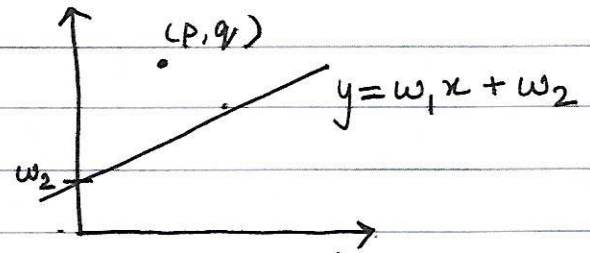
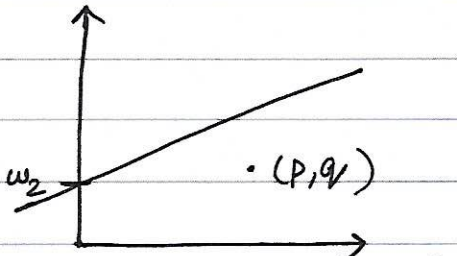
LINEAR REGRESSION

ABSOLUTE TRICK



(point above line)



(point below line)

To bring line closer to the point; when point above line.

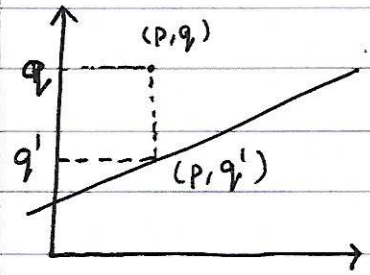
$$y = (w_1 + p\alpha)x + (w_2 + \alpha)$$ where,

α = learning rate

when point below line,

$$y = (w_1 - p\alpha)x + (w_2 - \alpha)$$

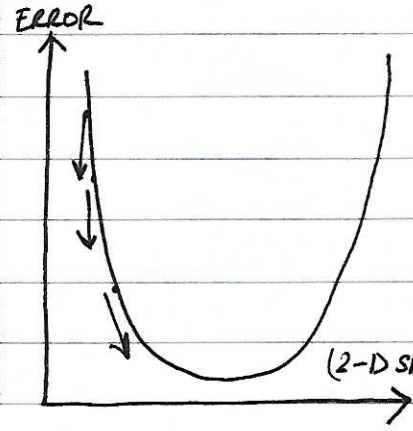
SQUARE TRICK



$$y = (w_1 + p(q-q')\alpha)x + (w_2 + (q-q')\alpha)$$

The equation above also works when point is below the line. $(q-q')$ becomes negative and both slope & y-intercept of line are decreased.

GRADIENT DESCENT

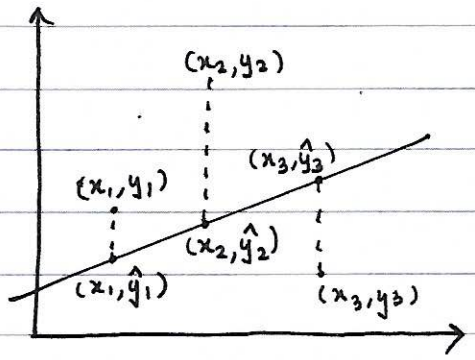


(2-D SIMPLIFICATION) (IN REALITY THIS WOULDN'T BE 2-D)

Move in the direction of negative gradient of error function, until the minima is reached.

$$w_i := w_i - \alpha \frac{\partial (\text{Error})}{\partial w_i}$$

MEAN ABSOLUTE ERROR



$$\text{ERROR} = \frac{1}{m} \sum_{i=1}^{m} |y_i - \hat{y}_i| \text{, where,}$$

m : no. of. training examples (or) points

MEAN SQUARED ERROR

$$\text{ERROR} = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

$$\frac{\partial}{\partial w_1} \text{ERROR} = -(y - \hat{y})x$$

$$\frac{\partial}{\partial w_2} \text{ERROR} = -(y - \hat{y})$$

$$\frac{\partial}{\partial w_1} \cdot \frac{1}{2m} \Sigma (y_i - \hat{y}_i)^2$$

$$= \frac{2}{2m} \Sigma (y_i - \hat{y}_i) \cdot \frac{\partial}{\partial w_1} (y_i - \hat{y}_i)$$

$$= \frac{1}{m} \Sigma (y_i - \hat{y}_i) \cdot \frac{\partial}{\partial w_1} y_i - (w_1 x_i + w_2)$$

$$= \frac{1}{m} \Sigma (y_i - \hat{y}_i) \cdot (-x_i)$$

Gradient Descent:

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} \text{ERROR}$$

$$= w_1 - \alpha (-(y - \hat{y})x)$$

$$= w_1 + \alpha (y - \hat{y})x$$

$$w_1 = w_1 + \alpha \cdot (q - q') \cdot p$$

$$w_2 = w_2 - \alpha \frac{\partial}{\partial w_2} \text{ERROR}$$

$$= w_2 - \alpha (-(y - \hat{y}))$$

$$= w_2 + \alpha (y - \hat{y})$$

$$w_2 = w_2 + \alpha (q - q')$$

The above is same as the square trick to fit a line through a set of points

Similarly, gradient descent with MAE as the error function is equivalent to 'absolute trick' method.

UNIVERSITY OF OXFORD

HIGHER DIMENSIONS

	Size	School Quality	...	No. of rooms	Price
House 1	900	6	...	2	$100K
House 2	560	4	...	1	$50K
...	...	...	...	...	
House m	2000	7	...	4	$250K
	(x_1)	(x_2)	...	(x_{n-1})	$(\hat{y})$

$$\hat{y} = w_1 x_1 + w_2 x_2 + \dots + w_{n-1} x_{n-1} + w_n$$

MULTIPLE LINEAR REGRESSION

For 'n' predictor variables (or) features:

$$y = m_1 x_1 + m_2 x_2 + \dots + m_n x_n + b$$

CLOSED FORM SOLUTION TO FIND MODEL PARAMETERS (w_i)

$$E(w_1, w_2) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x_i + w_2 - y_i)^2$$

To minimise error function, we take derivatives and set them to 0

$$\frac{\partial E}{\partial w_1} = \sum_{i=1}^{m} (w_1 x_i + w_2 - y_i) x_i$$

$$= w_1 \sum x_i^2 + w_2 \sum x_i - \sum x_i y_i$$

$$\frac{\partial E}{\partial w_2} = \sum (w_1 x_i + w_2 - y_i)$$

$$= w_1 \sum x_i + w_2 \sum 1 - \sum y_i$$

Setting to 0

$$w_1 \sum x_i^2 + w_2 \sum x_i = \sum x_i y_i$$

$$w_1 \sum x_i + w_2 m = \sum y_i$$

2 equations, 2 unknown variables can be solved using any closed form mathematical method. (2 linear equations, 2 variables)

$$w_1 a_1 + w_2 b_2 = c_1 \quad ; \quad w_1 a_2 + w_2 b_2 = c_2$$

UNIVERSITY OF OXFORD

CLOSED FORM SOLUTION – n DIMENSIONS / FEATURES

$$X = \begin{bmatrix} x_0^{(0)} & x_1^{(0)} & \dots & x_n^{(0)} \\ x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$

(rowwise training examples)
m : training examples
n : dimensions / features

$$y \ (\hat{y}) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$E(W) = \frac{1}{2m} (XW - y)^T (XW - y)$$

$$\frac{\partial E}{\partial W} = 2X^T (XW - y)$$

$$W = (X^T X)^{-1} X^T y \qquad \text{// Closed form, aka Normal Equation}$$

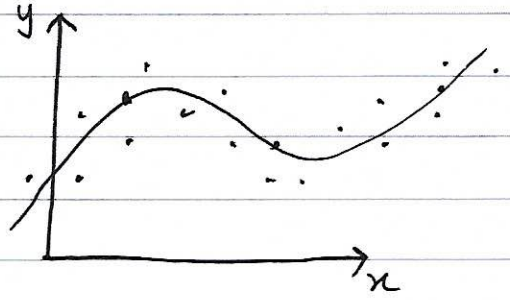
Closed form solution to find model params / weights would be expensive, if n is large, because finding inverse of matrix $X^T X$ is computationally expensive for high dimension matrices.

LINEAR REGRESSION WARNINGS

* LR works best when the data is linear. If not, you'll need to make adjustments (transform your training data), add features, or y use another kind of model.
* LR is sensitive to outliers.

POLYNOMIAL REGRESSION

Transforming features to improve the form/fit of hypothesis function



$\hat{y} = w_1 x^3 + w_2 x^2 + w_3 x + w_4$ (example)

REGULARISATION

Addresses problem of model 'overfitting'

Regularisation will remove features from a model (by setting their coeff. to 0) if the penalty for removing them is small. This makes model simpler & generalised

L1 Regularisation

$$-Error_{new} = Error_{\substack{without \\ Regularisation}} + \lambda \sum_{i=1}^{n} |w_i|$$

- Computationally inefficient (unless data is sparse)

L2 Regularisation

$$-Error_{new} = Error + \lambda \sum w_i^2$$

- Computationally efficient

FEATURE SCALING

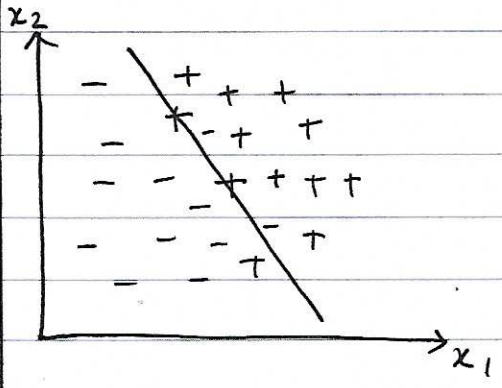
Transforming training data features into a common range of values.

$$x_j := \frac{x_j - \mu_j}{s_j} \quad ; df['h\_new'] = \frac{df['h'] - df['h'].mean()}{df['h'].std()}$$

- can speed-up convergence of gradient descent
- Should always be done with regularised methods

PERCEPTRON ALGORITHM

LINEAR BOUNDARIES



Boundary line:

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$WX + b = 0, \text{ where}$$

$W = (w_1, w_2)$ (weights)

$X = (x_1, x_2)$

b = bias

Prediction

$$\hat{y} = \begin{cases} 1 & \text{if } WX + b \geq 0 \\ 0 & \text{if } WX + b < 0 \end{cases}$$

HIGHER DIMENSIONS

Boundary plane:

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0$$

$$WX + b = 0$$

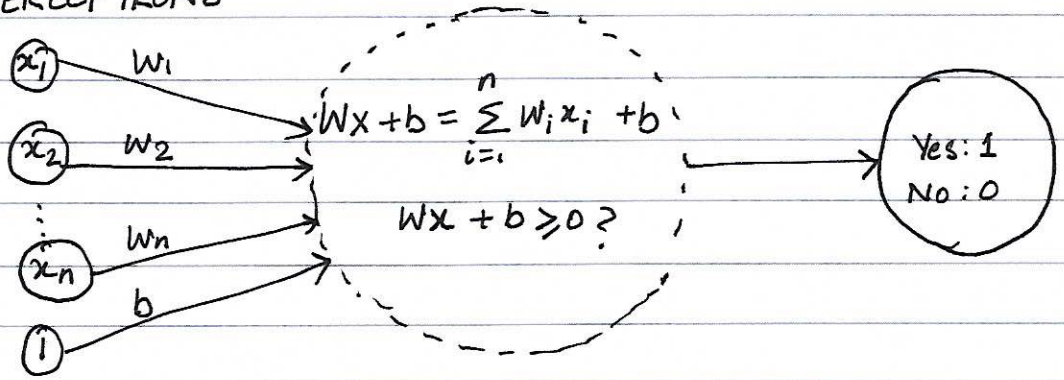
n-dimensional ~~plot~~ space

$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b = 0$$

$$Wx + b = 0$$

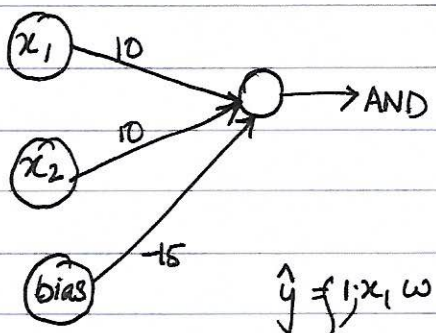
$$\hat{y} = \begin{cases} 1 & \text{if } Wx + b \geq 0 \\ 0 & \text{if } Wx + b < 0 \end{cases}$$

PERCEPTRONS



PERCEPTRONS AS LOGICAL OPERATORS

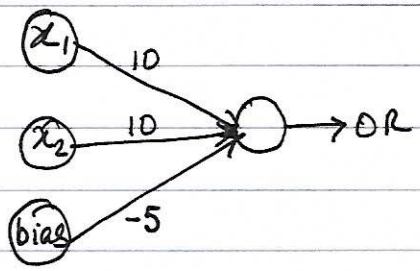
AND Perception



x_1	x_2	$\hat{y}$	AND
0	0	$10\times0+10\times0-15=-15$	0
0	1	$10\times0+10\times1-15=-5$	0
0	0	$10\times1+10\times0-15=-5$	0
1	1	$10\times1+10\times1-15=5$	1

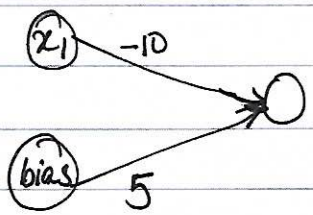
$$\hat{y} = \begin{cases} 1; x_1\omega_1 + x_2\omega_2 + b \geq 0 \\ 0; x_1\omega_1 + x_2\omega_2 + b < 0 \end{cases}$$

OR Perception



x_1	x_2	$\hat{y}$	OR
0	0	$10\times0+10\times0-5=-5$	0
0	1	$10\times1+10\times0-5=5$	1
1	0	$10\times0+10\times1-5=5$	1
1	1	$10\times1+10\times1-5=15$	1

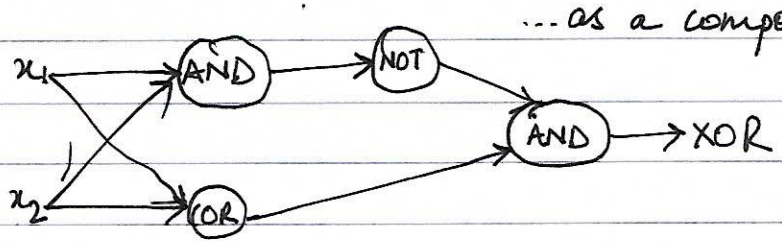
NOT Perception



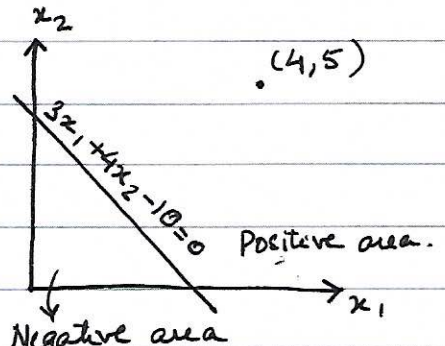
x_1	$\hat{y}$	NOT
0	$-10\times0+5=5$	1
1	$-10\times1+5=-5$	0

XOR Perception.

...as a composite perception



PERCEPTRON TRICK



Line : $3x_1 + 4x_2 - 10 = 0$

Point : (4,5)

Learning rate : 0.1

	3	4	-10
-	0.4	0.5	0.1
	2.6	3.5	-10.1

New line : $2.6x_1 + 3.5x_2 - 10.1 = 0$

(if point is below line, add params to coeff. instead of subtracting)

PERCEPTRON ALGORITHM

1. Start with random weights

$w_1, w_2, \ldots, w_n, b$

2. For every misclassified point $(x_1, x_2, \ldots, x_n)$:

   a. If prediction = 0, i.e. in negative area,

   For $i = 1..n$

   Change $w_i = w_i + \alpha x_i$

   $b = b + \alpha$

   b. If prediction in positive area or prediction = 1,

   For $i = 1..n$

   Change $w_i = w_i - \alpha x_i$

   $b = b + \alpha$

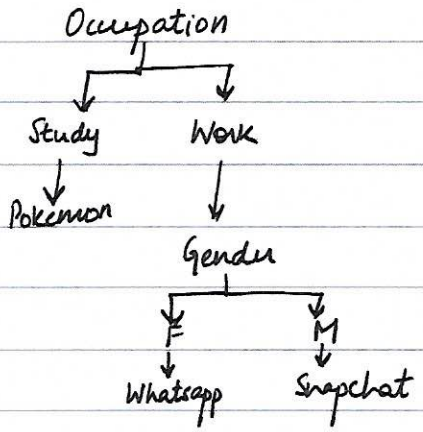
DECISION TREES

EXAMPLE: RECOMMENDING APPS (REGRESSION + CLASSIFICATION)

Training Data:

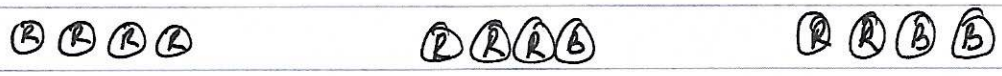
GENDER	OCCUPATION	APP
F	Study	Pokemon
F	Work	Whatsapp
M	Work	Snapchat
F	Work	Whatsapp
M	Study	Pokemon
M	Study	Pokemon

Decision Tree:



[Decision Trees tend to suffer from high variance or overfitting.]

ENTROPY



High knowledge. Low Entropy

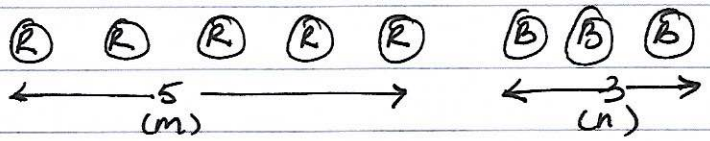
Medium knowledge Medium Entropy

Low Knowledge High Entropy

Probability of Winning.

	P(red)	P(blue)	P(winning)	$-\log_2 P(\text{win})$
R R R R	1	0	$1 \times 1 \times 1 \times 1 = 1$	$0+0+0+0 = 0$
R R R B	3/4	1/4	$3/4 \times 3/4 \times 3/4 \times 1/4 = 0.105$	$0.415+0.415+0.415+2 = 3.245$
R R B B	1/2	1/2	$1/2 \times 1/2 \times 1/2 \times 1/2 = 0.0625$	$1+1+1+1 = 4$

Average Entropy



$$\text{(Avg) Entropy} = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\left(\frac{3}{8}\right) = 0.9544$$

$$\text{General Form} = -\frac{m}{m+n}\log_2\left(\frac{m}{m+n}\right) - \frac{n}{m+n}\log_2\left(\frac{n}{m+n}\right)$$

Let $p_1 = \frac{m}{m+n}$; probability that answer is 'm'

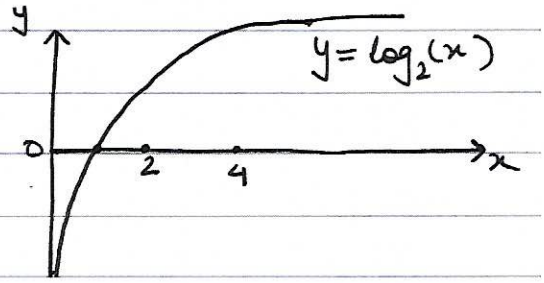
& $p_2 = \frac{n}{m+n}$.

Thus, Entropy $= -p_1 \log_2(p_1) - p_2 \log_2(p_2)$

MULTICLASS ENTROPY

$$-p_1 \log_2(p_1) - p_2 \log_2(p_2) - \ldots - p_n \log_2(p_n)$$

$$= -\sum_{i=1}^{n} p_i \log_2(p_i)$$



INFORMATION GAIN

Change in entropy

Information gain of a decision step = Entropy(parent) $- \frac{1}{2}$ [E(child1) + E(child2)]

Example: Recommending Apps

Gender	Occupation	App
F	Study	Pokemon
F	Work	Whatsapp
M	Work	Snapchat
F	Work	Whatsapp
M	Study	Pokemon
M	Study	Pokemon

P P P W W S

Entropy $= -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) = 1.46$

Split by gender:

	F	M
	P W W	S P P
Entropy	0.92	0.92

∴ Information gain, when split by gender,

$$IG = 1.46 - \frac{1}{2}(0.92 + 0.92) = 0.54$$

Similarly, when split by occupation,

$$IG = 1.46 - \frac{1}{2}(0 + 0.92) = 1$$

∴ It is best to split the original data set by occupation.

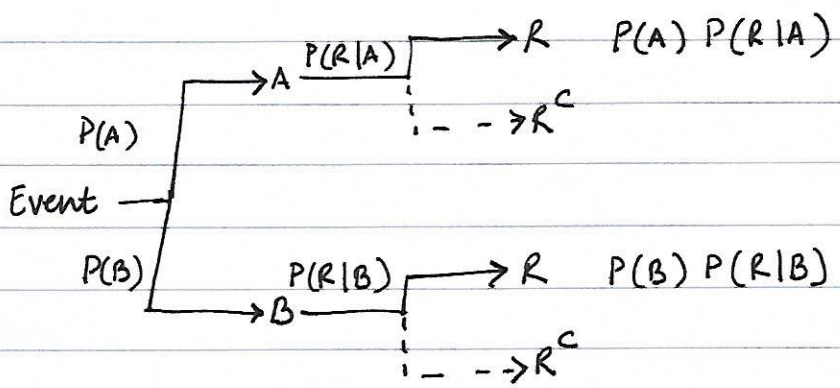
HYPERPARAMETERS

Aspects of the decision tree that are tuned to ensure that it generalises well to new problems.

- Maximum Depth. Large depth often causes overfitting. Too small = underfit.
- Minimum no. of ~~parent~~ samples to split
- Minimum no. of samples per leaf. Small = overfit; Large = underfit.

NAIVE BAYES

BAYES THEOREM



$$P(A|R) = \frac{P(A)\,P(R|A)}{P(A)\,P(R|A) + P(B)\,P(R|B)}$$

$$P(B|R) = \frac{P(B)\,P(R|B)}{P(A)\,P(R|A) + P(B)\,P(R|B)}$$

POSTERIOR

Alternatively,

Given,

$P(H)$ = Probability of hypothesis

$P(E|H)$ = Probability of evidence given hypothesis

$P(E)$ = Probability of evidence

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)} \quad \text{(OR)} \quad P(y|x_1,\ldots,x_n) = \frac{P(y)\,P(x_1,\ldots,x_n|y)}{P(x_1,\ldots,x_n)}$$

(For multiple features)

NAIVE ASSUMPTION

$$P(A \cap B) = P(A)\,P(B)$$

[In practice, this is true iff A & B are independent events]

CONDITIONAL PROBABILITY

$$P(A|B)\,P(B) = P(B|A)\,P(A) \Rightarrow P(A|B) \propto P(B|A)\,P(A)$$

UNIVERSITY OF OXFORD

NAIVE BAYES ALGORITHM

$$P(\text{spam} \mid \text{easy}, \text{money}) \propto P(\text{easy}, \text{money} \mid \text{spam})\, P(\text{spam})$$

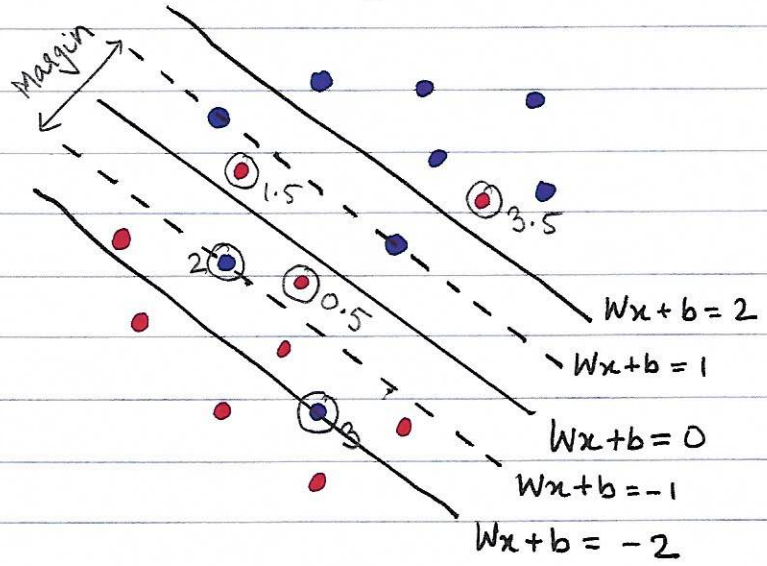
$$\propto P(\text{easy} \mid \text{spam})\, P(\text{money} \mid \text{spam})\, P(\text{spam})$$

(using naive assumption)

SUPPORT VECTOR MACHINES

ERROR FUNCTION

Error = Classification Error + Margin Error.



Classification error = (2+3) + (0.5 + 1.5+3.5)
i.e. distance of mis-classified points from margins.

Goal is to have as large a margin as possible.

Margin error

- Margin = $\frac{2}{|W|}$ & where $|W| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$
- Error = $|W|^2$ (because, we want a large margin to give a small error & vice-versa)
- Formula of margin comes from formula for distance between two parallel lines

$$ax + by + c_1 = 0$$
$$ax + by + c_2 = 0$$
$$\text{distance} = \left| \frac{c_2 - c_1}{\sqrt{a^2 + b^2}} \right|$$

THE C PARAMETER

A hyper parameter used to create a model.

Error = C . Classification Error + Margin Error

Small C

⇒ Emphasis on margin error ⇒ Large Margin
May make classification Errors

Large C

⇒ Emphasis on classification error
⇒ Classifies points well ; May have small margin

POLYNOMIAL KERNEL

When a set of points can't be seperated linearly, project data points from 2-Dimensions to say, 5-Dimensions.

2-D (Project to higher dimensions) 5-D

(x, y) ⟶ (x, y, x^2, xy, y^2)

Ex: (2, 3) (2, 3, 4, 6, 9)

Now, when data lives in higher dimensional space, we may get lucky and a 4-D hyperplane* may act as a good classification boundary.

(Find Hyperplane)

* Hyperplane : a subspace whose dimensions is one less than that of its ambient space.

If such a hyperplane is found, we project it back to 2-D, obtaining a degree-2 polynomial boundary.

(Project back into higher degree polynomial).

A kernel is simply a set of functions that may act as a classification / decision boundary.

Example :

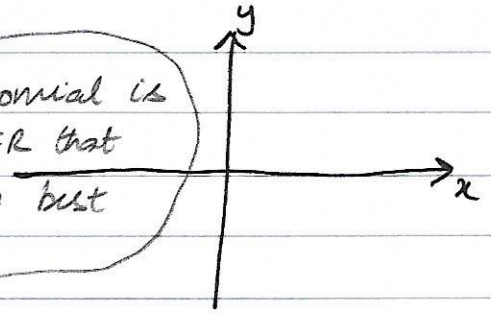
(1) Linear Kernel (x, y)

Candidate functions:

$2x - y = 1$

$5x - 2y = 3$, etc

Degree of polynomial is a HYPERPARAMETER that is tuned to obtain best model.



(2) Polynomial kernel. (x, y, x^2, xy, y^2) (degree 2)

$x^2 + y^2 = 1$

$xy = 1$

$y = x^2$

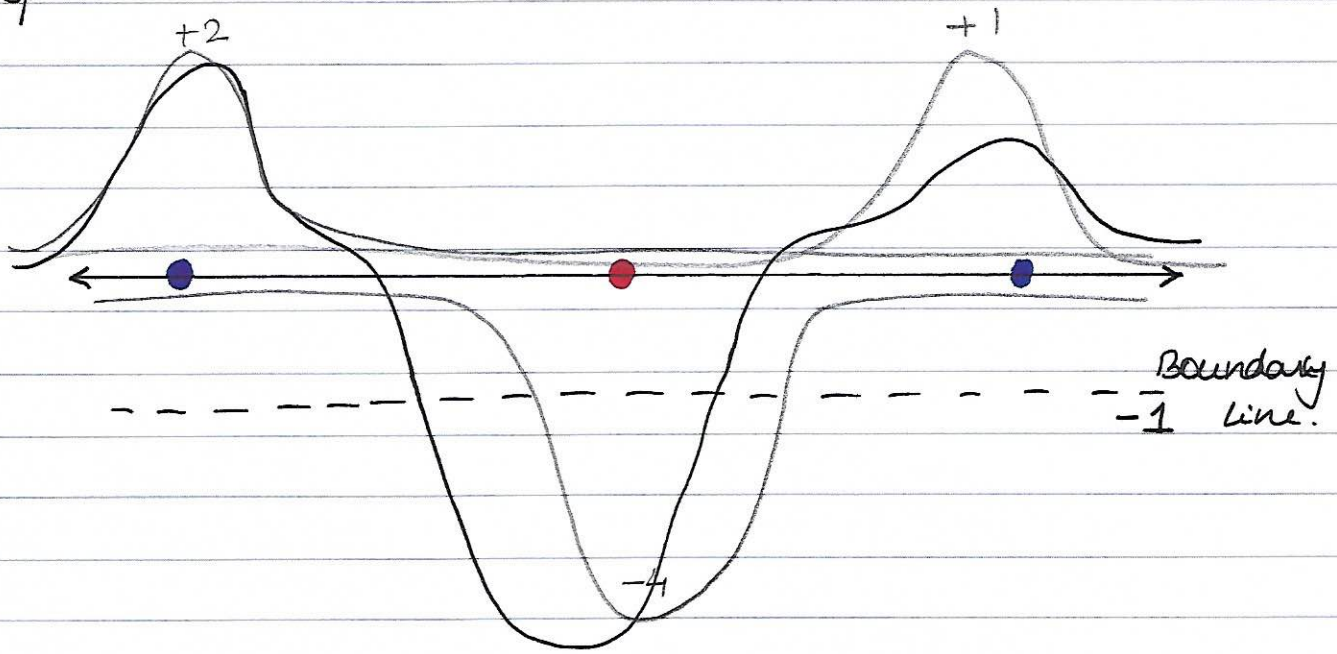
(3) Polynomial kernel $(x, y, \ldots, x^3, x^2y, \ldots y^3)$ (degree 3)

$y = x^3 + 2x^2 - x - 2$

RBF KERNEL

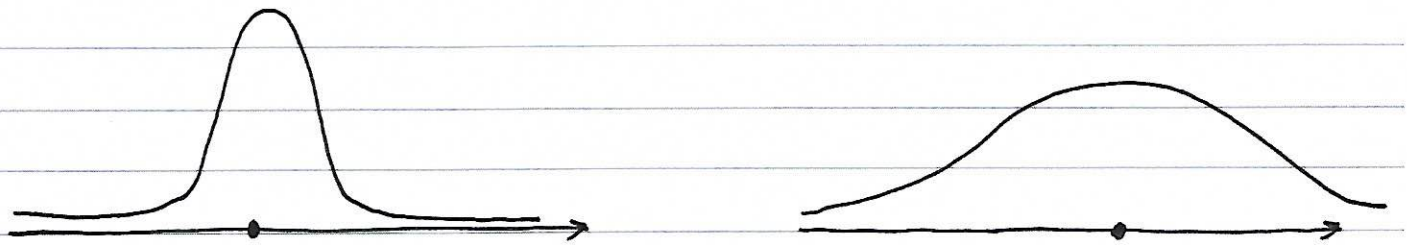
RBF : Radial Basis Function

Used to classify points that seem hard to separate in any space.



γ PARAMETER

γ (Gamma) is a hyperparameter used to tune SPF - RBF kernel models.



Large γ

Tends to overfit

Small γ

Tend to underfit

ENSEMBLE METHODS

(REGRESSION + CLASSIFICATION)

INTRODUCTION

Combining models such that the composite model is better at predicting than the individual models

Commonly used 'weak' learners in ensemble models are decision trees. (* Decision trees are high variance / overfitting algorithms)

RANDOMNESS

Introducing randomness into high variance algorithms before they are ensembled together. The introduction of randomness combats the tendency of these algorithms to overfit.

Ways to introduce randomness:

(1) Bootstrap the data : sampling the data with replacement and fitting your algorithm to the sampled data

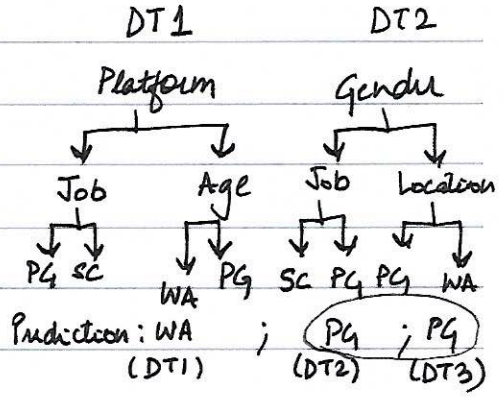
(2) Subset the features : in each split of a decision tree or with each algorithm used in an ensemble, only a subset of the total possible features are used.

RANDOM FORESTS

Given training data with 'n' features, instead of creating one decision tree using all features, create different trees using a subset of the 'n' features (randomly). Then, get prediction from all trees and the most commonly occurring prediction is the final prediction

Example.

Gender	Age	Location	Platform	Job	Hobby	App
F	15	US	iOS	School	Tennis	WA
F	25	FR	Andr.	Work	Chess	PG
M	32	UK	Andr.	School	Chess	SC



BAGGING (BOOTSTRAP AGGREGATING)

Train multiple 'weak' learners on different subsets of training data.

Then combine the weak learners using an approach such as voting, i.e. prediction made by largest majority of weak learners.

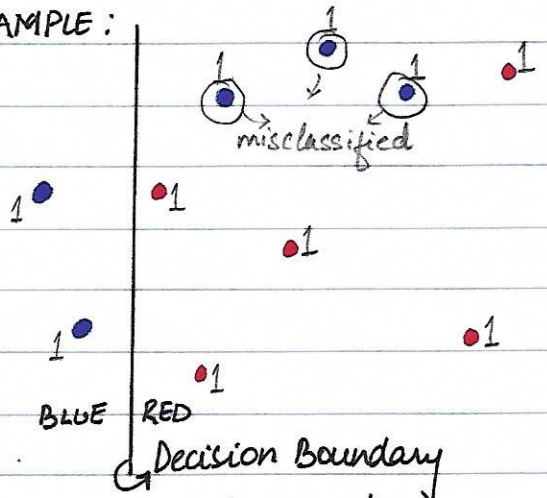
ADABOOST

(1) Create first weak learner (decision tree) such that it has max accuracy.

(2) 'Boost' misclassified points by first weak learner, such that the second weak learner will be punished more / generate higher error, if it misclassifies points misclassified by the first learner

(3) And so on.

EXAMPLE:



Step 1: With initial weights

Correct : 7

Incorrect : 3

Boost misclassified points by a factor of $7/3$, such that their weight becomes : $1 \times \frac{7}{3} = 2.33$

With new/boosted weights,

Correct : 7 (1×7)

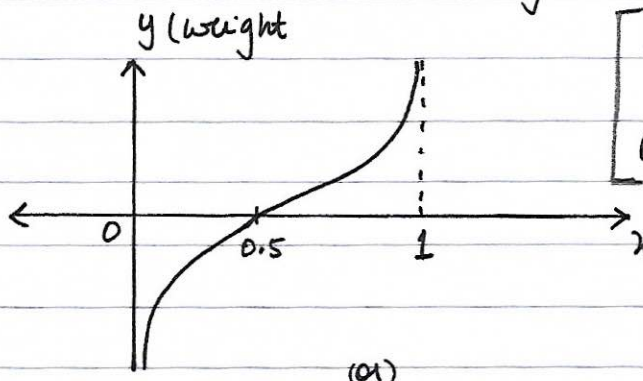
Incorrect : 7 $(\frac{7}{3} \times 3)$

In step 2, create a new weak learner that prioritises points misclassified in step 1. & ~~rep~~ & repeat a few times.

#repetitions or steps = Hyperparameter.

To combine the models / weak learners in multiple Adaboost steps, compute a weight for each weak learner using the following formula.

$$\text{weight} = \ln\left(\frac{\text{accuracy}}{1-\text{accuracy}}\right) = \ln\left(\frac{\#\text{correct}}{\#\text{incorrect}}\right)$$



$\ln\left(\frac{x}{0}\right) \to \infty$

$\ln\left(\frac{0}{x}\right) \to -\infty$

Now, prediction = 1 (or) Positive classification if sum of weights of learners is >0 or positive
= 0 otherwise.

Example,

B	Red
0.84	-0.84

Model 1
Weight = 0.84

Blue 1.3
Red -1.3

Model 2
Weight = 1.3

Blue	R
1.84	-1.84

Model 3
Weight = 1.84

Combining

0.84 1.3 1.84	-0.84 1.3 1.84	
+0.84 -1.3 1.84	-0.84 -1.3 +1.84	-0.84 -1.3 -1.84

→ -0.84, 1.3, -1.84 =

B +	B +	R −
B +	R −	R −

Ensemble methods can also be extended to regression problems. Not just for classification.

MODEL EVALUATION METRICS

CONFUSION MATRIX

	Diagnosed Sick	Diagnosed Healthy
Sick	1000 TRUE POS.	200 FALSE NEG.
Healthy	800 FALSE POS.	8000 TRUE NEG.

False Positive = Type 1 Error | False Negative = Type 2 Error

ACCURACY

$$\text{Accuracy} = \frac{\text{True Pos} + \text{True Neg}}{\text{Total}}$$

PRECISION AND RECALL

Medical Model	Spam Detector
False Pos = OK	False Pos = OK
False Neg = NOT OK	False Neg = NOT OK
OK if not all are sick	OK if not all spam is found,
But find all sick people	But if marked spam, Better be spam!
HIGH RECALL	HIGH PRECISION

$$\text{Precision} = \frac{\#\text{True Pos}}{\#\text{Pos}} = \frac{\#\text{True Pos}}{\#\text{True Pos} + \#\text{False Pos}}$$

$$\text{Recall} = \frac{\#\text{True Pos}}{\#\text{True Pos} + \#\text{False Neg}}$$

$$\text{F1 Score (a harmonic mean)} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F_β SCORE

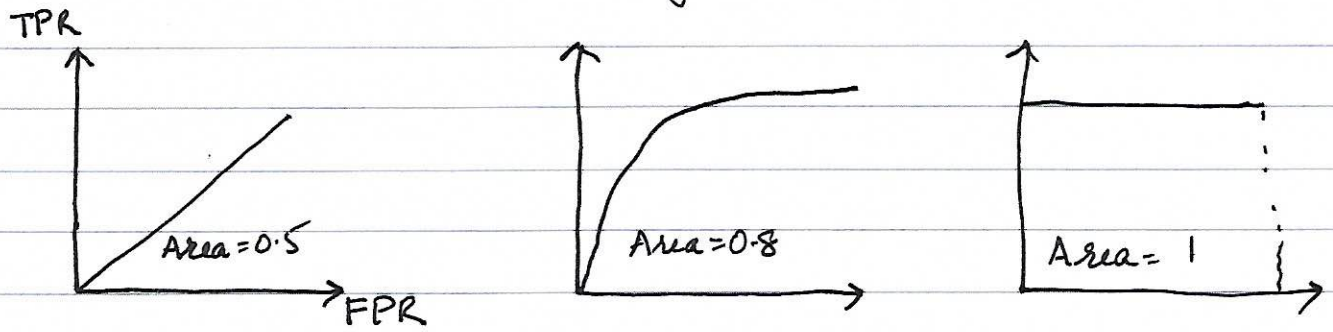
$$F_\beta = (1+\beta^2) \cdot \frac{\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

If $\beta = 0$, then we get precision
$\beta = \infty$, then we get recall
$\beta = 1$, then we get harmonic mean or F1 score

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

$$\text{True Positive Rate} = \frac{\text{True Positives}}{\text{All Positives}}$$

$$\text{False Positive Rate} = \frac{\text{False Positives}}{\text{All Negatives}}$$



Random Split | Good Split | Perfect Split

R2 SCORE FOR REGRESSION MODELS

$$R2 = 1 - \frac{\text{MSE of optimised Linear Reg. Model}}{\text{MSE of simple average Model.}}$$

Bad Model = $R2 \sim 0$
Good Model = $R2 \rightarrow 1$