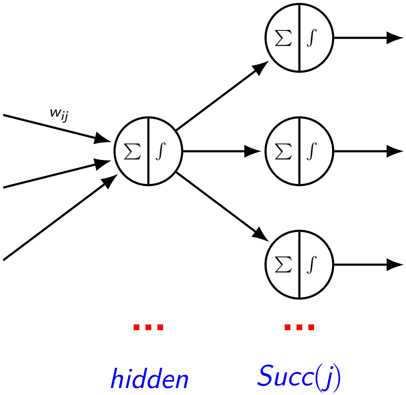
we will note:

$$\boxed{\delta_j = -\frac{\partial E}{\partial z_j} = (y_j - o_j)\ o_j\ (1 - o_j)}$$

$$\boxed{w_{ij} = w_{ij} + \eta\ \delta_j\ x_{ij}}$$

Backpropagation algorithm

– **Case 2: Neuron j is a hidden neuron**



$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} x_{ij}$$

Backpropagation algorithm

$$
\begin{aligned}
\frac{\partial E}{\partial z_j} &= \sum_{k \in succ(j)} \frac{\partial E}{z_k} \frac{\partial z_k}{\partial z_j} \\
&= \sum_{k \in succ(j)} -\delta_k \frac{\partial z_k}{\partial z_j} \\
&= \sum_{k \in succ(j)} -\delta_k \frac{\partial z_k}{\partial o_j} \frac{\partial o_j}{\partial z_j} \\
&= \sum_{k \in succ(j)} -\delta_k \; w_{jk} \frac{\partial o_j}{\partial z_j} \\
&= \sum_{k \in succ(j)} -\delta_k \; w_{jk} \; o_j \left(1 - o_j\right)
\end{aligned}
$$

Backpropagation algorithm

note

$$\boxed{\delta_j = -\frac{\partial E}{\partial z_j} = \ o_j \ (1 - o_j) \sum_{k \in succ(j)} \delta_k \ w_{jk}}$$

and

$$\boxed{w_{ij} = w_{ij} + \eta \ \delta_j \ x_{ij}}$$

Backpropagation algorithm

Input:

(x, y) training example.

η learning rate (e.g., $\eta = 0.1$).

n_i the number of network inputs.

n_h the number of units in the hidden layer.

n_o the number of output units.

The input from unit i into unit j is denoted x_{ij}, and the weight from unit i to unit j is denoted w_{ij}.

Output:

A neural network with one input layer, one hidden layer and one output layer with n_i, n_h and n_o number of neurons respectively and all its weights.

1. Create_feedforward_network (n_i, n_h, n_o)
2. Initialize all network weights to small random numbers (e.g., between -.05 and .05).

Backpropagation algorithm

3. Repeat until convergence: *small changes in the weights*

   For each training example (x, y)

   3.1 **Feed forward**: Propagate example x through the network and compute the output o_j from every neuron.

   3.2 **Propagate backward**: Propagate the errors backward.

   Case 1 For each output neuron k, calculate its error

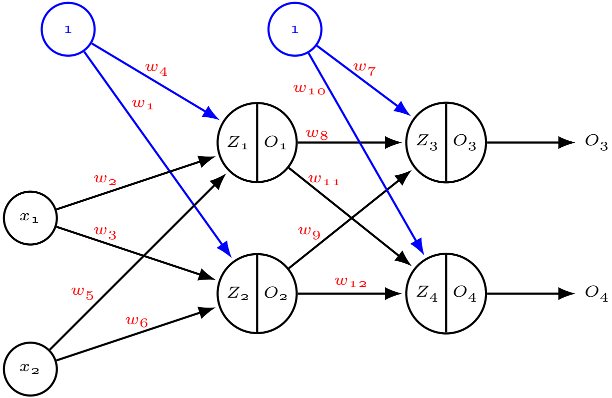
$$\boxed{\delta_k = o_k(1 - o_k)(y_k - o_k)}$$

   Case 2 For each hidden neuron h, calculate its error

$$\boxed{\delta_h = o_h(1 - o_h) \sum_{k \in succ(h)} w_{hk}\delta_k}$$

   3.3. **Update each weight**:

$$\boxed{w_{ij} = w_{ij} + \eta \ \delta_j \ x_{ij}}$$



$$(x_1 = 0.2\ ,\ x_2 = 0.3)\ \ (y_1 = 1\ ,\ y_2 = 0)$$

$$w_1 = w_4 = 0.01\ ,\ w_2 = 0.02\ ,\ w_3 = 0.03\ ,\ w_5 = 0.05\ ,\ w_6 = 0.06$$

$$w_7 = w_{10} = 0.1\ ,\ w_8 = 0.08\ ,\ w_9 = 0.09\ w_{11} = 0.11\ w_{12} = 0.12$$

$\eta = 0.1$

Feed Forward:

$z_1 = x_1 w_2 + x_2 w_3 + w_4 = 0.023$
$o_1 = \sigma(z_1) = 0.506$
$z_2 = x_1 w_5 + x_2 w_6 + w_1 = 0.038$
$o_2 = \sigma(z_1) = 0.509$
$z_3 = o_1 w_8 + o_2 w_9 + w_7 = 0.186$
$o_3 = \sigma(z_3) = 0.546$
$z_4 = o_1 w_{11} + o_2 w_{12} + w_{10} = 0.217$
$o_4 = \sigma(z_4) = 0.554$

Propagate backward:

$\delta_3 = o_3(1 - o_3)(y_1 - o_3) = 0.112$
$\delta_4 = o_4(1 - o_4)(y_2 - o_4) = -0.137$
$\delta_1 = o_1(1 - o_1)(w_8\delta_3 + w_{11}\delta_4) = -0.002$ $\delta_2 = o_2(1 - o_2)(w_9\delta_3 + w_{12}\delta_4) = -0.002$

Propagate backward Update weights:

$$
\begin{aligned}
w_7 &= w_7 + \eta\ \delta_3\ 1 &= 0.112\\
w_8 &= w_8 + \eta\ \delta_3\ o_1 &= 0.086\\
w_9 &= w_9 + \eta\ \delta_3\ o_2 &= 0.096\\
w_{10} &= w_{10} + \eta\ \delta_4\ 1 &= 0.085\\
w_{11} &= w_{11} + \eta\ \delta_4\ o_1 &= 0.103\\
w_{12} &= w_{12} + \eta\ \delta_4\ o_2 &= 0.113\\
w_1 &= w_1 + \eta\ \delta_2\ 1 &= 0.010\\
w_2 &= w_2 + \eta\ \delta_1\ x_1 &= 0.020\\
w_3 &= w_3 + \eta\ \delta_1\ x_2 &= 0.030\\
w_4 &= w_4 + \eta\ \delta_1\ 1 &= 0.010\\
w_5 &= w_5 + \eta\ \delta_2\ x_1 &= 0.045\\
w_6 &= w_6 + \eta\ \delta_2\ x_2 &= 0.060
\end{aligned}
$$

Repeat until Convergence (small changes in the weights)

Wisdom of the Crowd

Weight-guessing contest the ox.

Sir Francis Galton (1822-1911)

book, 2004, "The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations".

James Michael Surowiecki.

Naturally, not all crowds are wise (for example, greedy investors of a stock market bubble)

Lior Rokach, 2010, PATTERN CLASSIFICATION USING ENSEMBLE METHODS

Wisdom of the Crowd

In order to become wise, the crowd should comply with the following criteria:

- **Diversity of opinion** Each member should have private information.
- **Independence** Members' opinions are not determined by the opinions of those around them.
- **Decentralization** Members are able to specialize and draw conclusions based on local knowledge.
- **Aggregation** Some mechanism exists for turning private judgments into a collective decision.

Lior Rokach, 2010, PATTERN CLASSIFICATION USING ENSEMBLE METHODS

Majority Voting

e.g. Ensemble of ten classifiers (using ten learning method)

Classifier	A score	B score	C score	Selected Label
1	0.2	0.7	0.1	B
2	0.1	0.1	0.8	C
3	0.2	0.3	0.5	C
4	0.1	0.8	0.1	B
5	0.2	0.6	0.2	B
6	0.6	0.3	0.1	A
7	0.25	0.65	0.1	B
8	0.2	0.7	0.1	B
9	0.2	0.2	0.8	C
10	0.4	0.3	0.3	A

Voted result

	Class A	Class B	Class C
Votes	2	5	3

Majority Voting

x : sample.

$y_k(x)$: classification of the k'th classifier.

$I\{y = c\}$: Indicator function.

$$class(x) = \underset{c_i \in dom(y)}{argmax} \left(\sum_{k} I \{ y_k(x) = c_i \} \right)$$

Random Forest