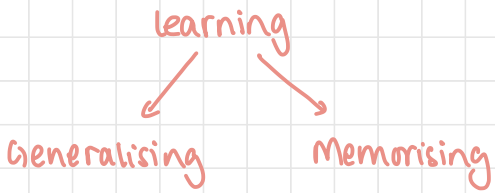
MACHINE INTELLIGENCE

UNIT-2

Supervised Learning with KNN, ANN, SVM

feedback/corrections: vibha@pesu.pes.edu

· VIBHA MASTI



Lazy vs Eager Learner

(i) Lazy Learner

- Data processed only when question asked
- Eg: piggy bank

(ii) Eager Learner

- Each transaction processed in real time
- Queries are instant
- Model keeps updating

Instance BASED LEARNING

- Stores training examples and creates model when query made with new instance

K-Nearest Neighbours

- Inductive bias: class of instance x is most similar to class of instances closest to x

- Instance-based learning (lazy learning)

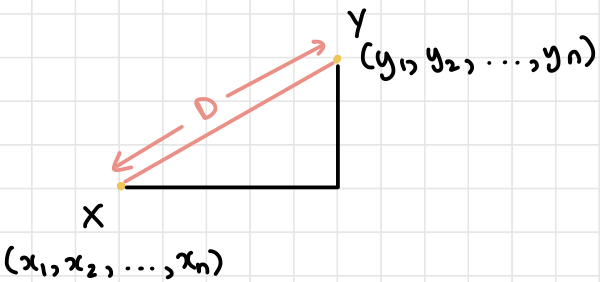
Overview

All instances correspond to points in nD space → Points may belong to a class or take a real value (classification vs regression) → Given a query point where it belongs in space → Find k nearest neighbours → Assign mode (classification) or mean (regression)

DISTANCE MEASURES

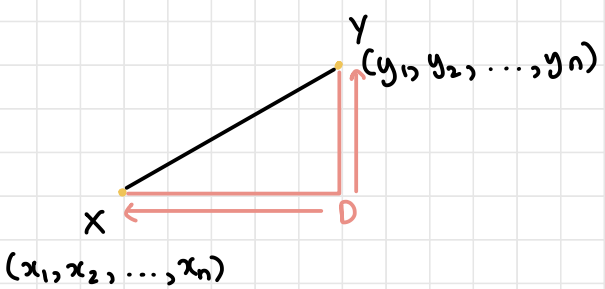
1. Euclidean distance

$$D(X,Y) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}$$



2. Manhattan distance

$$D(X,Y) = \sum_{i=1}^{n} |x_i - y_i|$$



3. Minkowski distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

if $p=1$: Manhattan
if $p=2$: Euclidean

kNN for Regression & Classification

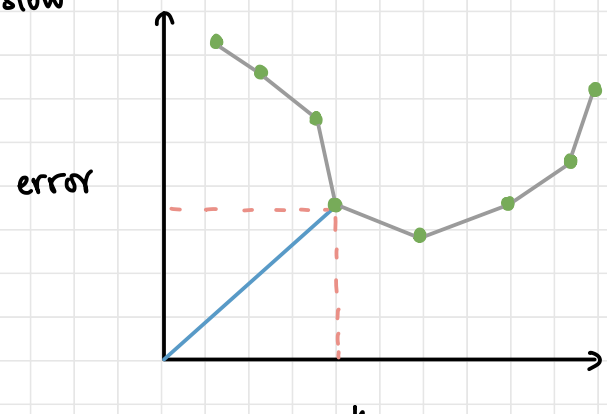
- Regression: real-valued target
- Classification: discrete-valued target

Choosing a Value of k

- k cannot be equal to the number of instances
- Usually, k = no. of classes + 1
  - in case of tie, decrement k by 1
- $k = \sqrt{n}$ also performs well (n = no. of instances in the dataset)

Elbow Method

- For every test instance, a curve of k vs error can be plotted and the min value of k with the least error is selected
- Elbow curve
- Very slow



Q: Find weight of ID 11. K=3. Use Euclidean Distance

ID	Height	Age	Weight	|Diff in Height|	|Diff in Age|	$\sqrt{\text{sum sq.}}$
1	5	45	77	0.5	7	7.02
2	5.11	26	47	0.39	12	12.01
3	5.6	30	55	0.1	8	8.00
4	5.9	34	59	0.4	4	4.02 ←
5	4.8	40	72	0.7	2	2.12 ←
6	5.8	36	60	0.3	2	2.02 ←
7	5.3	19	40	0.2	19	19.00
8	5.8	28	60	0.3	10	10.00
9	5.5	23	45	0	15	15
10	5.6	32	58	0.1	6	6.00
11	5.5	38	?	0	0	0.00

- Regression (continuous) problem: find average of k nearest values

Weight = 63.67

Error - Classification

- error = 1 if actual != predicted
- error = 0 otherwise

$$\text{Error (E)} = \frac{1}{n} \sum_{i=1}^{n} e_i$$

Error - Regression

- error = $(y_{pred} - y_{actual})^2$
- MSE

Perform KNN and find error - classification

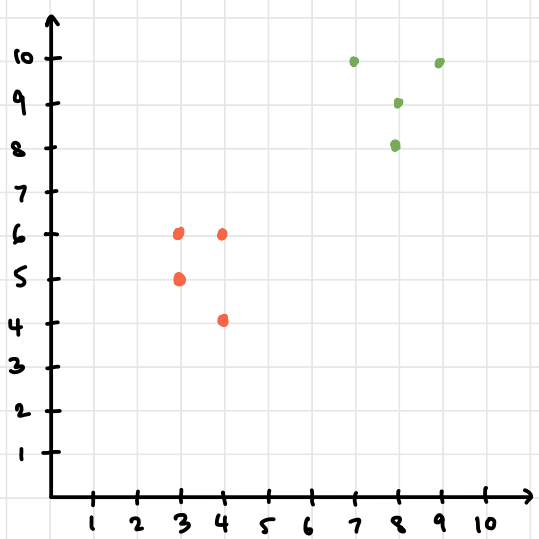
- 80:20 production and test set
- Upload prod data into memory
- Decide distance measure (Euclidean)
- Decide k value (k=4)
- For each data point in test, find distance wrt to all neighbours and choose 4 nearest neighbours
- Assign mode of neighbours to data point
- Calculate error and accuracy
- Produce model with achieved accuracy for a given k

TRAIN

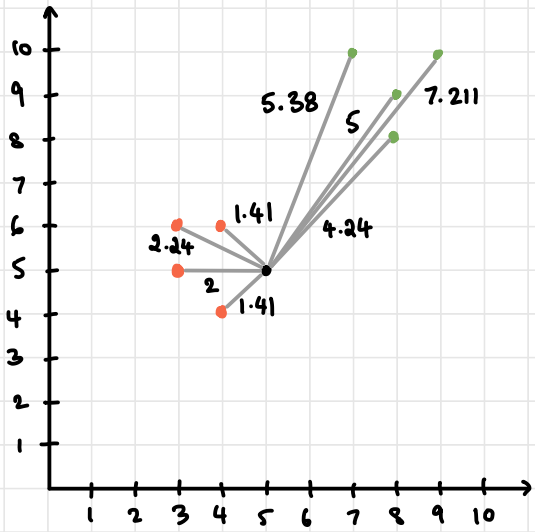
x	y	class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green

TEST

x	y	class
5	5	red
4	7	green

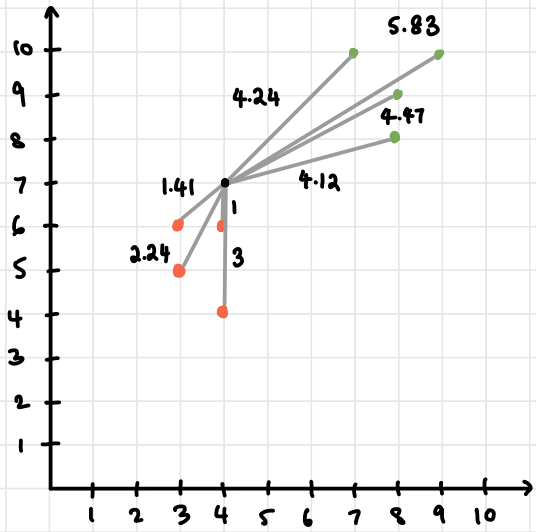


Using Euclidean Distance



pred = red
actual = red
error = 0

k = 3



pred = red
actual = green
error = 1

Error - classification

error = 0 if actual = predicted
error = 1 if actual ≠ predicted

$$E = \frac{1}{n} \sum_{i=1}^{n} e_i$$

MSE - regression

observed

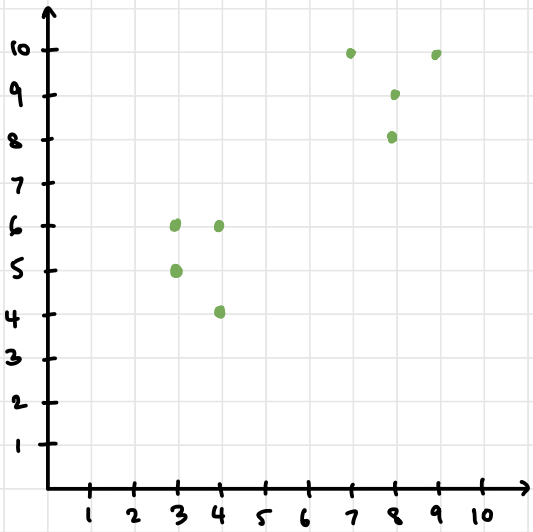
$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

predict

Perform KNN and find error - Regression

TRAIN

x	Y	class
3	5	3.5
3	6	2.6
4	6	2.8
4	4	3.1
7	10	8.7
8	9	8.5
8	8	9 3
9	10	9.7

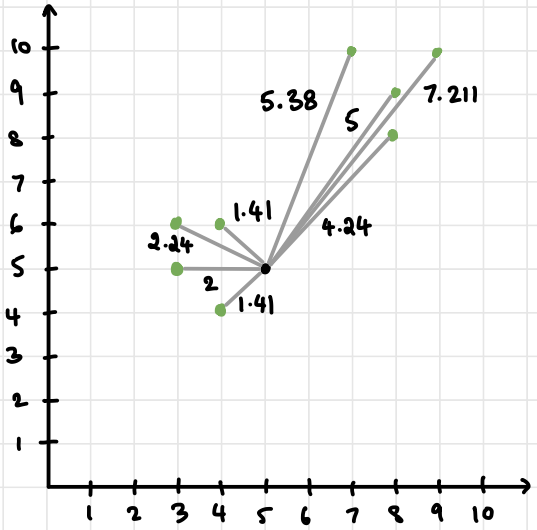


TEST

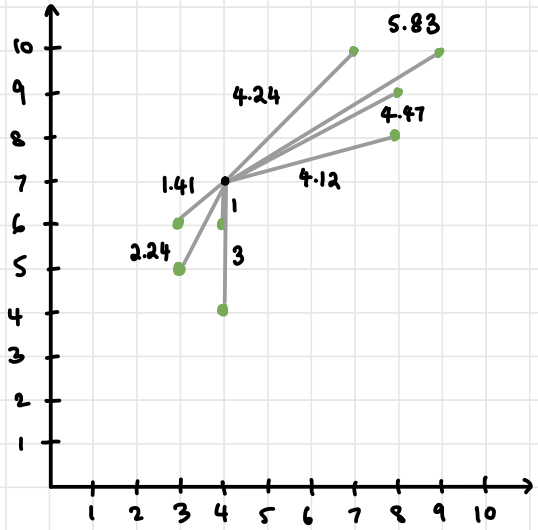
x	y	class
5	5	3.7
4	7	8.4

Using Euclidean Distance

K = 3



pred = 3.13
actual = 3.7



pred = 2.967
actual = 8.4

$error^2 = 0.3211$ $error^2 = 29.5211$

$$MSE = 14.92$$

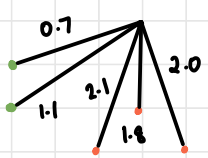
Distance Weighted kNN

- Weigh the contribution of each of the k neighbours acc. to their distance to query point x_q

$$\hat{f}(x_q) = \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \, \delta(v, f(x_i))$$

$$w_i = \frac{1}{d(x_q, x_i)}$$ ← distance measure used

Eg: $k = 5$



$$\text{weights} = \frac{1}{\text{dists}}$$

$$\text{Support for red} = \frac{1}{2.1} + \frac{1}{1.8} + \frac{1}{2.0}$$
$$= 1.532$$

$$\text{Support for green} = \frac{1}{0.7} + \frac{1}{1.1}$$
$$= 2.338$$

Bias-Variance Trade-off

K is very low
- chance of overfitting (high variance)

K is very high
- chance for underfitting (high bias)
- label based on majority in dataset

K neither too high or too low
- good trade-off

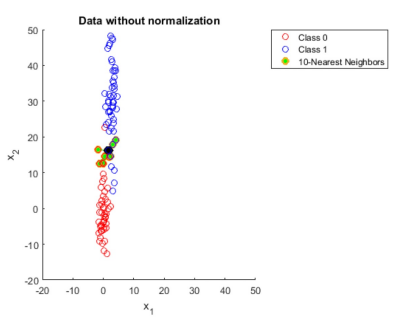
Pros of kNN

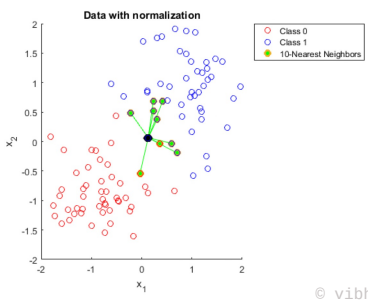
- No training required
- Only two parameters: k and distance measure

Cons of kNN

- Prediction time is computationally expensive
- Lazy algorithm — all data points to be stored
- Hard to scale
- Curse of dimensionality

Scaling





- Normalisation helps in uniformity

- Equal importance of each attribute
  - eg: salary & age – scales very different

ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

Pre normalizing

ID	Age	Income(rupees)
1	-1.192	-0.260
2	-0.447	0.608
3	1.043	0.173
4	-0.447	-1.563
5	1.043	1.042

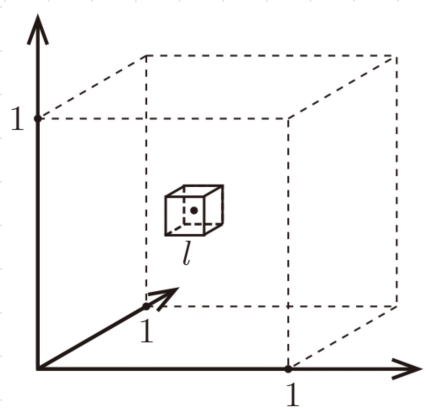
Post normalizing

Curse of Dimensionality

- When no. of dimensions is too high, points drawn from probability distribution not close together

- In d-D: assume uniform random distribution in unit cube

- https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html



- Try to find k nearest neighbours in d dimensions

- Assume all k neighbours lie in a hypercube of length ℓ (volume ℓ^d)

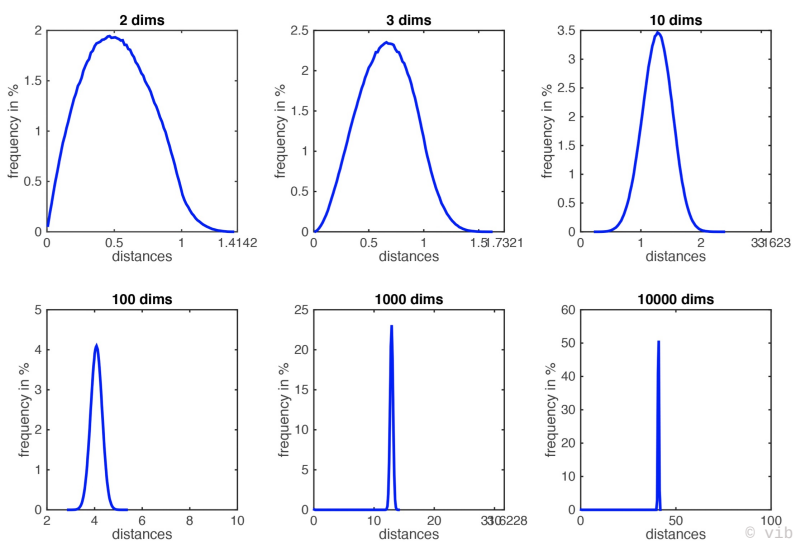
- The fraction of points in the cube (n = total points)

$$\ell^d \approx \frac{k}{n} \quad \Rightarrow \quad \ell \approx \left(\frac{k}{n}\right)^{1/d}$$

- If $n = 1000$, how big is ℓ?

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

- As $d \gg 0$, almost the entire space is required to find the k nearest neighbours
- Unknown point will not necessarily be 'similar' to its k nearest neighbours
- Histograms: distribution of pairwise distances between randomly distributed points as d grows



Solutions

1. weighted attributes for calculating distances
   - eg: predicting house prices — distance gives less weightage to colour and more to locality
   - eg: predicting maths marks — distance gives less weightage to english marks

2. cross-validation method: iteratively leaving out one of the attributes and testing
   - Best set of attributes
   - Long computational time

3. Principal Component Analysis for Dimensionality Reduction

4. Rule of thumb: 5 data instances per attribute for learning

Q: Is this a good way to calculate weights in a weighted kNN?

$$F(x, y) = \sum_{i=1}^{n} w_i f_i \qquad w_i = \frac{h_i^{-p}}{\sum_{j=1}^{n} h_j^{-p}}$$

f_i : dataset values

h: distance between query point and a data point, say Euclidean distance.

yes, as points closer are given heavier weights for +ve values of p

Q:

x_1	x_2	class
7	7	bad
7	4	bad
3	4	good
1	4	good
Q.P: 3	7	

$k=3$

Euclidean distance

No normalisation

(a) Vanilla

(b) Weighted

(a)

$\lvert x_1 - x_{1qp} \rvert^2$	$\lvert x_2 - x_{2qp} \rvert^2$	total^2
16	0	16 ←
16	9	25
0	9	9 ←
4	9	13 ←

mode = good (using vanilla kNN)

(b) Weighted

$\lvert x_1 - x_{1qp} \rvert^2$	$\lvert x_2 - x_{2qp} \rvert^2$	Euclidean		weight	class
16	0	4	←	0.25	bad
16	9	5		0.2	bad
0	9	3	←	0.33	good
4	9	$\sqrt{13}$	←	0.174	good

support for bad = 0.25

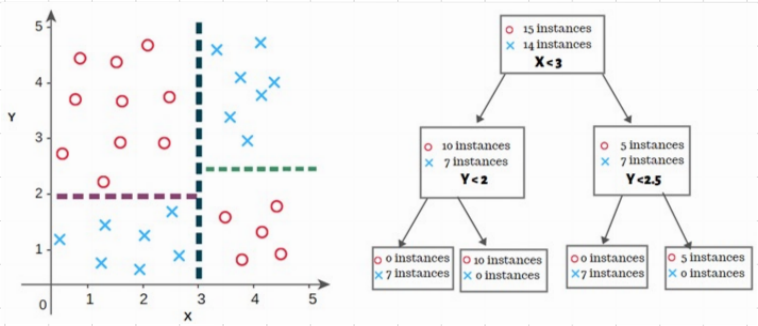
support for good = 0.504

Decision Boundaries

- Decision boundary: region of problem space where output label of classifier is ambiguous
- Separates underlying vector space into multiple sets, one for each class

Decision Boundary for Decision Trees

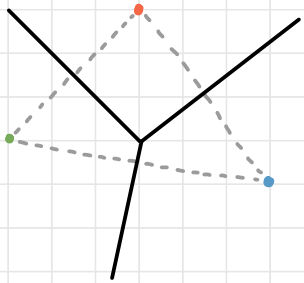
- Input space divided into axis-parallel rectangles
- Each rectangle: one class
- https://towardsdatascience.com/decision-tree-overview-with-no-maths-66b256281e2b



- Can represent arbitrary decision boundary

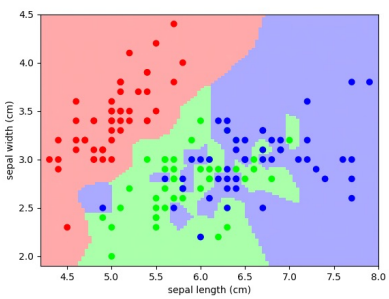
Decision Boundary for 1-KNN

- Voronoi tesselation
- Boundary drawn with points at the same distance from two training examples

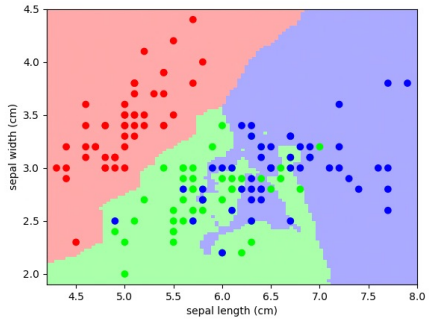


- ML Playground tool
- The decision boundary is a convex hull (fails with non-linear data)

k=1



k=3



source: scipy-lectures

- Piecewise non-linear decision boundary

Disadvantages of kNNs

- Arduous (strenuous) to find right k value
- Sensitive to noise
- Performance low for large datasets (calculating distances is linear)
- Performance low for high dimensionality (distances in each dim)
- High memory to store all training examples

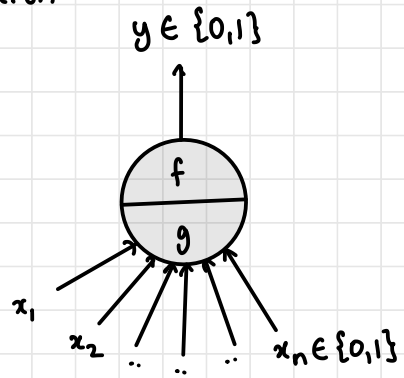
Artificial Neural Network

Biological Neuron	Artificial Neuron
Dendrites	Input
Cell nucleus (soma)	Node
Axon	Output
Synapse	Interconnection

- Input layer, hidden layers, output layer

McCulloch Pitts Neuron - Logic gates

- Simple neuron



- g aggregates the input function and f takes a decision based on the aggregation

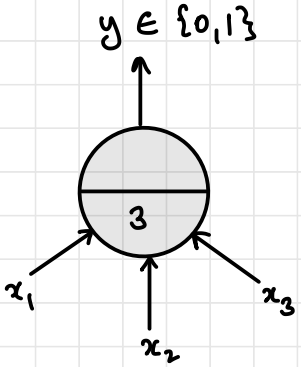
- Inputs can be excitatory or inhibitory

- If any x_i is inhibitory, $y = 0$. Else

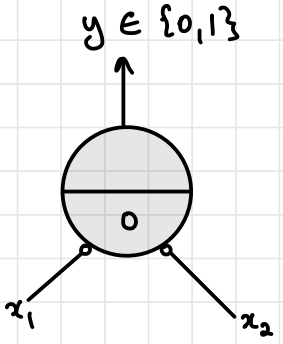
$$g(x_1, x_2, \dots, x_n) = g(x) = \sum_{i=1}^{n} x_i$$

$$y = f(g(x)) = 1 \text{ if } g(x) \geq \theta$$
$$= 0 \text{ if } g(x) < \theta$$

- AND function

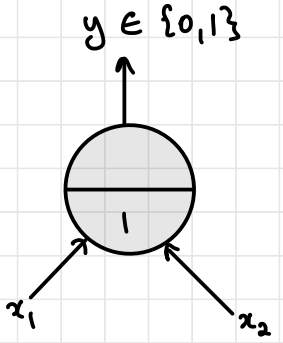


- NOR function (○ → inhibition)



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

- OR function



Geometric Representation of MP Neuron

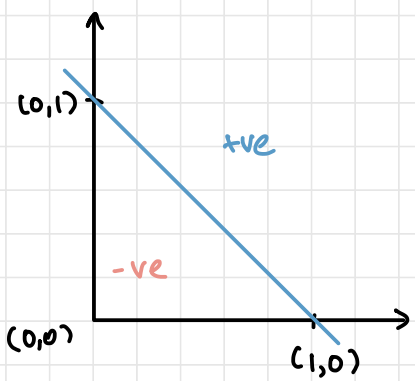
- Single MP neuron splits input points into two parts

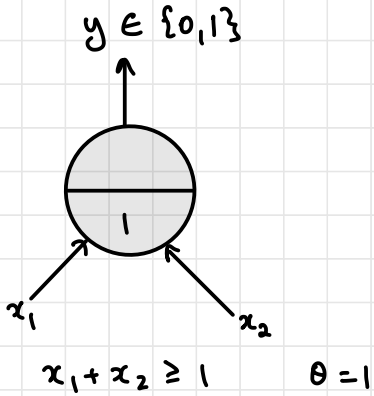
  Points lying on or above the line $\sum_{i=1}^{n} x_i - \theta = 0$

- 0: below line
  1: above line
  } must be linearly separatable

- Cannot show XOR

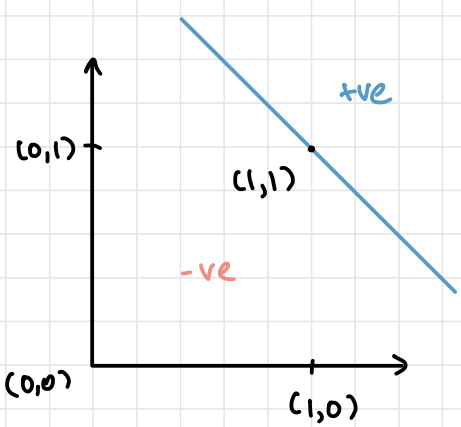
(a) OR

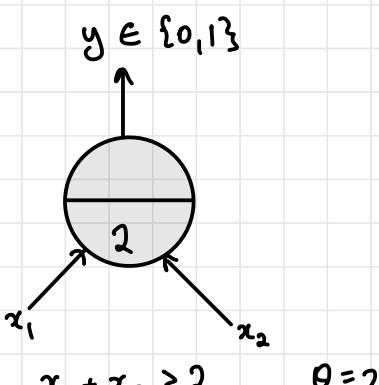




$x_1 + x_2 \geq 1$ $\quad \theta = 1$

(b) AND

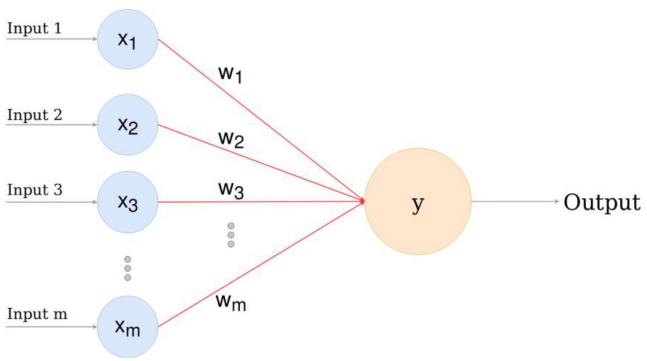




$x_1 + x_2 \geq 2$ $\quad \theta = 2$

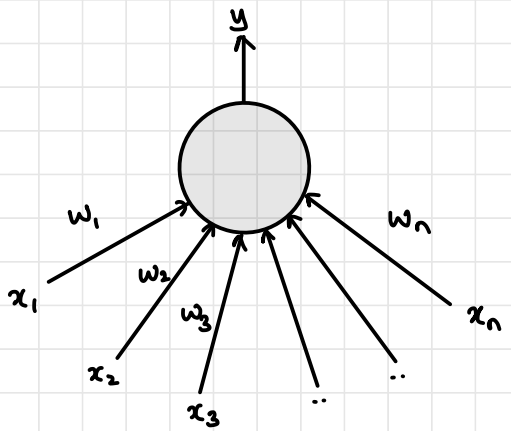
Perceptron

- More general – proposed by Frank Rosenblatt



The Perceptron Model

- Every input has an associated weight
- Mechanism for learning the weights and biases
- Inputs are numerical, not boolean



$$y = 1 \text{ if } \sum_{i=1}^{n} w_i x_i - \theta \geq 0$$

bias is threshold

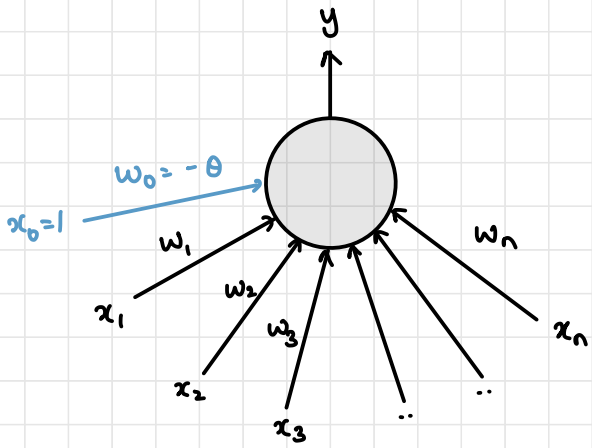
$$= 0 \text{ if } \sum_{i=1}^{n} w_i x_i - \theta < 0$$

A Mathematical Convention

$y = 1$ if $\sum_{i=0}^{n} w_i x_i \geq 0$

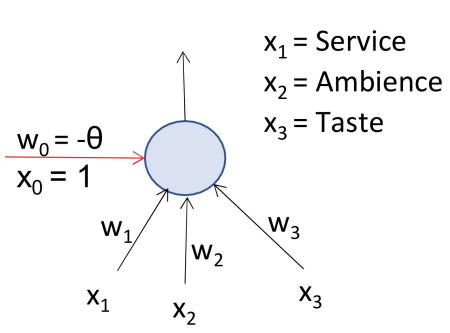
$= 0$ if $\sum_{i=0}^{n} w_i x_i < 0$

Where $w_0 = -\theta$ and $x_0 = 1$
(bias)



An Intuitive Example

- Setting a threshold: eating at a restaurant



- Threshold θ maybe low or high depending on the customer's standards

Perceptron Function

$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975

Comparing MP Neurons and Perceptrons

Differences

- Weights and threshold can be learnt for perceptron
- Inputs for perceptron maybe real-valued

Similarities

- Separate input space into 2 classes
- All inputs that give 1 lie on one side and those that give 0 lie on the other side
- Output is binary

Logical OR

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

w_0 = bias

w_1, w_2 } unknown

Negative:

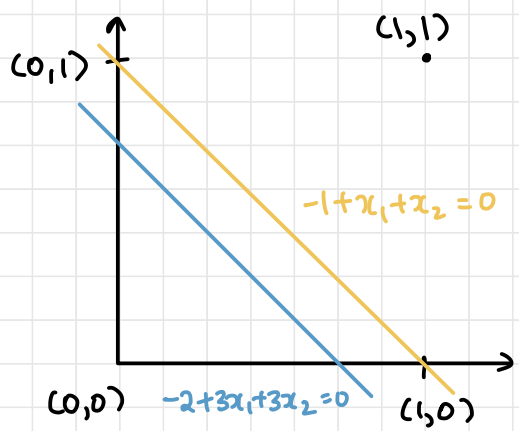
$$w_0 x_0 + w_1 x_1 + w_2 x_2 < 0$$

$$w_0 < 0 \longrightarrow (1)$$

Positive

$$w_0 x_0 + w_1 x_1 + w_2 x_2 \geq 0$$

$$w_0 + w_1 \geq 0 \longrightarrow (2)$$
$$w_0 + w_2 \geq 0 \longrightarrow (3)$$
$$w_0 + w_1 + w_2 \geq 0 \longrightarrow (4)$$



Possible solution: $w_0 = -1$, $w_1 = 1$, $w_2 = 1$

Perceptron equation:

$$\hat{y} = w_0 x_0 + w_1 x_1 + w_2 x_2$$

$$\hat{y} = -1 + x_1 + x_2 \quad : \quad y = \begin{cases} 1, & \hat{y} \geq 0 \\ 0, & \hat{y} < 0 \end{cases}$$

Another possibility

$$w_0 = -2, \; w_1 = 3, \; w_2 = 3 \qquad -2 + 3x_1 + 3x_2 = 0$$

Logical AND

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

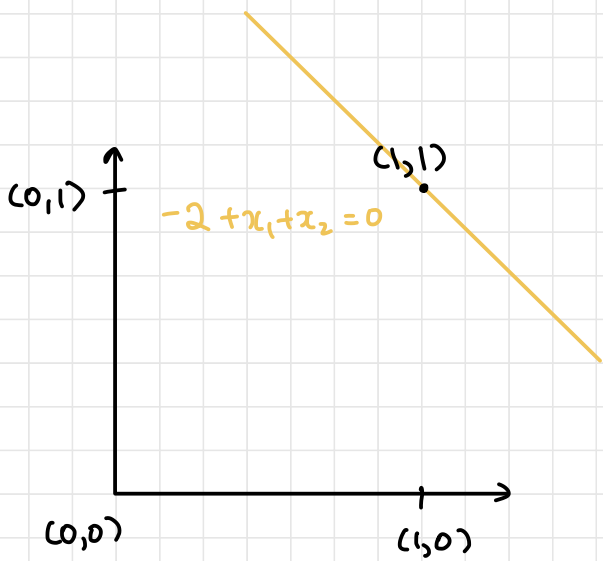
w_0 = bias

w_1, w_2 } unknown

Negative:

$$w_0 x_0 + w_1 x_1 + w_2 x_2 < 0$$

$$w_0 < 0 \rightarrow ①$$
$$w_0 + w_1 < 0 \rightarrow ②$$
$$w_0 + w_2 < 0 \rightarrow ③$$



Positive

$$w_0 x_0 + w_1 x_1 + w_2 x_2 \geq 0$$

$$w_0 + w_1 + w_2 \geq 0 \rightarrow ④$$

Possible solution: $w_0 = -2, \; w_1 = 1, \; w_2 = 1$

Perceptron Learning Algorithm

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

source: (Lecture 2)
http://www.cse.iitm.ac.in/~miteshk/CS6910.html

Check YouTube: CS7015 IITM Deep Learning

Lecture 2.1 to 2.7

https://www.youtube.com/watch?v=4TC5s_xNKSs&list=PLZEpw4xZOspJEn5s4wUj6uA1mw2_dk0ap

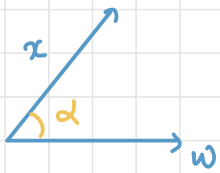
Working of Algorithm

- Initialise weights w with random vector
- Iterate over all positive & negative training examples (P ∪ N)
- If an input x belongs to P (positive training class), the value of the dot product $w \cdot x$ should be ≥ 0
  - So that perceptron predicts positive class
- If an input $x \in N$, the value of the dot product $w.x$ should be < 0
  - So that perceptron predicts negative class
- Case 1: $x \in P$ and $w.x < 0$: increase the values in w so next iteration of $w.x$ might be ≥ 0
- Case 2: $x \in N$ and $w.x \geq 0$: decrease the values in w so next iteration of $w.x$ might be < 0

Why it Works

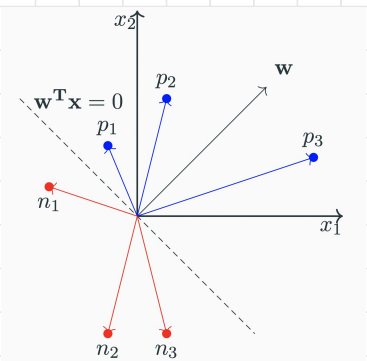
- If $x \in P$ and $w.x \geq 0$, then the angle between w & x should be $\leq 90^\circ$

$$w.x = |w||x| \cos\alpha$$



$\alpha \leq 90^\circ \Rightarrow \cos\alpha \geq 0$

- Similarly, if $x \in N$, $w.x < 0$ and $\alpha > 90^\circ$



source: IITM CS6910

Intuition for why the update works

(α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$	(α_{new}) when $\mathbf{w_{new}} = \mathbf{w} - \mathbf{x}$
$cos(\alpha_{new}) \propto \mathbf{w_{new}}^T\mathbf{x}$	$cos(\alpha_{new}) \propto \mathbf{w_{new}}^T\mathbf{x}$
$\propto (\mathbf{w} + \mathbf{x})^T\mathbf{x}$	$\propto (\mathbf{w} - \mathbf{x})^T\mathbf{x}$
$\propto \mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{x}$	$\propto \mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{x}$
$\propto cos\alpha + \mathbf{x}^T\mathbf{x}$	$\propto cos\alpha - \mathbf{x}^T\mathbf{x}$
$cos(\alpha_{new}) > cos\alpha$	$cos(\alpha_{new}) < cos\alpha$
α is decreasing	α is increasing

Logical XOR

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

w_0 = bias

w_1, w_2 } unknown

Negative:

$$w_0 x_0 + w_1 x_1 + w_2 x_2 < 0$$

$$w_0 < 0 \rightarrow (1)$$

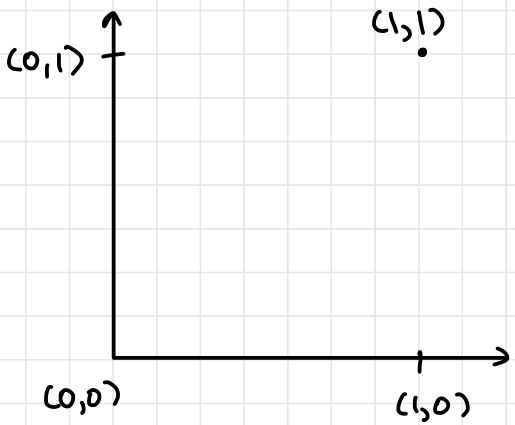
$$w_0 + w_1 + w_2 < 0 \rightarrow (2)$$

Positive

$$w_0 x_0 + w_1 x_1 + w_2 x_2 \geq 0$$

$$w_0 + w_1 \geq 0 \rightarrow (3)$$

$$w_0 + w_2 \geq 0 \rightarrow (4)$$



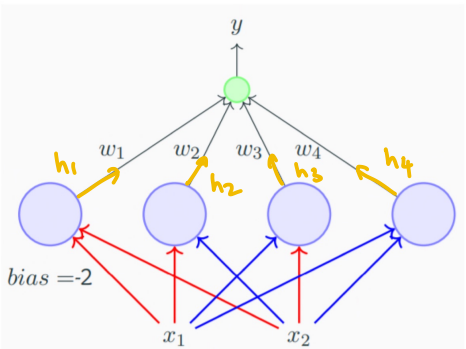
No solution! Contradiction

Single Perceptron

- Fails when data linearly inseparable
- Linear decision surface
- Problems: Prof Preet's slides (L2.2)

Multi-Layer Perceptron Network - ANN

- If data is not linearly separable, we can use a network of neurons
- Assume input True = +1, False = -1 (not 1 & 0)



Red edge: **w = -1**
Blue edge: **w = +1**

- Can implement any Boolean function
- https://www.youtube.com/watch?v=CJr4Dst0uZE&list=PLZEpw4xZOspJEn5s4wUj6uA1mw2_dk0ap&index=17

very good video ↗

- Perceptron 1 fires when $x_1 = -1$ & $x_2 = -1$
  Perceptron 2 fires when $x_1 = -1$ & $x_2 = 1$
  Perceptron 3 fires when $x_1 = 1$ & $x_2 = -1$
  Perceptron 4 fires when $x_1 = 1$ & $x_2 = -1$

XOR Using Multi-layered Perceptron

x_1	x_2	y	h_1	h_2	h_3	h_4	$\hat{y} = \sum_{i=0}^{4} w_i h_i$
-1	-1	0	1	0	0	0	$w_0 + w_1$
-1	1	1	0	1	0	0	$w_0 + w_2$
1	-1	1	0	0	1	0	$w_0 + w_3$
1	1	0	0	0	0	1	$w_0 + w_4$

Positive

$w_0 + w_2 \geq 0$

$w_0 + w_3 \geq 0$

Negative

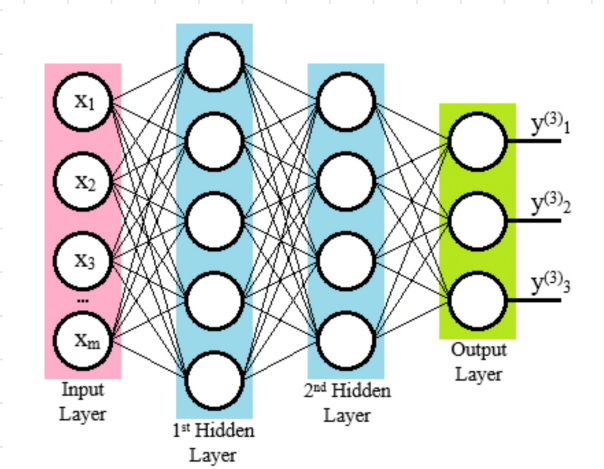
$w_0 + w_1 < 0$

$w_0 + w_4 < 0$

$w_0 = -1$, $w_1 = -1$, $w_2 = 1$, $w_3 = 1$, $w_4 = -1$

Multilayered ANN

- Deep ANN: more than 1 hidden layer



1) Input Layers

- attributes for one sample
- fed into network

2) Hidden Layers

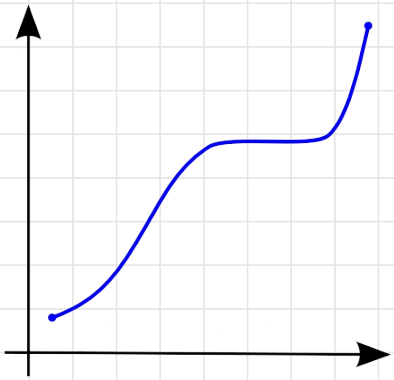
- responsible for deriving complex relationships between input and output

3) Output Layer

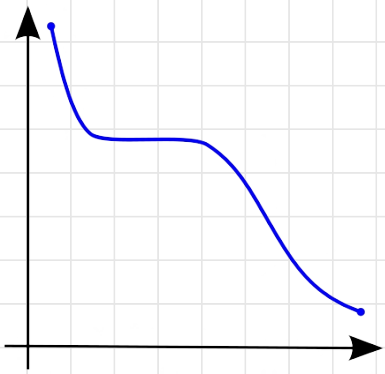
- emits output

Monotonic Functions

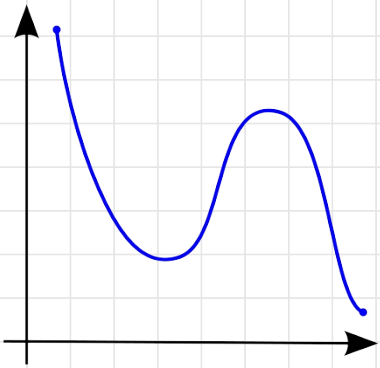
source: wikipedia



monotonically non-decreasing



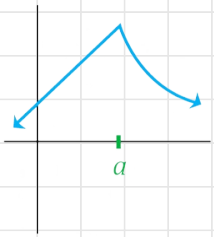
monotonically non-increasing



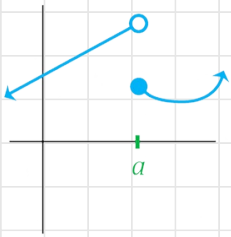
non-monotonic

Differentiability

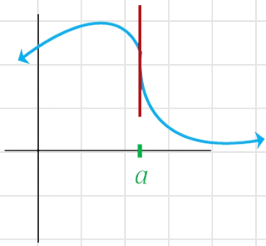
source: calc workshop



Cusp / Corner



Discontinuous



Vertical Tangent

ACTIVATION FUNCTIONS

- Non-linear transformation applied to input before propagating it to the next layer of neurons

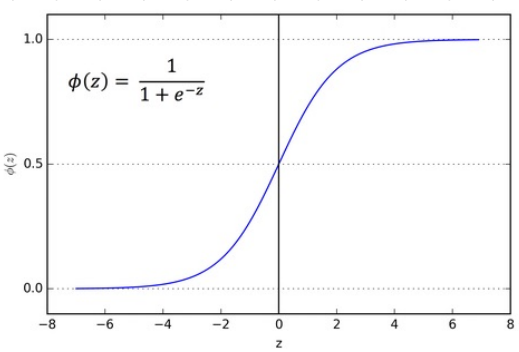
$$Y = \text{Activation}\left(\sum_{i=0}^{n} w_i x_i + b\right)$$

- If there are no activation functions, behaviour of NN is purely linear (linear regression)

- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

1. Sigmoid

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



source: towards data science

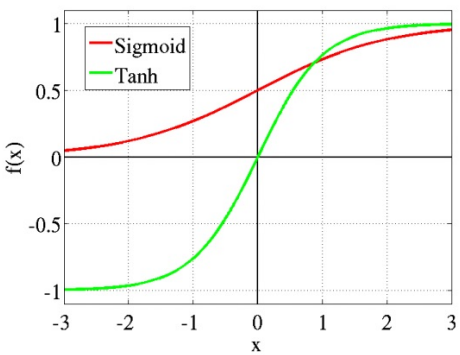
- Range : (0, 1)
- Differentiable, monotonic (not its derivative)
- Can cause NN to get stuck during training (problem of vanishing gradient)

2. tanh

$$f(x) = \tanh(x)$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- Range: $(-1, 1)$
- Better than sigmoid as negative maps to negative
- Differentiable, monotonic (not its derivative)



source: towards data science

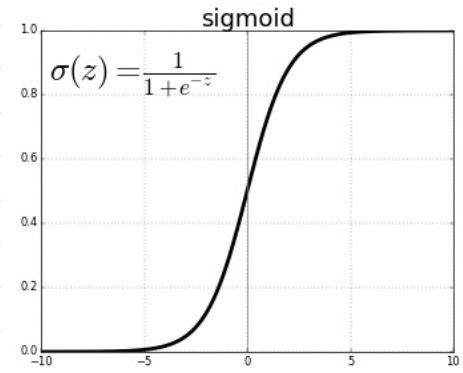
3. ReLU (Rectified Linear Unit)

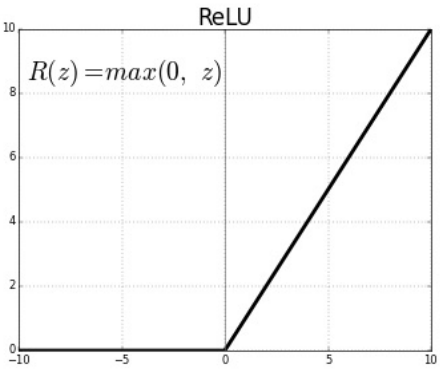
$$f(x) = \max(0, x)$$

$$f(x) = \begin{cases} x & , x > 0 \\ 0 & , x \leq 0 \end{cases}$$

$$f'(x) = \begin{cases} 1 & , x > 0 \\ 0 & , x \leq 0 \end{cases}$$

- Defined to be differentiable even though it is not at $x = 0$



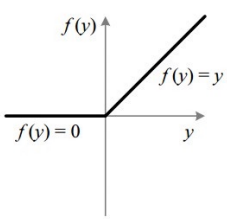


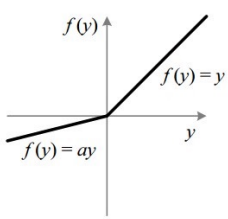
source: towards data science

- Range: $[0, \infty)$
- Does not map negative values properly – dying ReLU problem

4. Leaky ReLU

- Leak increases range and tries to solve dying ReLU problem





source: towards data science

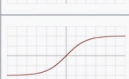
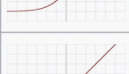
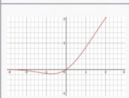
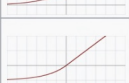
$$f(x) = \begin{cases} x & , x > 0 \\ ax & , x \leq 0 \end{cases}$$

- a is usually 0.01

$$f'(x) = \begin{cases} 1 & , x > 0 \\ a & , x \leq 0 \end{cases}$$

- Defined to be differentiable even though it is not at $x = 0$

- Cheatsheet (at $x = 0$, arbitrary decisions made)

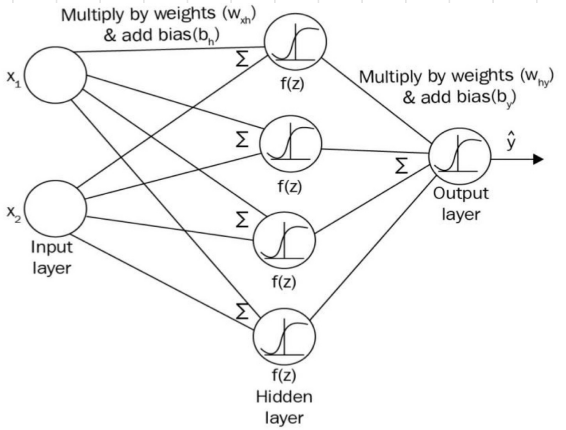
Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range
Identity		x	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f(x)(1 - f(x))$	$(0, 1)$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[9]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Gaussian Error Linear Unit (GELU)[4]		$\frac{1}{2}x\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right) = x\Phi(x)$	$\Phi(x) + x\phi(x)$	$(-0.17\ldots, \infty)$
Softplus[10]		$\ln(1 + e^x)$	$\frac{1}{1 + e^{-x}}$	$(0, \infty)$
Exponential linear unit (ELU)[11]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α	$\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$(-\alpha, \infty)$
Scaled exponential linear unit (SELU)[12]		$\lambda\begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$	$\lambda\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$
Leaky rectified linear unit (Leaky ReLU)[13]		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\infty, \infty)$

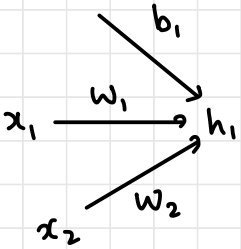
source: Wikipedia

https://en.wikipedia.org/wiki/Activation_function

Forward Propagation

- Total no. of layers in ANN is no. of hidden layers + output layers



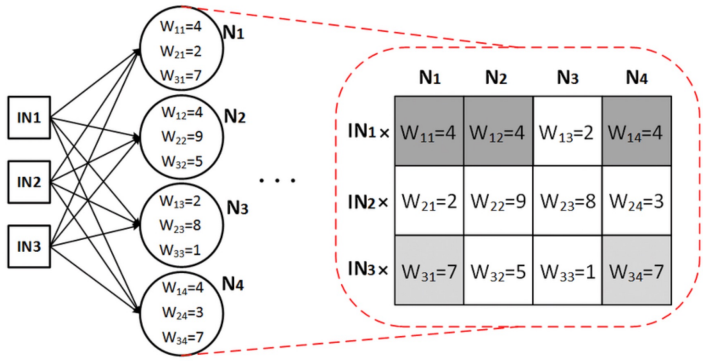


$$h_1 = w_1 x_1 + w_2 x_2 + b_1$$

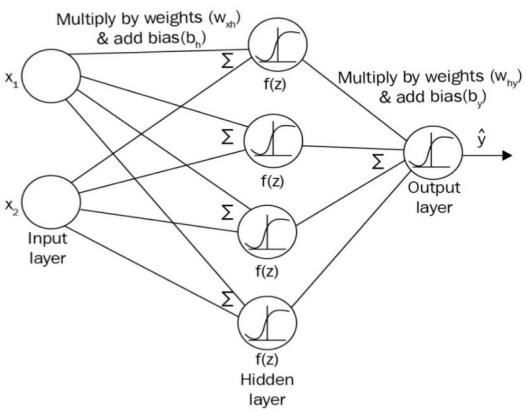
output = activation(h_1)

- Weight matrix : weights from previous layer to current layer
- Dimensions = no. of units in prev layer x no. of units in current layer

Calculation of Weight Matrix



Forward Propagation

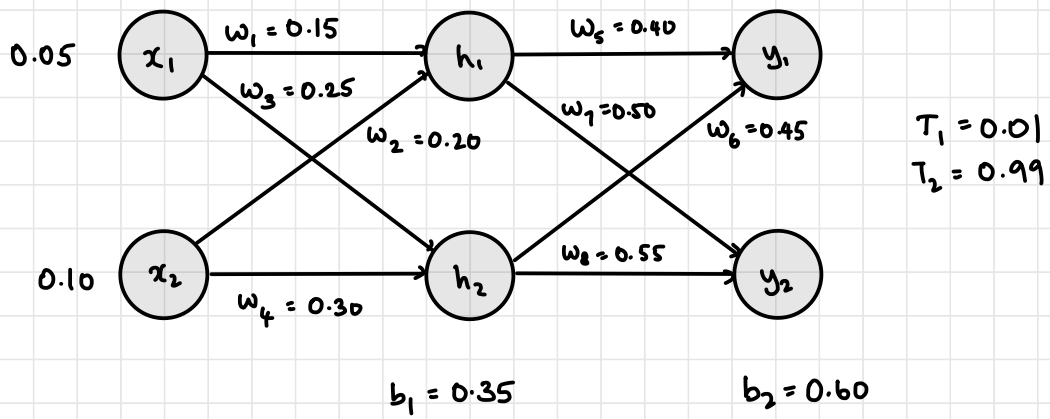


- w_{xh} : weight between input and hidden layer
- b_h : bias of hidden layer
- $z_1 = x \cdot w_{xh} + b_h$: netsum
- $a_1 = \sigma(z_1)$: activation function : output of hidden layer

- w_{hy} : weight between hidden layer and output layer
- b_y : bias of output layer
- $z_2 = a_1 \cdot w_{hy} + b_y$: netsum
- $\hat{y} = \sigma(z_2)$: sigmoid activation : final output
- Loss : error function

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

Example



$$h_1 = (x_1 w_1 + x_2 w_2 + b_1) = (0.05 \times 0.15 + 0.10 \times 0.20 + 0.35) = 0.3775$$

$$h_2 = (x_1 w_3 + x_2 w_4 + b_1) = (0.05 \times 0.25 + 0.10 \times 0.30 + 0.35) = 0.3925$$

$$O_{h_1} = a(h_1) = \frac{1}{1 + e^{-0.3775}} = 0.5933$$

$$O_{h_2} = a(h_2) = \frac{1}{1 + e^{-0.3925}} = 0.5969$$

$$\hat{y}_1 = a(o_{h_1} w_5 + o_{h_2} w_6 + b_2) = a(0.5933 \times 0.40 + 0.5969 \times 0.45 + 0.60)$$
$$= a(1.1059)$$
$$= 0.7514$$

$$\hat{y}_2 = a(o_{h_1} w_7 + o_{h_2} w_8 + b_2) = a(0.5933 \times 0.5 + 0.5969 \times 0.55 + 0.60)$$
$$= a(1.2249)$$
$$= 0.7729$$

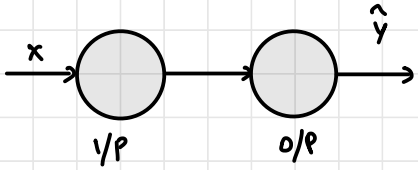
$$\text{Loss} = \frac{1}{2} \sum (y - \hat{y})^2 \qquad [\text{MSE}]$$

$$= \frac{1}{2} \left((0.7514 - 0.01)^2 + (0.7729 - 0.99)^2 \right)$$

$$= 0.2984$$

Backpropagation

- Iteratively modify weights to minimize loss function between actual and predicted values



$$L = \frac{1}{2} (\hat{y} - y)^2$$

$$\hat{y} = a(z) = \frac{1}{1 + e^{-z}} = (1 + e^{-z})^{-1}$$

$$z = wx + b$$

Review of Vector Calculus

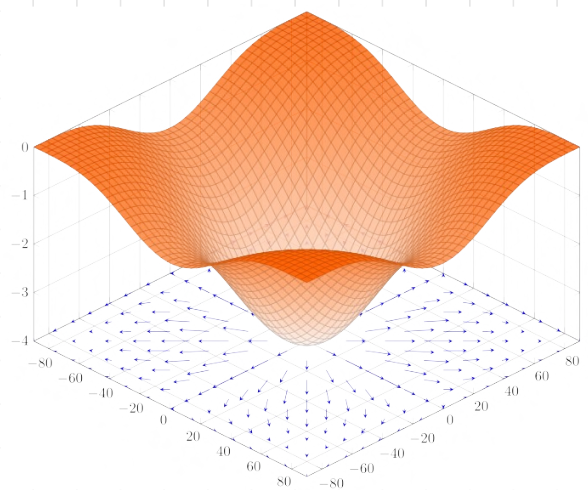
nabla; del operator → $\nabla = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k}$

- If ϕ is a scalar function $\phi(x, y, z)$, then $\nabla\phi$ is the gradient of ϕ

$$\nabla\phi = \frac{\partial\phi}{\partial x}\hat{i} + \frac{\partial\phi}{\partial y}\hat{j} + \frac{\partial\phi}{\partial z}\hat{k}$$

- If $\phi = f(x, y, z)$, $\nabla\phi$ is a vector that points in the direction of the steepest slope

- Gradient of $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a projected vector field on the bottom plane



source: wikipedia

$$\nabla f = \frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j}$$

GRADIENT DESCENT

- Minimising the cost function by taking steps in the opposite direction of the gradient

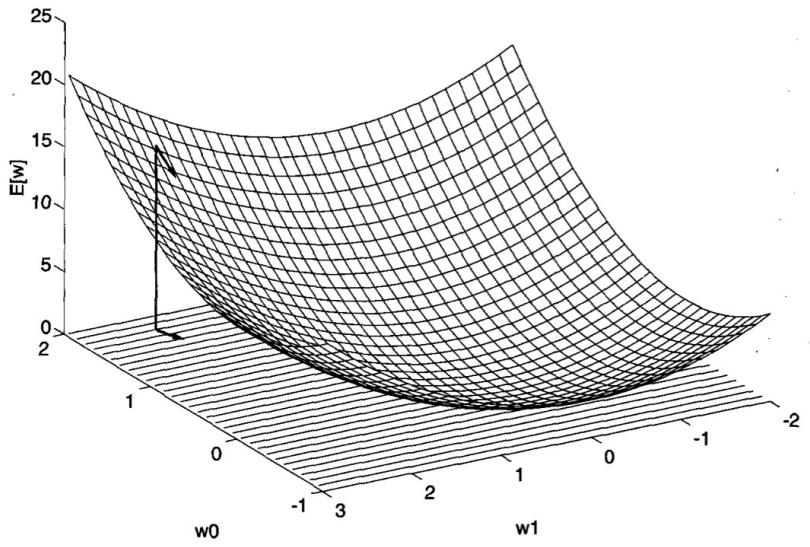


FIGURE 4.4
Error of different hypotheses. For a linear unit with two weights, the hypothesis space H is the w_0, w_1 plane. The vertical axis indicates the error of the corresponding weight vector hypothesis, relative to a fixed set of training examples. The arrow shows the negated gradient at one particular point, indicating the direction in the w_0, w_1 plane producing steepest descent along the error surface.

T2

- At each step, weight vector is altered in the direction that produces steepest descent along the error surface
- Gradient of error function wrt weight vector points in the direction of steepest ascent
- $-\nabla E$ in direction of steepest descent

Gradient Descent

- The weights are updated according to the rule

$$\vec{w} = \vec{w} + \Delta\vec{w}$$

- Where $\Delta\vec{w}$ is given by

$$\Delta\vec{w} = -\eta \nabla E(\vec{w})$$

- η : positive constant called the learning rate
- Component-Wise

$$w_i = -\eta \frac{\partial E}{\partial w_i}$$

MSE / LSE Loss Function

- For a single output unit

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- D — set of training examples
- t_d — target (expected) value
- o_d — output (predicted) value
- $\frac{1}{2}$ — to simplify calculations

- Multiple output units

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Cross-Entropy Loss Function

$$E = \sum_{i=1}^{n} -t_i \ln(o_i)$$

- n = no. of classifications
- $o_i = a(z)$
- $z = wx + b$

Binary Cross-Entropy

$$L = -[y \ln(\hat{y}) + (1-y) \ln(1-\hat{y})]$$ } both notations used

$$\hat{y} = a(z)$$

$$z = wx + b$$

Derivative of L wrt w

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$

$$\frac{\partial L}{\partial w} = \left(\frac{-y}{\hat{y}} + \frac{(1-y)}{1-\hat{y}} \right) \frac{\partial \hat{y}}{\partial w}$$

$$= \frac{\hat{y} - y}{\hat{y}(1-\hat{y})} \underbrace{\frac{\partial \hat{y}}{\partial w}}$$

$$\frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w} = a(z)(1-a(z))\, x \quad \text{(sigmoid)}$$

$$\frac{\partial L}{\partial w} = \frac{\hat{y} - y}{\hat{y}(1-\hat{y})} a(z)(1-a(z))\, x$$

Backpropagation Algorithm

- Employs gradient descent to minimise MSE

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Terminology

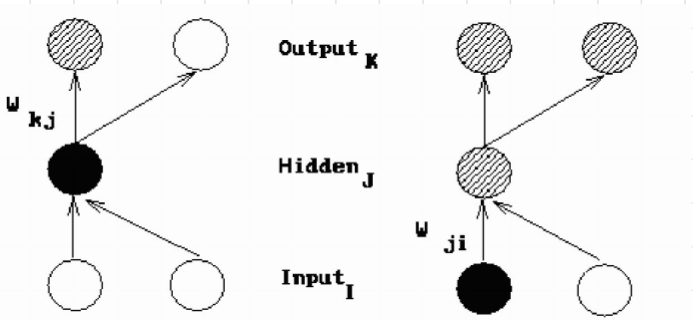
- x_{ji} = the ith input to unit j
- w_{ji} = the weight associated with the ith input to unit j
- $net_j = \sum_i w_{ji} x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output computed by unit j
- t_j = the target output for unit j
- σ = the sigmoid function
- $outputs$ = the set of units in the final layer of the network
- $Downstream(j)$ = the set of units whose immediate inputs include the output of unit j

T2

Derivation of SGD

- For sigmoid activation functions

https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf



$$w_{ji} = w_{ji} + \Delta w_{ji}$$

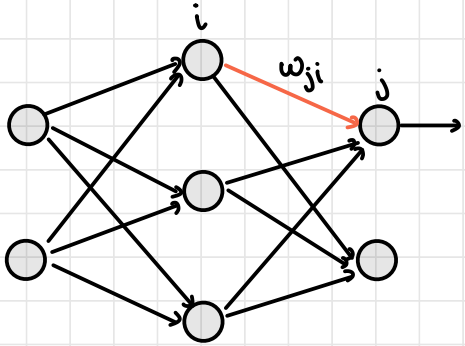
$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

Case 1: Training Rule for Output Unit Weights

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \underbrace{\frac{\partial E_d}{\partial o_j}}_{(a)} \underbrace{\frac{\partial o_j}{\partial net_j}}_{(b)} \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{(c)} \rightarrow (1)$$



(a) $$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

$$= \frac{1}{2} \sum_{k \in outputs} \frac{\partial}{\partial o_j} (t_k - o_k)^2$$

only when $j = k$, $\frac{\partial}{\partial o_j} (t_k - o_k)^2 \neq 0$

$$\frac{\partial E_d}{\partial o_j} = \frac{1}{2} \frac{\partial}{\partial o_j} (t_j - o_j)^2$$

$$\frac{\partial E_d}{\partial o_j} = -(t_j - o_j) \longrightarrow (2)$$

(b) $$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j}$$

Derivative of $\sigma(x)$

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

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

$$\frac{1}{\sigma} \frac{d\sigma}{dx} = \frac{1}{1+e^{-x}} \cdot e^{-x} = \frac{e^{-x}+1}{1+e^{-x}} \quad \frac{-1}{1+e^{-x}}$$

$$\frac{d\sigma}{dx} = \sigma(1-\sigma)$$

$$\frac{\partial o_j}{\partial net_j} = o_j(1-o_j) \longrightarrow (3)$$

Combining (2) & (3)

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1-o_j) \longrightarrow (4)$$

$$\delta_j = (t_j - o_j)\, o_j(1-o_j)$$

(c) $$\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial \sum_{i \in I} w_{ji} x_{ji}}{\partial w_{ji}} = x_{ji} \longrightarrow (5)$$

Combining (4) & (5) in (1)

$$\frac{\partial E_d}{\partial w_{ji}} = -(t_j - o_j) o_j (1 - o_j) x_{ji}$$

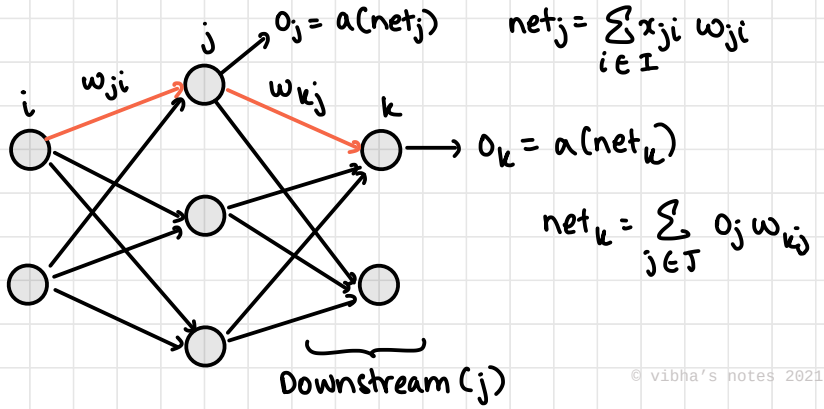
Weight Updation

$$\Delta w_{ji} = \eta (t_j - o_j) o_j (1 - o_j) x_{ji} = \eta \delta_j x_{ji}$$

Case 2: Training Rule for Hidden Unit weights

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \sum_{k \in Downstream(j)} \underbrace{\frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}}_{(a)} \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{(b)} \longrightarrow (6)$$



(a) $$\frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \longrightarrow (7)$$

where $\frac{\partial E_d}{\partial net_k} = -\delta_k$

$$= -\delta_k w_{kj} o_j (1-o_j)$$

(b) $$\frac{\partial net_j}{\partial w_{ji}} = x_{ji} \longrightarrow (8)$$

Combining (7) & (8) in (6)

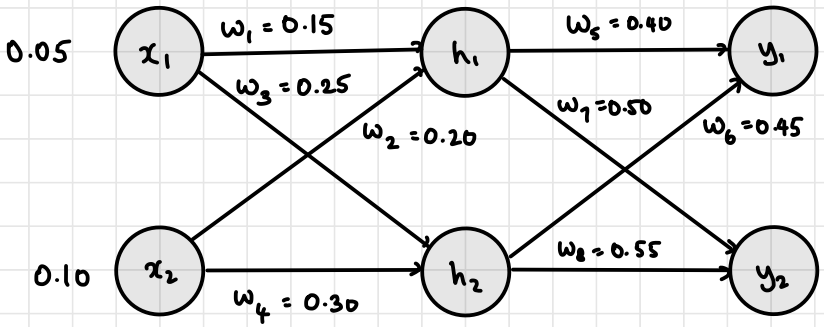
$$\frac{\partial E_d}{\partial w_{ji}} = \left(\sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1-o_j) \right) x_{ji}$$

Weights Updation

$$\Delta w_{ji} = \eta x_{ji} \left(\sum_{k \in Downstream(j)} \delta_k w_{kj} o_j (1-o_j) \right)$$

where $\delta_k = (t_k - o_k) o_k (1-o_k)$

Q: Backpropagate once



$T_1 = 0.01$

$T_2 = 0.99$

$\eta = 0.5$ $\quad b_1 = 0.35$ $\quad b_2 = 0.60$

$\hat{y}_1 = 0.7514$

$\hat{y}_2 = 0.7729$

Loss = 0.2984

$$O_{h_1} = a(h_1) = \frac{1}{1+e^{-0.3775}} = 0.5933$$

$$O_{h_2} = a(h_2) = \frac{1}{1+e^{-0.3925}} = 0.5969$$

$$w_5 = w_5 + \Delta w_5$$

$$= w_5 + \eta (0.01 - 0.7514)(0.7514)(1-0.7514)\,0.5933$$

$$= 0.40 - \eta\,(0.08216)$$

$$w_5 = 0.3589$$

$$w_6 = w_6 + \Delta w_6$$

$$= w_6 + \eta (0.01 - 0.7514)(0.7514)(1 - 0.7514)(0.5969)$$

$$= 0.45 + \eta (-0.0827)$$

$$w_6 = 0.4087$$

$$w_7 = w_7 + \Delta w_7$$

$$= w_7 + \eta (0.99 - 7729)(0.7729)(1 - 0.7729)(0.5933)$$

$$= 0.50 + 0.1130$$

$$= 0.5113$$

$$w_8 = w_8 + \Delta w_8$$

$$= w_6 + \eta (0.99 - .7729)(0.7729)(1 - 0.7729)(0.5969)$$

$$= 0.55 + 0.1137$$

$$= 0.5614$$

$$w_1 = w_1 + \Delta w_1$$

$$\Delta w_1 = \eta \left[\sum_{k \in Down(j)} (t_k - o_k) o_k (1 - o_k) w_{kj} \right] x_{ji} \; o_j (1 - o_j)$$

$$w_1 = 0.15 + 0.5 \left[(0.01 - 0.7514)(0.7514)(1 - 0.7514)(0.40) + (0.99 - 0.7729)(0.7729)(1 - 0.7729)(0.50) \right] 0.05 (0.5933)(1 - 0.5933)$$

$$w_1 = 0.15 + 0.5 \times 0.01206 \times -0.0363$$

$$w_1 = 0.1498$$

$$w_2 = 0.20 + 0.5 \left[\begin{array}{l} (0.01 - 0.7514)(0.7514)(1-0.7514)(0.40) + \\ (0.99 - 0.7729)(0.7729)(1-0.7729)(0.50) \end{array} \right] 0.10$$
$$(0.5933)(1-0.5933)$$

$$w_2 = 0.20 + 0.01206 \times -0.0363$$

$$w_2 = 0.1996$$

-0.0414

$$w_3 = 0.25 + 0.5 \left[\begin{array}{l} (0.01 - 0.7514)(0.7514)(1-0.7514)(0.45) + \\ (0.99 - 0.7729)(0.7729)(1-0.7729)(0.55) \end{array} \right] 0.05$$
$$(0.5969)(1-0.5969)$$

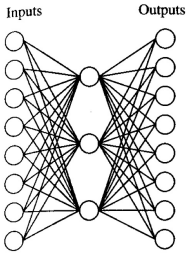
$$w_3 = 0.2498$$

$$w_4 = 0.30 + 0.5 \left[\begin{array}{l} (0.01 - 0.7514)(0.7514)(1-0.7514)(0.45) + \\ (0.99 - 0.7729)(0.7729)(1-0.7729)(0.55) \end{array} \right] 0.10$$
$$(0.5969)(1-0.5969)$$

$$w_4 = 0.2995$$

Overfitting

- Fitting of noise
- Solution: validation set



Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

FIGURE 4.7
Learned Hidden Layer Representation. This $8 \times 3 \times 8$ network was trained to learn the identity function, using the eight training examples shown. After 5000 training epochs, the three hidden unit values encode the eight distinct inputs using the encoding shown on the right. Notice if the encoded values are rounded to zero or one, the result is the standard binary encoding for eight distinct values.

- Look up: Universal Approximation Theorem

Gradient Descent Algorithm

- Repeat until convergence (gradient = 0 at min)

$$w_j := w_j - \frac{\alpha \times \partial E(w_0, w_1)}{\partial w_j}$$

for j = 0, 1

- Update weights

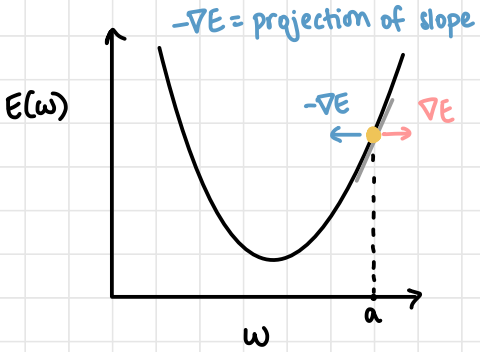
α = learning rate

$$temp_0 = w_0 - \left(\frac{\alpha \times \partial J(w_0, w_1)}{\partial w_0}\right)$$

$$temp_1 = w_1 - \left(\frac{\alpha \times \partial J(w_0, w_1)}{\partial w_0}\right)$$

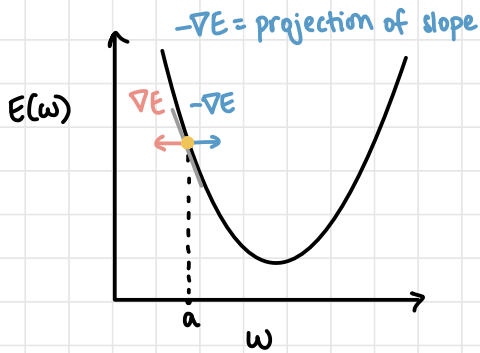
$$w_0 = temp_0$$
$$w_1 = temp_1$$

Visualization in 2D



$$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

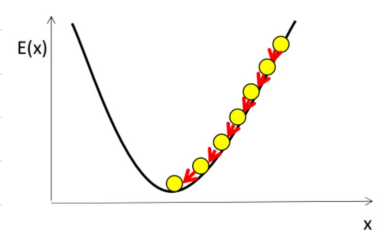
$w = w - \alpha$(positive number)



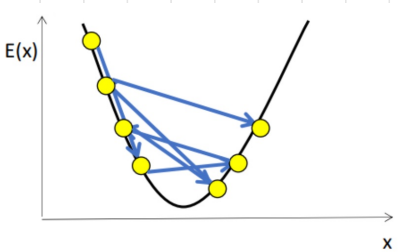
$$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$w = w - \alpha$(negative number)

Effect of Learning Rate α



too small

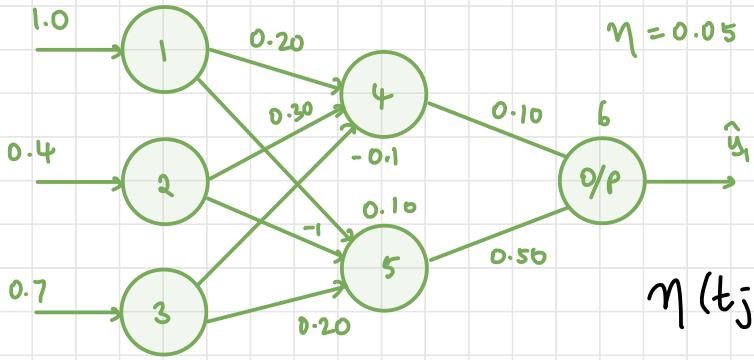


too large

Drawback of GD

- Can get stuck in local minimum

Q: Perform one pass of FP and one of BP (assume σ)



$$\eta (t_j - o_j) o_j (1 - o_j) x_{ji}$$

$$x = \begin{bmatrix} 1.0 \\ 0.4 \\ 0.7 \end{bmatrix} \quad w = \begin{bmatrix} 0.2 & 0.10 \\ 0.3 & -1 \\ -0.1 & 0.20 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{Net} = w^T x + b = \begin{bmatrix} 0.2 & 0.3 & -0.1 \\ 0.10 & -1 & 0.20 \end{bmatrix} \begin{bmatrix} 1.0 \\ 0.4 \\ 0.7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 \\ -0.16 \end{bmatrix}$$

$$z = \sigma(x) = \begin{bmatrix} 0.5622 \\ 0.4601 \end{bmatrix}$$

$$net_2 = w_2^T z + b_2 = \begin{bmatrix} 0.10 & 0.50 \end{bmatrix} \begin{bmatrix} 0.5622 \\ 0.4601 \end{bmatrix} = 0.2863$$

$$\hat{y} = 0.5711 = o_j$$

$$y = 0.65$$

$$\text{Loss} = \frac{1}{2} (0.65 - 0.5711)^2 = 3.113 \times 10^{-3}$$

Backpropagation

$$w_{64} = w_{64} + \Delta w$$

$$= 0.10 + 0.05 \left((t_j - o_j)\, o_j\, (1 - o_j)\, x_{ji} \right)$$

$$= 0.10 + 0.05\ (0.65 - 0.5711)\ 0.5711\ (0.5622)$$

$$= 0.10 + 0.05 \times 0.0253$$

$$= 0.101$$

$$w_{65} = w_{65} + \Delta w$$

$$= 0.50 + 0.05\ (0.65 - 0.5711)\ 0.5711\ (0.4601)$$

$$= 0.5010$$

$$w_{41} = 0.20 + \Delta w$$

$$= 0.20 + 0.05 \left[\sum_{k \in DS} \left[(t_k - o_k)\, o_k\, (1 - o_k)\, w_{kj} \right] x_{ji}\ (1 - o_j)\, o_j \right]$$

$$= 0.20 + 0.05 \left[(0.65 - 0.5711)\ 0.5711\ (1 - 0.5711)\ (0.10) \right.$$

$$\left. (1.0)\,(1 - 0.5622)\ 0.5622 \right]$$

$$= 0.20 + 2.3784 \times 10^{-5}$$

$$= 0.2000$$

$$w_{42} = w_{42} + \Delta w$$

$$= 0.30 + 0.05 \left[\sum_{k \in DS} \left[(t_k - o_k) o_k (1 - o_k) w_{kj} \right] x_{ji} (1 - o_j) o_j \right]$$

$$= 0.30 + 0.05 \left[(0.65 - 0.5711) 0.5711 (1 - 0.5711) 0.1 \right]$$

$$(0.40) (1 - 0.5622) 0.5622$$

$$= 0.30 + 9.51 \times 10^{-6}$$

$$= 0.30$$

$$w_{43} = w_{43} + \Delta w$$

$$= -0.1 + 0.05 \left(\sum_{k \in DS} \left[(t_k - o_k) o_k (1 - o_k) w_{kj} \right] x_{ji} (1 - o_j) o_j \right)$$

$$= -0.1 + 0.05 \left[(0.65 - 0.5711) 0.5711 (1 - 0.5711) 0.1 \right]$$

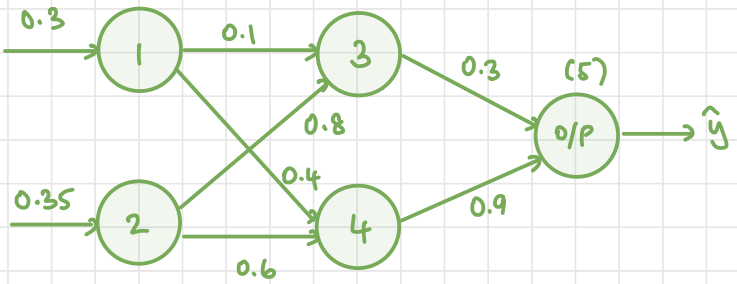
$$(0.70) (0.5622) (1 - 0.5622)$$

$$= -0.09998$$

and so on

Q: Perform one pass of FP and one of BP (assume σ) for w_{53} and w_{31}

$\eta = 0.5$



Actual $y = 0.73$

$$x = \begin{bmatrix} 0.3 \\ 0.35 \end{bmatrix} \quad w_1 = \begin{bmatrix} 0.1 & 0.4 \\ 0.8 & 0.6 \end{bmatrix} \quad x^T = [0.3 \quad 0.35]$$

$$net_2 = \begin{bmatrix} 0.31 \\ 0.33 \end{bmatrix} \quad a_2 = \begin{bmatrix} 0.5769 \\ 0.5818 \end{bmatrix} \quad o = 0.8261$$

Backpropagation

$$\Delta w_{53} = \eta (t_j - o_j) o_j (1 - o_j) x_{ji}$$

$$= 0.5 (0.73 - 0.8261) 0.8261 (1 - 0.8261) 0.5769$$
$$= -3.9822 \times 10^{-3}$$

$$w_{53} = 0.2960$$

$$\Delta w_{31} = \eta o_j (1 - o_j) x_{ji} \sum_{k \in \text{down}(j)} (t_k - o_k) o_k (1 - o_k) w_{kj}$$

$$= 0.5 (0.5769)(1 - 0.5769) 0.3 \left[(0.73 - 0.8261) 0.8261 (1 - 0.8261)(0.3) \right]$$

$$= -1.5164 \times 10^{-4}$$

$$w_{31} = 0.0998$$

Mathematical Foundations for SVMs

- Build models to map input to output data

- ML models → optimisation problem
  - minimisation of $f(x)$
  - maximisation of $f(x) \Rightarrow$ minimisation of $-f(x)$

  https://web.stanford.edu/group/sisl/k12/optimization/#!index.md

1. Unconstrained Optimisation

- Find x that minimises $f(x)$; global minima

- Stationary / critical point : where $f'(x) = 0$
  - local minimum
  - local maximum
  - saddle point

- Second derivative
  - $f''(x) < 0$: local maximum
  - $f''(x) > 0$: local minimum
  - $f''(x) = 0$: could be saddle point

HESSIAN MATRIX

- Let $f : R^n \rightarrow R$ with input vector $x \in R^n$ and output scalar $f(x) \in R$

- If all partial derivatives of f exist and are continuous, then $H_{n \times n}$ is defined as

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

source: wikipedia

- or $(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i\,\partial x_j}$.
- For 2 variables

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

- x is a vector ⇒ must find gradient
- If H_{ij} is positive definite: local min
  If H_{ij} is negative definite: local max
  If H_{ij} is indefinite: saddle point

Q: Find the Hessian of $f(x,y) = 8x - 2x^2y^2$. Also find gradient.

$$\nabla f = (8 - 4xy^2)\hat{i} - 4x^2y\hat{j} = \begin{bmatrix} 8 - 4xy^2 & -4x^2y \end{bmatrix}$$

$$H = \begin{bmatrix} -4y^2 & -8yx \\ -8yx & -4x^2 \end{bmatrix}$$

2. Constrained Optimisation

- Constraints place limits on the domain of the variables
- Eg: marathon runner

$$L(x,\lambda,\alpha) = f(x) + \sum_i \lambda_i g^i(x) + \sum_j \alpha_j h^{(j)}(x)$$

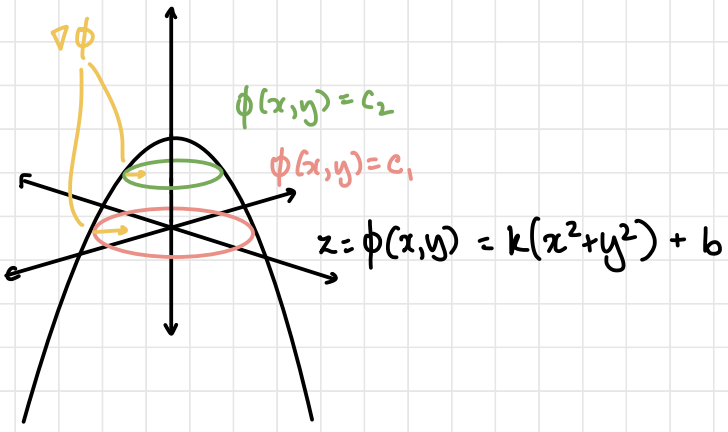
- Constrained minimum

$$\min_{x \in S} f(x) \quad \min_{x} \max_{\lambda} \max_{\alpha, \alpha \geq 0} L(x,\lambda,\alpha)$$

- Constraint surface places limit on value of function to be optimised

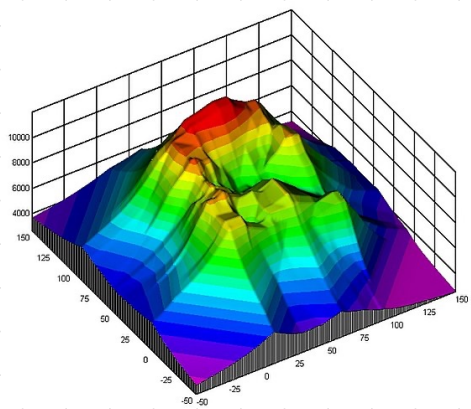
Review of Vector Calculus (part II)

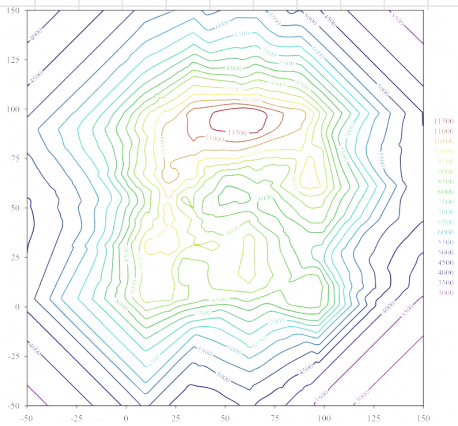
- If $\phi(x,y,z)$ is a function, $\nabla\phi$ = the direction of max ascent (a vector)



$\nabla\phi$ is radial to the circles $x^2+y^2=c$

- On a level surface $\phi(x,y,z) = c$, the gradient at any point is perpendicular to the surface
- This is because the value of $\nabla\phi$ along the surface is 0 (since $\phi(x,y,z)$ is constant)
- Level surfaces in higher dimensions are represented as contours





source: wikipedia

Minimization Problem

- To minimize value of a function $f(\vec{x})$ subject to the constraints $g(\vec{x}) \leq 0$ (boundary — level surface) called the feasible region
- Start with $\vec{x}$ lying on the constraint boundary $g(\vec{x}) = 0$

Direction Requirements

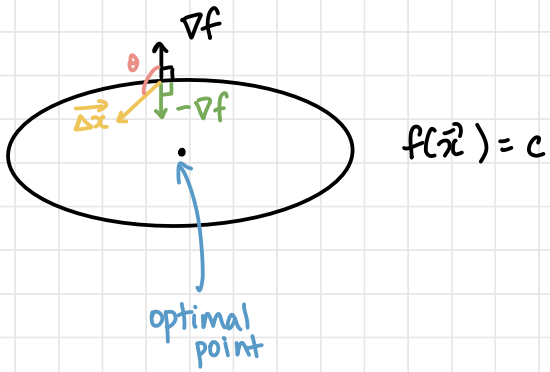
1. Descent Direction

- If a vector decreases the value of the objective function $f(\vec{x})$, it is said to move in the descent direction

- Test if vector $\vec{\Delta x}$ is moving in descent direction
  - ↳ Find $\nabla f(\vec{x})$
  - ↳ Find $\nabla f(\vec{x}) \cdot \vec{\Delta x}$

- If the dot product is ≤ 0, the vector is moving in descent direction

- Reason: if $\vec{a} \cdot \vec{b}$ is negative, θ is obtuse. This means angle between $\vec{\nabla f}$ and $\vec{\Delta x}$ is obtuse and $\vec{\Delta x}$ reduces the value of $f(\vec{x})$



Q: $f(x) = 2x_1 + 3x_1x_2^2$

Tell if the following vectors are in descent direction starting from $(4, -2)$

$$\nabla(x) = [2 + 3x_2^2 \quad 6x_1x_2] \quad \text{at } (4,-2), \quad [14 \quad -48]$$

(a) $[2 \quad 1]$

$[2 \quad 1] \cdot [14 \quad -48] = 28 - 48 = -20$: descent

(b) $[2 \quad -1]$

$[2 \quad -1] \cdot [14 \quad -48] = 28 + 48 = 76$: not descent

(c) $[-2 \quad -1]$

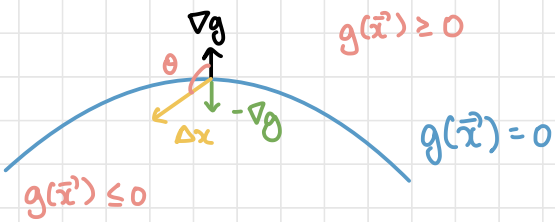
$[-2 \quad -1] \cdot [14 \quad -48] = -28 + 48 = 20$: not descent

(d) $[-2 \quad 1]$

$[-2 \quad 1] \cdot [14 \quad -48] = -28 - 48 = -76$: descent

2. Feasible direction

- Vector that will not violate the constraints is said to be moving in a feasible direction
- If the constraint is $g(\vec{x}) \leq 0$



- To test if a test vector $\vec{\Delta x}$ is moving in a feasible direction
  - ↳ Find $\nabla g(\vec{x})$
  - ↳ Find $\nabla g(\vec{x}) \cdot \vec{\Delta x}$
- If the dot product is ≤ 0, the vector is moving in feasible direction

Q: Given the constraint $x_1 + x_2^2 - 4 \leq 0$

(a) Choose any point on the boundary of this constraint

$(0, 2)$

(b) Test the vectors $(3,4)$, $(-5,4)$, $(-3,-2)$ for feasibility

$\nabla g = [1 \quad 2x_2]$ at $(0,2) = [1 \quad 4]$

1. $(3, 4) \cdot (1, 4) = 3+16 = 19$ — infeasible
2. $(-5, 4) \cdot (1, 4) = -5+16 = 11$ — infeasible
3. $(-3, -2) \cdot (1, 4) = -3-8 = -11$ — feasible

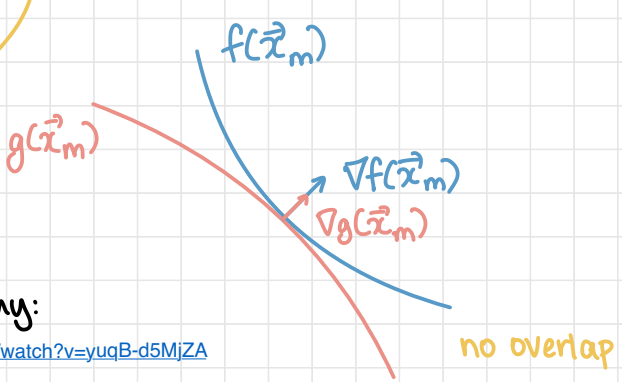
Descent & Feasible Cones

- Overlap between set of all descent vectors and set of all feasible vectors is called set of productive vectors
- Overlap gets smaller as minimum point is approached
- Once there is no overlap, optimal point reached

Non-Linear Optimisation

- First step is to start on the boundary
- Let the optimal point be represented by vector $\vec{x}_m$ $(\vec{x}^*)$
- $\nabla f(\vec{x}_m) = -\lambda \nabla g(\vec{x}_m)$ —— assume $g(\vec{x})$ is an equality constraint

λ = Lagrange Multiplier



Khan Academy:
https://www.youtube.com/watch?v=yuqB-d5MjZA

Lagrange Multiplier

- Define single Lagrangian function $L(x, \lambda) = f(x) + \lambda g(x)$
- To find optimal point / stationary point x_m (also x^*) of L wrt both x and λ, we need to solve the constrained optimization problem
- If x is d-dimensional,

$$\nabla_x L = 0 \longrightarrow d \text{ equations}$$

$$\frac{\partial L}{\partial \lambda} = 0 \longrightarrow 1 \text{ equation}$$

- Totally: $d+1$ equations, $d+1$ unknowns

Q: Maximize $f(x) = 1 - x_1^2 - x_2^2$ subject to $g(x) = x_1 + x_2 - 1 = 0$

$$L(x, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1) = 0$$

$$\nabla_x L = \begin{pmatrix} -2x_1 + \lambda \\ -2x_2 + \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 - 1 = 0$$

Solving (using calculator)

$$x_1 = \frac{1}{2} \quad x_2 = \frac{1}{2} \quad \lambda = 1$$

Optimal point x_m or $x^* = (1/2, 1/2)$

Multiple Constraints

- One multiplier per constraint

Karush-Kuhn-Tucker (KKT) Conditions

- Inequality: same d+1 equations plus additional constraints

(1) $g(x) \geq 0$

(2) $\lambda \geq 0$

(3) $\lambda g(x) = 0$

Support Vector Machines

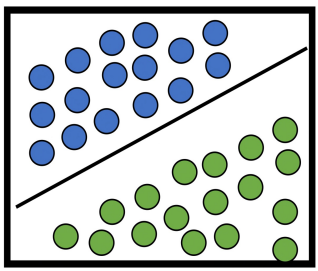
- KNN: classified based on neighbours
- SVM: divide space based on training examples and find where query point lies

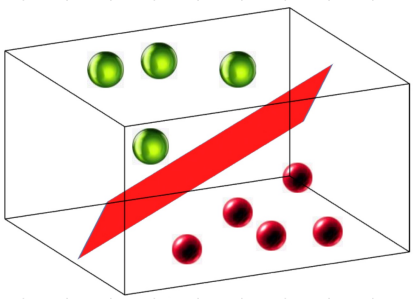
Properties

1. Non linearity can be handled
2. Multi classification
3. Classification & regression
4. Computationally expensive

Linearly separatable Data

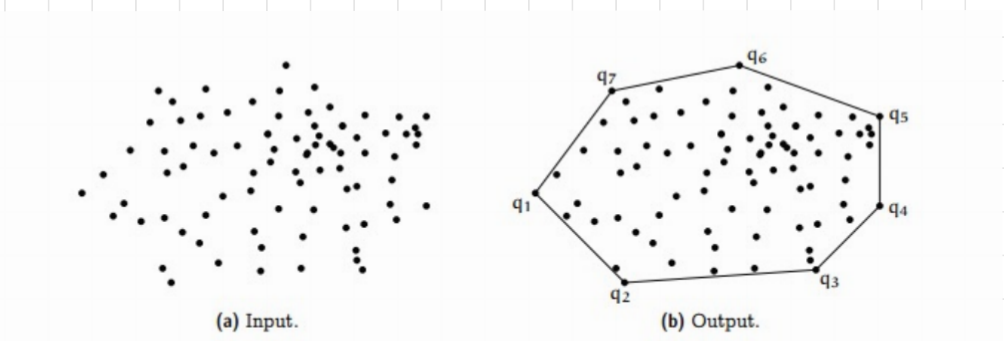
- Hyperplane can separate data





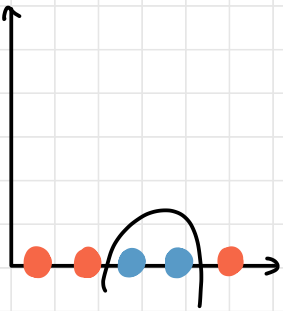
- Convex hull problem: CLRS

The convex hull of a set of points is defined as the smallest convex polygon that encloses all of the points in the set. Convex means that the polygon has no corner that is bent inwards.

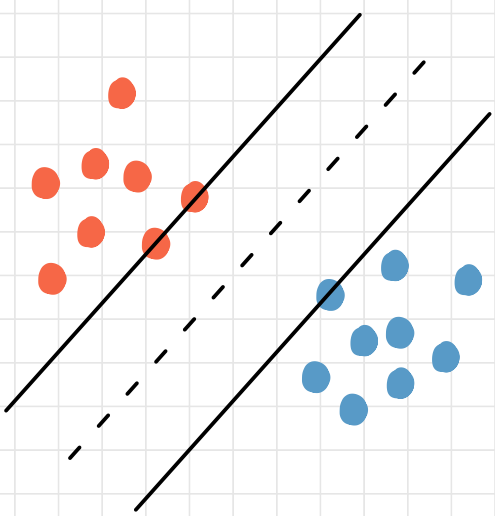


https://medium.com/@harshitsikchi/convex-hulls-explained-baab662c4e94

Transforming Data to Linear Separatability



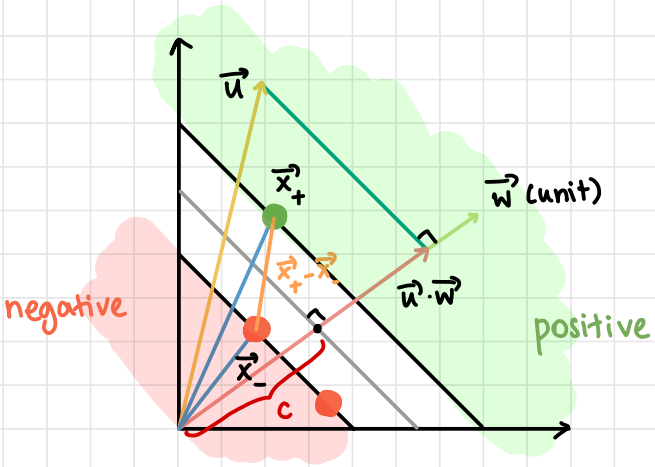
- Kernel trick: transform $x \rightarrow f(x) = x^2$
- Vapnik's idea: find widest street/channel/gutter to separate pos & neg classes



- Hyper plane with widest margin
- Watch MIT SVM: https://www.youtube.com/watch?v=_PwhiWxHK8o

Binary Classification

- Assume decision boundary & gutter found
- Let sample to be classified be represented as $\vec{u}$ (unknown) and a unit vector $\perp$ to the gutter be $\vec{w}$
- Rule for classification:
  - ↳ if projection of $\vec{u}$ onto $\vec{w}$ falls on positive side of street, $\vec{u}$ is positive
  - ↳ if projection of $\vec{u}$ onto $\vec{w}$ falls on negative side of street, $\vec{u}$ is negative



- In other words,

$$\vec{w} \cdot \vec{u} \geq c \quad \text{——} \quad \text{pos}$$
$$\vec{w} \cdot \vec{u} \leq c \quad \text{——} \quad \text{neg}$$

$$\vec{w} \cdot \vec{u} - c \geq 0 \longrightarrow (1)$$
$$\vec{w} \cdot \vec{u} - c \leq 0 \longrightarrow (2)$$

Let $\vec{u} = \vec{x}_+$ for +ve sample and $\vec{x}_-$ for negative sample

At the boundary, let the two constants be c_1 & c_2

$$\vec{w} \cdot \vec{x}_+ - c_1 = 0 \rightarrow (3) \qquad \vec{w}\vec{x}_- - c_2 = 0 \rightarrow (4)$$

Let $A = -\frac{(c_2 + c_1)}{2}$ and $B = -\frac{(c_2 - c_1)}{2}$

Then $-c_1 = A - B$ and $-c_2 = A + B$

Replacing in (3) & (4)

$$\vec{w}\vec{x}_+ + A - B = 0 \rightarrow (5) \qquad \vec{w}\vec{x}_- + A + B = 0 \rightarrow (6)$$

Dividing (5) & (6) by B

$$\frac{\vec{w}}{B}\vec{x}_+ + \frac{A}{B} - 1 = 0 \rightarrow (7) \qquad \frac{\vec{w}}{B}\vec{x}_- + \frac{A}{B} + 1 = 0 \rightarrow (8)$$

no longer unit

Redefining $\frac{\vec{w}}{B} = \vec{w}$ and $\frac{A}{B} = b$ in (7) & (8)

$$\vec{w}\vec{x}_+ + b = 1 \longrightarrow (9)$$

$$\vec{w}\vec{x}_- + b = -1 \longrightarrow (10)$$

Converting to inequality

$$\vec{w}\vec{x}_+ + b \geq 1 \longrightarrow (11)$$

$$\vec{w}\vec{x}_- + b \leq -1 \longrightarrow (12)$$

The predicted values of y

$$y_i = 1 \quad : +ve$$
$$y_i = -1 \quad : -ve$$

Multiplying (11) & (12) with y_i

$$y_i(\vec{w}\cdot\vec{x}_+ + b) - 1 \geq 0 \longrightarrow (13)$$
$$y_i(\vec{w}\cdot\vec{x}_- + b) - 1 \geq 0 \longrightarrow (14)$$

Generalizing for all $\vec{x}$s

$$y_i(\vec{w}\cdot\vec{x} + b) - 1 \geq 0 \longrightarrow (15)$$

Solving for $\vec{x}_+$ & $-\vec{x}_-$ from (13) & (14)

$$\vec{x}_+ \geq (\vec{w})^{-1}(1-b) \longrightarrow (15)$$

$$-\vec{x}_- \geq (\vec{w})^{-1}(1+b) \longrightarrow (16)$$

The width of the gutter is the projection of $\vec{x}_+ - \vec{x}_-$ onto $\vec{w}$, if $\vec{x}_+$ and $\vec{x}_-$ lie on the boundaries

$$\vec{x}_+ - \vec{x}_- = (15) + (16) = (\vec{w})^{-1}(2) \longrightarrow (17)$$

Width from (17)

$$\text{width} = \frac{2(\vec{w})^{-1}\,\vec{w}}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

The goal is to maximize width, or minimize $\|\vec{w}\|$

Primal Problem

maximize $f(w,b) = \frac{2}{\|w\|}$

subject to constraints

$$y_i(w^T x_i + b) - 1 \geq 0 \quad \text{(eq 15)}$$

for $i = 1$ to m where m = training instances

Optimization

maximize $\frac{2}{\|w\|}$ = minimize $\frac{\|w\|}{2}$ = minimize $\frac{\|w\|^2}{2}$

Using Lagrangian Multiplier Method

Multiple constraints

$$L(x,\lambda) = f(x) + \lambda_1 g_1(x) + \lambda_2 g_2(x) + \ldots + \lambda_k g_k(x)$$

Objective $f(w) = \frac{1}{2}\|w\|^2$ to minimize

Subject to m constraints $g(w,b) = y_i(w^T x_i + b) - 1$

$i = 1$ to m

Lagrangian Function

Lagrangian

$$L = f(w) - \sum_{i=1}^{m} \alpha_i g_i(w,b) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(w^T x_i + b) - 1 \right] \rightarrow (1)$$

Derivative wrt w

Why vector? check pg 48 of L4 - SVM in Prof Preeti's slides

$$\frac{\partial L}{\partial w} = \vec{w} - \sum \alpha_i y_i \vec{x_i} = 0$$

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \quad \longrightarrow (2)$$

Derivative wrt b

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \longrightarrow (3)$$

Replacing (2) & (3) in (1)

$$L = \frac{1}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j\right) - \left(\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j\right) - b\underbrace{\sum_{i=1}^{m} \alpha_i y_i}_{0} + \sum_{i=1}^{m} \alpha_i$$

$$L(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j\right)$$

SVM - Dual Optimization Problem

watch: https://www.youtube.com/watch?v=6-ntMIaJpm0

$$\max_{\alpha} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \right)$$

ST constraints $(\forall\, i = 1 \text{ to } m)$

$$c_1: \quad \alpha_i \geq 0$$

$$c_2: \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$c_3: \quad \sum_{i=1}^{m} \alpha_i y_i x_i x + b = 1$$

$$c_4: \quad \sum_{i=1}^{m} \alpha_i y_i x_i x + b = -1$$

- vectors that lie on boundary: support vectors $(\alpha > 0)$

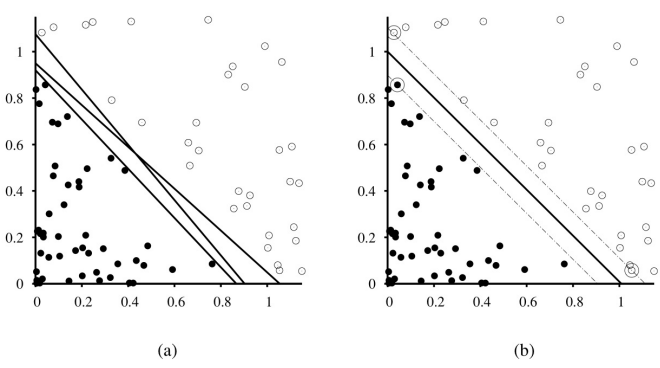


Figure 18.30 Support vector machine classification: (a) Two classes of points (black and white circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large circles) are the examples closest to the separator.

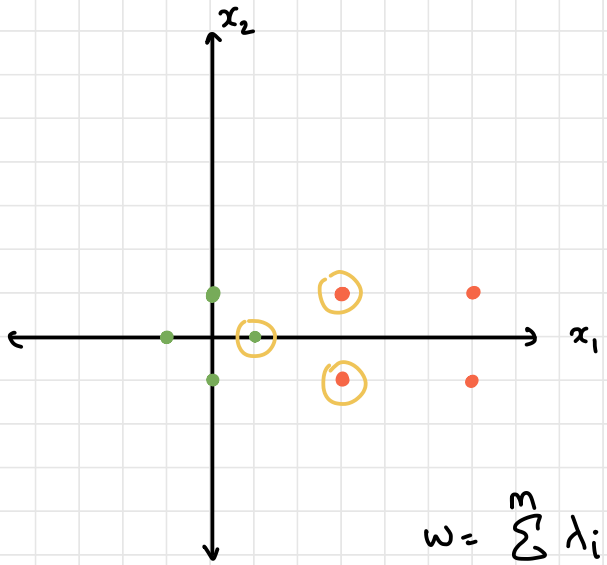
Q: (3,1), (3,-1), (6,1), (6,-1) | (1,0), (0,1), (0,-1), (-1,0)

+ve | -ve

$s_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ -ve

$s_2 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$ +ve

$s_3 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ +ve



$$w = \sum_{i=1}^{m} \lambda_i y_i x_j$$

3 Lagrangian multipliers

$$c_1 : \alpha_i \geq 0$$

$$c_2 : \sum_{i=1}^{3} \lambda_i y_i = 0 = -\lambda_1 + \lambda_2 + \lambda_3 = 0 \longrightarrow (1)$$

$$c_3 : \sum_{i=1}^{3} \lambda_i y_i \vec{x_i} \cdot x_1 + b = -1$$

$$c_4 : \sum_{i=1}^{3} \lambda_i y_i \vec{x_i} \cdot x_2 + b = +1$$

$$c_5 : \sum_{i=1}^{3} \lambda_i y_i \vec{x_i} \cdot x_3 + b = +1$$

$C_3: \lambda_1(-1)\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \lambda_2(1)\begin{bmatrix}3 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \lambda_3(1)\begin{bmatrix}3 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + b = -1$

$-\lambda_1 + 3\lambda_2 + 3\lambda_3 + b = -1 \longrightarrow (2)$

$C_4: \lambda_1(-1)\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}3\\1\end{bmatrix} + \lambda_2(1)\begin{bmatrix}3 & 1\end{bmatrix}\begin{bmatrix}3\\1\end{bmatrix} + \lambda_3(1)\begin{bmatrix}3 & -1\end{bmatrix}\begin{bmatrix}3\\1\end{bmatrix} + b = +1$

$-3\lambda_1 + 10\lambda_2 + 8\lambda_3 + b = 1 \longrightarrow (3)$

$C_5: \lambda_1(-1)\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}3\\-1\end{bmatrix} + \lambda_2(1)\begin{bmatrix}3 & 1\end{bmatrix}\begin{bmatrix}3\\-1\end{bmatrix} + \lambda_3(1)\begin{bmatrix}3 & -1\end{bmatrix}\begin{bmatrix}3\\-1\end{bmatrix} + b = +1$

$-3\lambda_1 + 8\lambda_2 + 10\lambda_3 + b = 1 \longrightarrow (4)$

$\lambda_1 = 1/2 \quad \lambda_2 = 1/4 \quad \lambda_3 = 1/4 \quad b = -2$

$$w = \sum_{i=1}^{3} \lambda_i x_i y_i = \frac{1}{2} \times \begin{bmatrix}1\\0\end{bmatrix}(-1) + \frac{1}{4}\begin{bmatrix}3\\1\end{bmatrix}(1) + \frac{1}{4}\begin{bmatrix}3\\-1\end{bmatrix}(1)$$

$$= \begin{bmatrix}-1/2\\0\end{bmatrix} + \begin{bmatrix}3/4\\1/4\end{bmatrix} + \begin{bmatrix}3/4\\-1/4\end{bmatrix}$$

$$= \begin{bmatrix}1\\0\end{bmatrix} \longrightarrow wx + b = 0 \text{ is line}$$

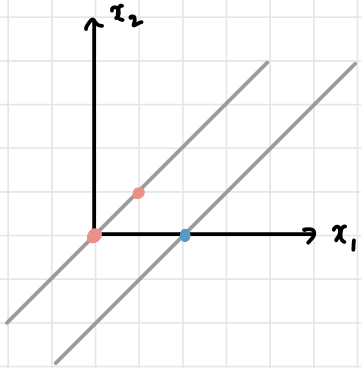
$(1)x_1 + 0(x_2) - 2 = 0 \Rightarrow \boxed{x_1 = 2}$

Q: The samples are (0,0), (1,1), (2,0) (SVs)
-ve -ve +ve

Test sample: (6,0)

How many Lagrangian Multipliers? 3 (1 for each support vect)

Find weight & bias.



$$c_1 = \sum \lambda_i y_i = 0$$

$$-\lambda_1 - \lambda_2 + \lambda_3 = 0 \longrightarrow (1)$$

$$c_2 = wx + b = +1 \quad \text{pos}$$
$$wx + b = -1 \quad \text{neg}$$

$$w = \sum_{i=1}^{m} y_i x_i \lambda_i \quad x_j + b = -1$$

$$-\lambda_1 \begin{pmatrix} 0 \\ 0 \end{pmatrix} [0 \;\; 0] - \lambda_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} [0 \;\; 0] + \lambda_3 \begin{pmatrix} 2 \\ 0 \end{pmatrix} [0 \;\; 0] + b = -1$$

$$b = -1 \longrightarrow (2)$$

$$-\lambda_1 \begin{pmatrix} 0 \\ 0 \end{pmatrix} [1 \quad 1] - \lambda_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} [1 \quad 1] + \lambda_3 \begin{pmatrix} 2 \\ 0 \end{pmatrix} [1 \quad 1] + b = -1$$

$$-2\lambda_2 + 2\lambda_3 + b = -1 \longrightarrow (3)$$

$$w = \sum_{i=1}^{m} y_i x_i \lambda_i \; x_j + b = +1$$

$$-\lambda_1 \begin{pmatrix} 0 \\ 0 \end{pmatrix} [2 \quad 0] - \lambda_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} [2 \quad 0] + \lambda_3 \begin{pmatrix} 2 \\ 0 \end{pmatrix} [2 \quad 0] + b = 1$$

$$-2\lambda_2 + 4\lambda_3 + b = 1 \longrightarrow (4)$$

Solving ①, ②, ③ & ④

$$\lambda_3 = \lambda_1 + \lambda_2 \qquad b = -1$$

③:

$$-2\lambda_2 + 2\lambda_1 + 2\lambda_2 - 1 = -1$$

$$2\lambda_1 = 0 \Rightarrow \lambda_1 = 0$$

④.

$$-2\lambda_2 + 4\lambda_2 - 1 = 1$$

$$2\lambda_2 = 2$$

$$\lambda_2 = 1$$

$$\lambda_1 = 0 \quad \lambda_2 = 1 \quad \lambda_3 = 1 \quad b = -1$$

$$w = \sum x_i y_i \lambda_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix}(-1)(0) + \begin{pmatrix} 1 \\ 1 \end{pmatrix}(-1)(1) + \begin{pmatrix} 2 \\ 0 \end{pmatrix}(1)(1)$$

$$= \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

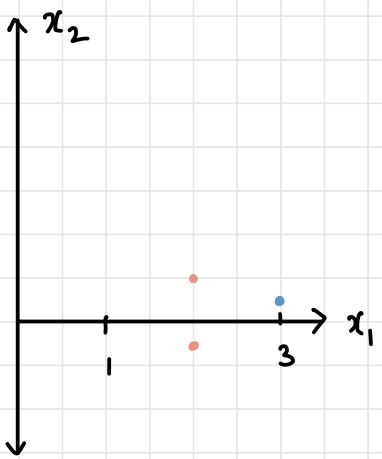
$w = [1 \quad -1] \longrightarrow \perp$ to line

$x_1 - x_2 - 1 = 0 \Rightarrow \boxed{x_2 = x_1 - 1}$

If $wx_i + b > 1$, +ve sample, not SV

Q: Samples: $(3, 1/4)$, $(2, -1/4)$, $(2, 1/2)$. Find out w & b.

-ve +ve +ve



$\begin{pmatrix}3\\1/4\end{pmatrix}$ $\begin{pmatrix}2\\-1/4\end{pmatrix}$ $\begin{pmatrix}2\\1/2\end{pmatrix}$

$-$ $+$ $+$

C_1 $\quad \sum \lambda_i y_i = 0$

$$-\lambda_1 + \lambda_2 + \lambda_3 = 0 \longrightarrow (1)$$

C_2 $\quad wx + b = +1 \quad (+ve)$

$wx + b = -1 \quad (-ve)$

$$w = \sum_{i=1}^{m} \lambda_i x_i y_i$$

x_1:

$$\left(\lambda_1 \begin{pmatrix}3\\1/4\end{pmatrix}(-1) + \lambda_2 \begin{pmatrix}2\\-1/4\end{pmatrix}(1) + \lambda_3 \begin{pmatrix}2\\1/2\end{pmatrix}(1)\right)\begin{bmatrix}3 & 1/4\end{bmatrix} + b = -1$$

$$\frac{-145}{16}\lambda_1 + \frac{95}{16}\lambda_2 + \frac{49}{8}\lambda_3 + b = -1 \longrightarrow (2)$$

x_2:

$$\left(\lambda_1 \begin{pmatrix}3\\1/4\end{pmatrix}(-1) + \lambda_2 \begin{pmatrix}2\\-1/4\end{pmatrix}(1) + \lambda_3 \begin{pmatrix}2\\1/2\end{pmatrix}(1)\right)\begin{bmatrix}2 & -1/4\end{bmatrix} + b = +1$$

$$\frac{-95}{16}\lambda_1 + \frac{65}{16}\lambda_2 + \frac{31}{8}\lambda_3 + b = 1 \longrightarrow (3)$$

x_3:

$$\left(\lambda_1 \begin{pmatrix}3\\1/4\end{pmatrix}(-1) + \lambda_2 \begin{pmatrix}2\\-1/4\end{pmatrix}(1) + \lambda_3 \begin{pmatrix}2\\1/2\end{pmatrix}(1)\right)\begin{bmatrix}2 & 1/2\end{bmatrix} + b = +1$$

$$\frac{-49}{8}\lambda_1 + \frac{31}{8}\lambda_2 + \frac{17}{4}\lambda_3 + b = 1 \longrightarrow (4)$$

$$\lambda_1 = 2 \quad \lambda_2 = \frac{2}{3} \quad \lambda_3 = \frac{4}{3} \quad b = 5$$

$$w = \sum_{i=1}^{m} \lambda_i x_i y_i = 2\begin{pmatrix} 3 \\ 1/4 \end{pmatrix}(-1) + \frac{2}{3}\begin{pmatrix} 2 \\ -1/4 \end{pmatrix}(1) + \frac{4}{3}\begin{pmatrix} 2 \\ 1/2 \end{pmatrix}$$

$$w = \begin{pmatrix} -2 \\ 0 \end{pmatrix} \perp \text{to line}$$

$$b = 5$$

$$-2x_1 + 5 = 0$$

$$x_1 = \frac{5}{2}$$

NON-LINEARLY SEPARABLE DATA

- Transform data to be linearly separable (kernel)
  - can do so by mapping points to higher dim space
- Let $x = (x_1, x_2, x_3)^T$, $y = (y_1, y_2, y_3)^T$
- Assume we need to map to 9D space with function ϕ

$$\phi(x) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$

$$\phi(y) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

- Final result required is dot product of 2 support vectors

- $\phi(x)^T \phi(y) = \sum_{i=1}^{3} \sum_{j=1}^{3} x_i x_j y_i y_j$

- Computational complexity: $O(n^2)$ — here $n = 3$

- Can use a kernel trick to avoid unnecessary computations

- Using kernel function $k(x,y) = (x^T y)^2$

$$k(x,y) = (x_1 y_1 + x_2 y_2 + x_3 y_3)^2$$

$$k(x,y) = \sum_{i=1}^{3} \sum_{j=1}^{3} x_i x_j y_i y_j$$

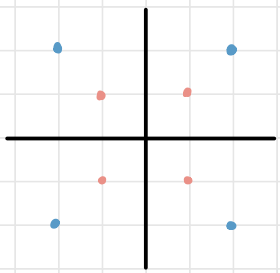
- Rewrite dual problem as

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Kernel Functions

1. Polynomial : $k(x_1, x_2) = (x_1 \cdot x_2 + c)^d$
2. Gaussian / RBF

Q: $\underbrace{\begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\-2\end{pmatrix}, \begin{pmatrix}-2\\2\end{pmatrix}, \begin{pmatrix}-2\\-2\end{pmatrix}}_{+ve}, \underbrace{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}}_{-ve}$

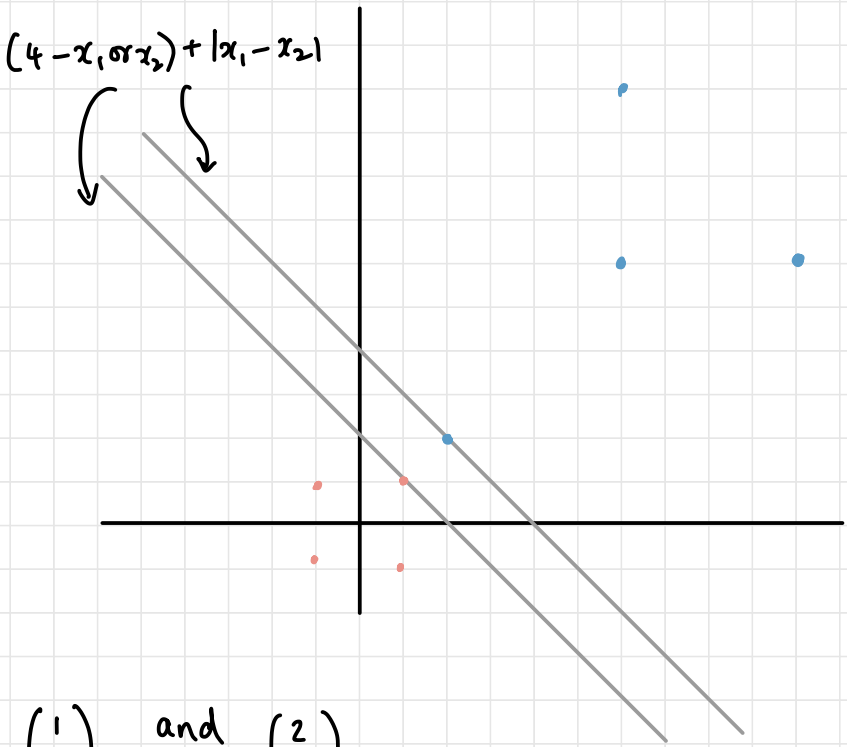


(kernel)

$$\phi\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{cases} \begin{pmatrix}(4-x_2)+|x_1-x_2|\\(4-x_1)+|x_1-x_2|\end{pmatrix} & \text{if } \sqrt{x_1^2+x_2^2} \geq 2 \\ \begin{pmatrix}x_1\\x_2\end{pmatrix} & \text{otherwise} \end{cases}$$

New points:

$$\underbrace{\begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}10\\6\end{pmatrix}, \begin{pmatrix}6\\10\end{pmatrix}, \begin{pmatrix}6\\6\end{pmatrix}}_{+ve}, \underbrace{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}}_{-ve}$$



SVs: $\begin{pmatrix}1\\1\end{pmatrix}$ and $\begin{pmatrix}2\\2\end{pmatrix}$

$\begin{pmatrix}1\\1\end{pmatrix}$ -ve $\quad$ $\begin{pmatrix}2\\2\end{pmatrix}$ +ve

$$c_1 \cdot \sum_{i=1}^{2} \lambda_i y_i = 0$$

$$c_2 : \sum_{i=1}^{2} \lambda_i y_i \vec{x_i} \cdot x_1 + b = -1$$

$$c_3 \sum_{i=1}^{2} \lambda_i y_i \vec{x_i} \cdot x_2 + b = +1$$

$$c_1: \quad -\lambda_1 + \lambda_2 = 0 \longrightarrow (1)$$

$$c_2: \quad \left(\lambda_1(-1)\begin{pmatrix}1\\1\end{pmatrix} + \lambda_2(1)\begin{pmatrix}2\\2\end{pmatrix}\right)\cdot\begin{pmatrix}1\\1\end{pmatrix} + b = -1$$

$$-2\lambda_1 + 4\lambda_2 + b = -1 \longrightarrow (2)$$

$$c_3: \quad \left(\lambda_1(-1)\begin{pmatrix}1\\1\end{pmatrix} + \lambda_2(1)\begin{pmatrix}2\\2\end{pmatrix}\right)\cdot\begin{pmatrix}2\\2\end{pmatrix} + b = 1$$

$$-4\lambda_1 + 8\lambda_2 + b = 1 \longrightarrow (3)$$

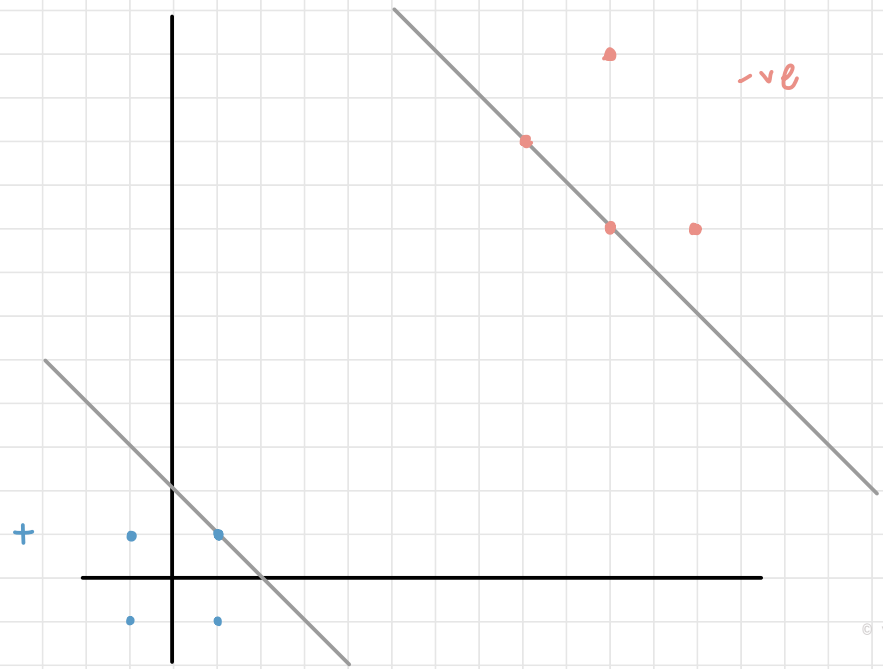
Q:

$$\phi(x_1, x_2) = \begin{cases} \begin{pmatrix} 6 - x_1 + (x_1 - x_2)^2 \\ 6 - x_2 + (x_1 - x_2)^2 \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} \geq 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \end{cases}$$

$\underbrace{(1,1), (-1,1), (-1,-1), (1,-1)}_{+ve}$, $\underbrace{(2,0), (0,2), (-2,0), (0,-2)}_{-ve}$

Transformed points

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 8 \\ 10 \end{pmatrix}, \begin{pmatrix} 10 \\ 8 \end{pmatrix}, \begin{pmatrix} 12 \\ 10 \end{pmatrix}, \begin{pmatrix} 10 \\ 12 \end{pmatrix}$$



Q: Construct an SVM that computes XOR function. Use values +1 and -1 for both input & output. Map the input (x_1, x_2) into a space consisting of $(x_1, x_1 x_2)$.

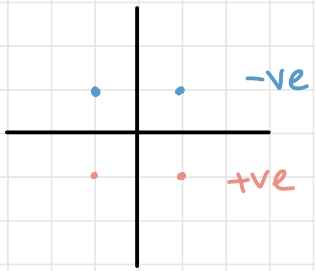
Draw the four input points in this space and the max margin seperator.

What is the margin? Infer if the datapoints are separable or not.

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

New plane

x_1	x_2	y
-1	1	-1
-1	-1	1
1	-1	1
1	1	-1



$$\sum_{i=1}^{m} \lambda_i w_i = 0$$