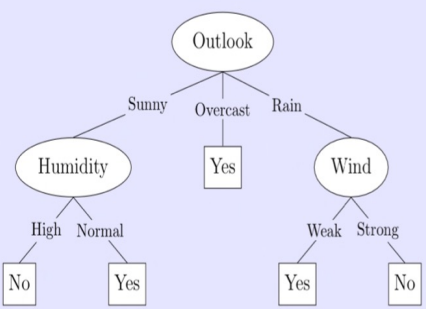
Decision Tree:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

$= \{D\}$

given a new query you just need to pass your data to DT to find your output.



$\Rightarrow$ DT

Algorithm 1 DECISIONTREETRAIN(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then              (only single class in the leaf node)
    return LEAF(guess)                                  // base case: no need to split further
else if remaining features is empty then                (if no features left to split then stop)
    return LEAF(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for                                             → Storing score of a feature (Information Gain)
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DECISIONTREETRAIN(NO, remaining features \ {f})
    right ← DECISIONTREETRAIN(YES, remaining features \ {f})
    return NODE(f, left, right)
end if
```

How to calculate these scores?

Entropy: Let's say $Y = Y_1, Y_2 \cdots Y_K$ is random variable.

$$H(Y) = -\sum_{i=1}^{K} P(Y_i) \log_b(P(Y_i))$$

for categorical values

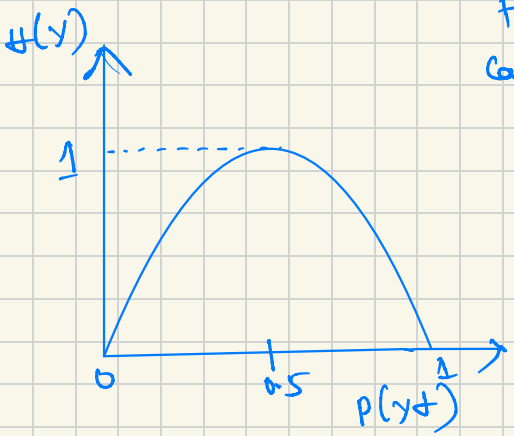
$$H(Y) = -\int P(Y_i) \log_b(P(Y_i))\, dy$$

for real values

Here $b = 2$ or $b = 2.718$

$\log_2 = \lg$

$\log_e = \ln$



for two class classification, Entropy can never be more than 1.

$$P(y+) = 1 - P(y-)$$

$y = y_1, y_2 \cdots y_k$

↳ if they are equiprobable then entropy is maximum.

Ex- let $P(y+) = P(y-) = 0.5$

$$E(y) = -P(y+)\log(P(y+)) - P(y-)\log(P(y-))$$

$$= -0.5 \log 0.5 - 0.5 \log 0.5$$

$$= -(\log 0.5) = -(\log 1 - \log 2)$$

$$= 1.$$ ✓

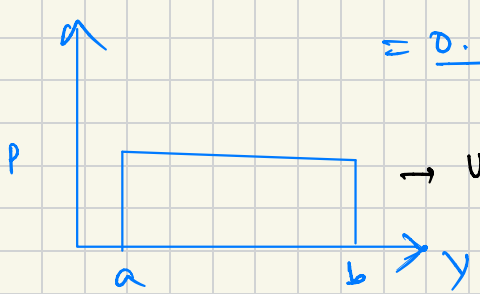
If any one of random variable is most probable & others are 0 then, ⟶ entropy is minimal.

let $P(y+) = 1 \quad P(y-) = 0$

$$E(y) = -P(y+)\log(P(y+)) - P(y-)\log(P(y-))$$

$$= -1 * 0 - 0 * \log 0$$

$$= 0.$$



↳ Uniform distribution, so, if r.v y is equiprobable then it has maximum Entropy.

Y_1 → less peaked
Y_2 → very peaked

So, Y_1 has more data variance
Whears Y_2 has low data variance

So, $H(Y_1) > H(Y_2)$

entropy of Y_1 always greater than Y_2.

Partition of Data:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

initial Entropy $H_D(Y) = -\frac{8}{14} \times \log\left(\frac{8}{14}\right) - \frac{6}{14} \times \log\left(\frac{6}{14}\right)$

$= -.57 \times (-.8074) - .42 \times (-1.2224)$

$= .46 + .52$

$= .98$

Let, D → D_1 (5 rows) $H_{D_1}(Y) = 0.97$
D → D_2 (4 rows) $H_{D_2}(Y) = 0$
D → D_3 (5 rows) $H_{D_3}(Y) = .97$

Information gain (IG):

$$\left(.97 \times \frac{5}{14} + 0 \times \frac{4}{14} + .97 \times \frac{5}{14}\right) - .98$$

weight entropy after partitioning dataset — entropy before partitioning

$$IG(Y, var) = H_D(Y) - \left[\sum_{i=1}^{k} \frac{|D_i|}{|D|} * H_{D_i}(Y)\right]$$

Gini Impurity : (I_G)

~ Similar to entropy

$$I_G(Y) \neq IG(Y)$$

for random variable $Y = Y_1, Y_2 \dots Y_k$

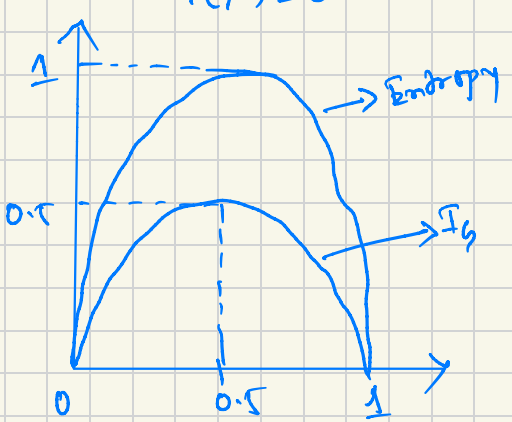
$$I_G(Y) = 1 - \sum_{i=1}^{k} (P(Y_i))^2$$

Case 1: $P(Y+) = 0.5$, $P(Y-) = 0.5$

$$I_G(Y) = 1 - [(0.5)^2 + (0.5)^2]$$
$$= 1 - [.25 + .25]$$
$$= .5$$

Case 2: $P(Y+) = 1$, $P(Y-) = 0$

$$I_G(Y) = 1 - [1^2 + 0^2]$$
$$= 1 - 1 = 0$$



max value for Gini Impurity is 0.5.

H(Y) is computationally intensive due to log calculation whereas computing square has less computational cost compare to log computation.

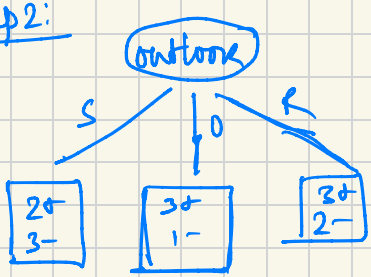
So, I_G is preferable.

** Construct a DT:

Step 1: $D = \{y^+, y^-\} = \{8, 6\}$ & $H(y) = .98$

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Step 2:



$$\left[-\frac{5}{14}\left(\frac{2}{5}\log\left(\frac{2}{5}\right)+\frac{3}{5}\log\left(\frac{3}{5}\right)\right)\right.$$

$$-\frac{4}{14}\left(\frac{3}{4}\log\left(\frac{3}{4}\right)+\frac{1}{4}\log\left(\frac{1}{4}\right)\right)$$

$$\left.-\frac{5}{14}\left(\frac{3}{5}\log\left(\frac{3}{5}\right)+\frac{2}{5}\log\left(\frac{2}{5}\right)\right)\right]-.98$$

$$=\left[-\frac{5}{7}\left(\frac{2}{5}\log\left(\frac{2}{5}\right)+\frac{3}{5}\log\left(\frac{3}{5}\right)\right)-\frac{4}{14}\left(\frac{3}{4}\log\left(\frac{3}{4}\right)+\frac{1}{4}\log\left(\frac{1}{4}\right)\right)\right]$$

$$-.98$$

$$=-\frac{5}{7}(-.528-.4422)+\frac{4}{14}(-.3112-.5)$$

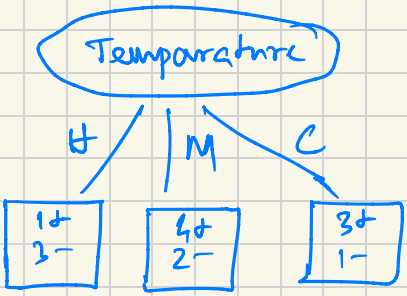
$$-.98$$

$$=(.3914+.2312)-.98$$

$$=|-.6231-.98|$$

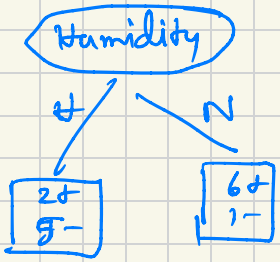
$$=.35$$

Step 3:



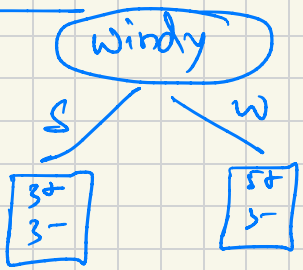
find IG as above

Step 4:



find IG as above

Step 5:



find IG as above

Step 6:

$IG(Y, \text{outlook}) = 0.35$

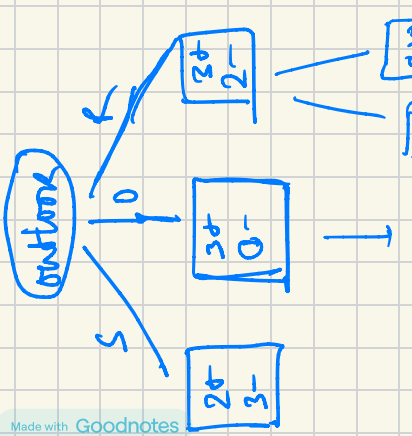
$IG(Y, \text{Temparature}) = \_\_$

$IG(Y, \text{Humidity}) = \_\_$

$IG(Y, \text{Windy}) = \_\_$

We will choose the heighest IG as root.

$$IG(Y,f) = H_D(Y) - \sum_{i=1}^{K} \frac{|D_i|}{|D|} * H_{D_i}(Y)$$

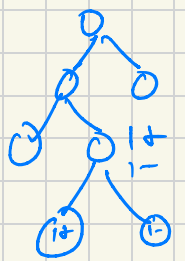


We are recursively branching each nodes based on I_G.

** three points where to stop growing the tree —

(1) pure node → stop growing the tree.

(2) Can't grow the tree anymore due to lack of points



In this case no point of growing the node.
Suppose, your dataset has 10k points & you have only one points which follows one path. That point may be noisy. (It will overfit)

(3) if we are too deep → stop growing the tree.

As, the depth of the tree ↑ → chance of overfitting ↑

depth is small → underfit.

In DT → depth of the tree is a hyperparameter.
↳ we can use CV to find it.

Pruning: Remove subtrees for better generarization (decrease variance)

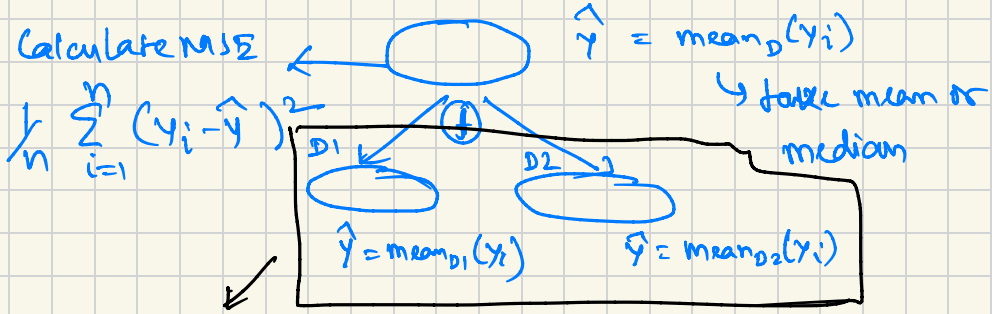
Prepruning: Early stopping

Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set

Prepruning is faster but postpruning is more accurate.

DT for Regression:

- IG used for classification
- MSE or MAE is used for regression



find weighted mean square

$w_1 MSE_{D1} + w_2 MSE_{D2}$

Now subtract $\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y})^2 - \left[w_1 MSE_{D1} + w_2 MSE_{D2}\right]$

whichever split minimizes MSE, should consider for split.

Random Forest:

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:
for i = 1 to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values (F_1, ... , F_f)
  for i = 1 to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match
    F_i
    Call BuildTree(N_i)
  end for
end if

Adaboost Algorithm:

Given n examples (x_i, y_i) where $x \in X$ & $y \in \pm 1$

(1) Initialize $D_1(i) = 1/n$

$\longrightarrow D_1$ is first dataset.

(2) for t in $(1$ to $T)$:

i) Train weak classifier on Distribution $D_t(i)$, $h_t(x) : x \to \{+1, -1\}$

ii) Find error with $h_t(x)$

$$\varepsilon_t = \sum_{i=1}^{n} D_t(i) \left[y_i \neq h_t(x_i) \right]$$

if $\varepsilon_t < 0.5$ then stop here

else:

find weight for each record of the dataset

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$

iii) update $D_{t+1}(i) = \dfrac{D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$

Where $Z_t = \sum_{i=1}^{n} D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\}$

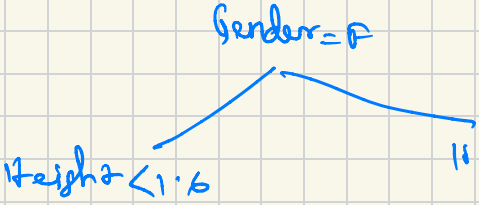
output classifier is

$$h(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

Gradient boosting Algorithm :

Height	Favorite Color	Gender	weight (Kg)	1st residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

Avg weight = 71.2



Dimensionality Reduction 8

volume of a d-dim hyperspace

$$V_d(r) = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \qquad V_3(r) = \frac{r^3 \pi^{d/2}}{\Gamma\left(\frac{5}{2}\right)}$$

$$\Gamma(5/2) = \int_0^{5/2} e^{-x} x^{\frac{5}{2}-1} dx$$

$$V_3(r) = \frac{r^3 \pi^{3/2}}{\frac{3}{4}\sqrt{\pi}}$$

$$\Gamma\left(1+\frac{3}{2}\right) = \frac{3}{2}\Gamma\frac{3}{2} = \frac{3}{2}\times\frac{1}{2}\sqrt{\pi} = \frac{3}{4}\sqrt{\pi}$$

$$V_3 r = \frac{4}{3}\pi r^3 \quad \checkmark$$

$$\frac{\text{vol(sphere)}}{\text{vol(cube)}} = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \Big/ (2r)^d$$

$$\frac{\pi^{d/2}}{2^d \frac{d}{2}\Gamma\frac{d}{2}}$$

Gamma function

$$\Gamma(n) = \int_0^\infty e^{-x} x^{n-1} dx$$

$$\Gamma(1) = 1, \quad \Gamma(1/2) = \sqrt{\pi}$$

$$\Gamma(n+1) = n\Gamma(n)$$

$$\Gamma(n+1) = n!$$

$$\Gamma(n)\Gamma(1-n) = \frac{\pi}{\sin n\pi}$$

two way we can solve PCA

① Maximum variance formulation

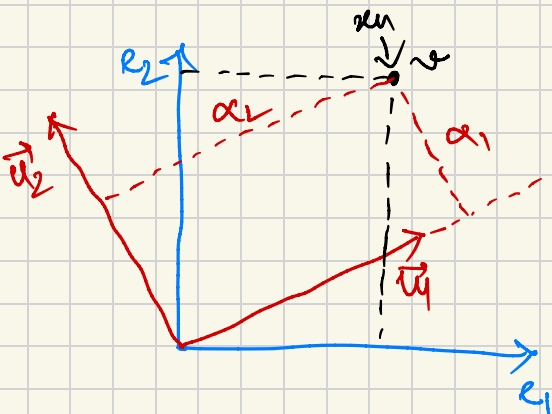
② Minimum Error formulation

Minimum Error Formulation

PCA can be obtained by minimizing the reconstruction error.

Find a transformation that minimizes:

$$J(x,\tilde{x}) = \frac{1}{N}\sum_{n=1}^{N} \| x_n - \tilde{x}_n \|$$



$$v = a_1 u_1 + a_2 u_2 \cdots$$

$$x_n = \sum_{i=1}^{D} a_{ni} u_i$$

$$a_{ni} = x_n^T u_i$$

Therefore, we get the projection onto a new orthonormal basis.

$$u_i^T u_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$

$$x_n = \sum_{i=1}^{D} (x_n^T u_i) u_i$$

For reconstruction, we can use the first M elements of the basis and a shared offset for the remaining dimensions.

$$\tilde{x}_n = \sum_{i=1}^{M} (x_n^T u_i) u_i + \sum_{i=M+1}^{D} b_i u_i$$

$$x_n - \tilde{x}_n = \sum_{i=1}^{D} (x_n^T u_i) u_i - \sum_{i=1}^{M} (x_n^T u_i) u_i - \sum_{i=M+1}^{D} b_i u_i$$

$$= \sum_{i=M+1}^{D} (x_n^T u_i) u_i - \sum_{i=M+1}^{D} b_i u_i$$

$$= \sum_{i=M+1}^{D} (x_n^T u_i) u_i - b_i u_i$$

$$J(x, \tilde{x}) = \frac{1}{N} \sum_{n=1}^{N} \| x_n - \tilde{x}_n \|^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left\| \sum_{i=M+1}^{D} (x_n^T u_i) u_i - b_i u_i \right\|^2$$

Now we need to minimize u_i & b_i.

$$\frac{\partial L(x, \tilde{x})}{\partial b_i} = -\frac{1}{N} \sum_{n=1}^{N} 2 \left(\sum_{i=M+1}^{D} (x_n^T u_i) u_i - b_i u_i \right) u_i = 0$$

$$= \frac{2}{N} \sum_{n=1}^{N} \left(\sum_{i=M+1}^{D} x_n^T u_i - b_i \right) = 0$$

$$\Rightarrow \boxed{b_i = \sum_{i=M+1}^{D} \bar{x}^T u_i}$$

$$J(x,\tilde{x}) = \frac{1}{N}\sum_{n=1}^{N}\left\| \sum_{i=M+1}^{D} \left(x_n^T u_i - \tilde{x}^T u_i\right) u_i \right\|^2$$

$$\frac{\partial J(x,\tilde{x})}{\partial u_i} = \frac{1}{N}\sum_{n=1}^{N} 2\left[\sum_{i=M+1}^{D}\left(x_n^T u_i - \tilde{x}^T u_i\right) u_i \left(x_n^T - \tilde{x}^T\right)\right]$$

$$= \frac{2}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}\left[\left(x_n^T u_i^T u_i - \tilde{x}^T u_i^T u_i\right)\left(x_n^T - \tilde{x}^T\right)\right]$$

$$= \frac{2}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} u_i^T \underbrace{\left(x_n^T - \tilde{x}^T\right)\left(x_n^T - \tilde{x}^T\right)}_{S} u_i = 0$$

$$\Rightarrow \sum_{i=M+1}^{D} u_i^T S u_i = 0$$

So, if we put $u_i = 0$ then $J(x,\tilde{x})$ will be minimized

so, we need to put constraint

$$J(x,\tilde{x}) = \sum_{i=M+1}^{D} u_i^T S u_i + \lambda_i\left(1 - u_i^T u_i\right)$$

we know u_i is orthonormal.

for $M = 0$ & $D = 1$

$$\frac{J(x,\tilde{x})}{\partial u_1} = \frac{\partial}{\partial u_1}\left(u_1^T S u_1\right) + \lambda_1 \frac{\partial}{\partial u_1}\left(1 - u_1^T u_1\right)$$

$$= u_1 S - \lambda_1 u_1 = 0$$

$$\Rightarrow \boxed{u_1 S = \lambda_1 u_1}$$

Here u_i is the corresponding eigenvector of eigenvalue λ_i.

Our intension is to find top $(D-M)$ eigenvalues so that J will be minimized.

K Means clustering:

Consider this as a minimization problem.

Data: $X = \{x_1, x_2 \cdots x_N\}$ $x_n \in \mathbb{R}^D$

Goal: partition into K clusters by minimizing

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \| x_n - \mu_k \|^2$$

find cluster assignment (latent var)

$z_{nk} \in \{0, 1\}$

and

Cluster means $\mu_k \in \mathbb{R}^D$

$z_{nk} = 1$ iff point n is assigned to cluster k.

$$z_{nk} = \begin{bmatrix} z_{n1} \\ z_{n2} \\ \vdots \\ z_{nK} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

1-hot-encoding.

Algorithm (EM algorithm)

1. Initialize with a random $\mu_k \in \mathbb{R}^D$
2. Repeat until convergence
   - → find the assignment (fixed means)

E-step

$$z_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

   - → find the means (fixed assignment)

M-step

$$\mu_k = \frac{\sum_n z_{nk} x_n}{\sum_n z_{nk}}$$

EM → Expectation Maximization.

J is non-convex for μ_k and z_{nk} together

2 K-means converges to a local minima.

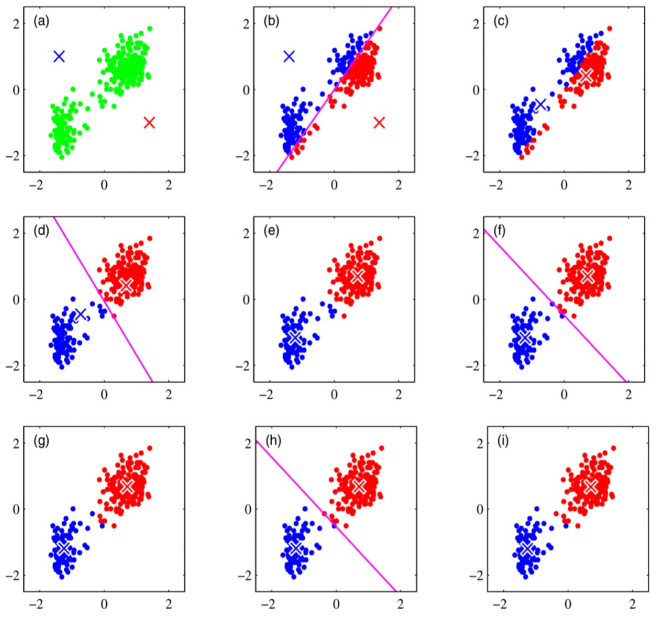
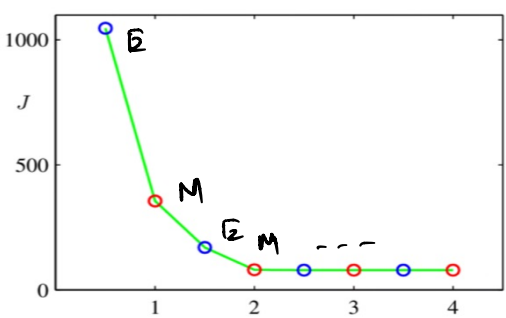


Figure 9.1 Illustration of the K-means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of the algorithm.

446

Figure 9.2 Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.



$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \| x_n - \mu_k \|^2$$

It is a convex function for μ_k if z_{nk} is fixed.

So, $$\frac{\partial J}{\partial \mu_k} = -2 \sum_{n=1}^{N} z_{nk} (x_n - \mu_k)^T = 0$$

$$\Rightarrow \sum_{n=1}^{N} z_{nk} x_n - \mu_k \sum_{n=1}^{N} z_{nk} = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{n=1}^{N} z_{nk} x_n}{\sum_{n=1}^{N} z_{nk}}$$

Summing up the datapoints within k class

No of points within k class.