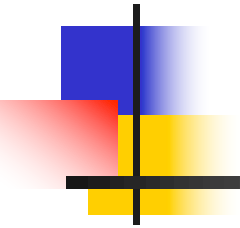


Clustering and classification (Week-10: L46-L50)

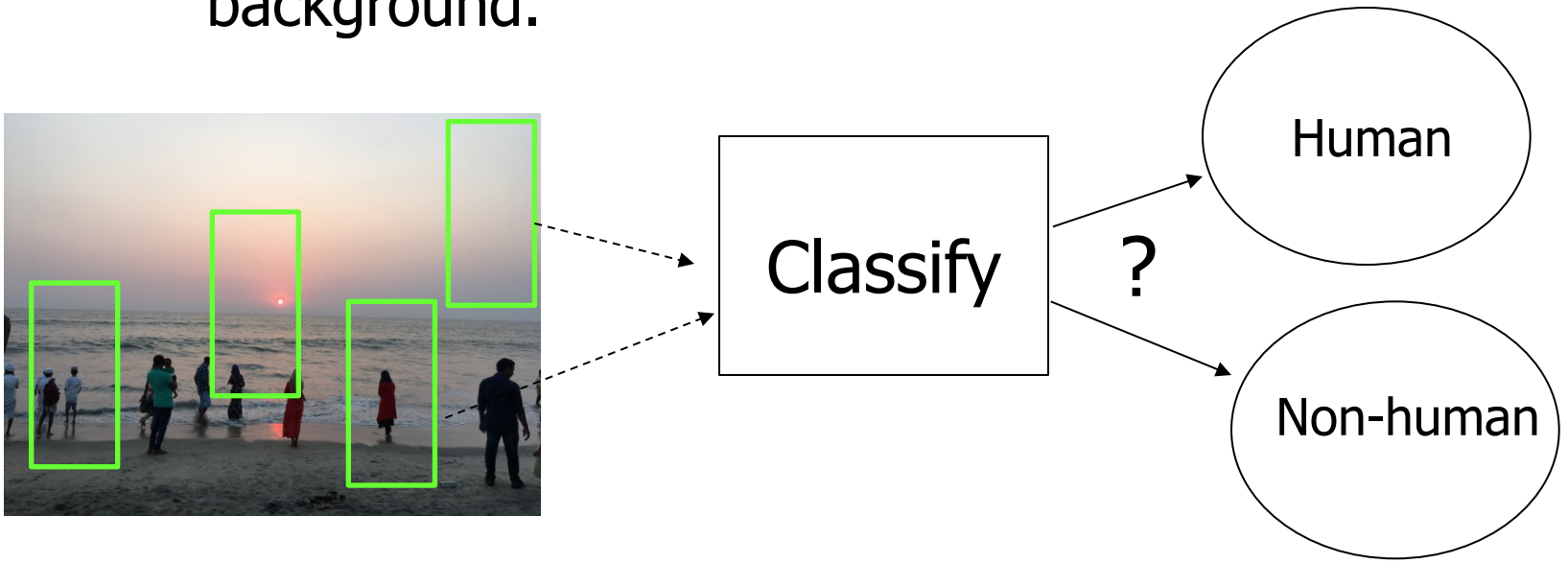
Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Classification

Classification:

Task of assigning a known category or class to an object.

- detection of pedestrian in an image patch.
- recognition of an alphabet given a 2-D pattern.
- assigning a pixel of an image to its foreground or background.



Clustering

Clustering: the task of organizing objects into groups whose members are *similar in some way*.

Cluster: a collection of objects *similar to each other*, but dissimilar to the objects belonging to other clusters .

- Regions of homogeneity in an image.
  - Segments.
- Grouping of similar components.

Class and cluster

A class: well studied group of objects identified by their common properties or characteristics.

A cluster: a group with ‘loosely’ defined similarity among the objects.

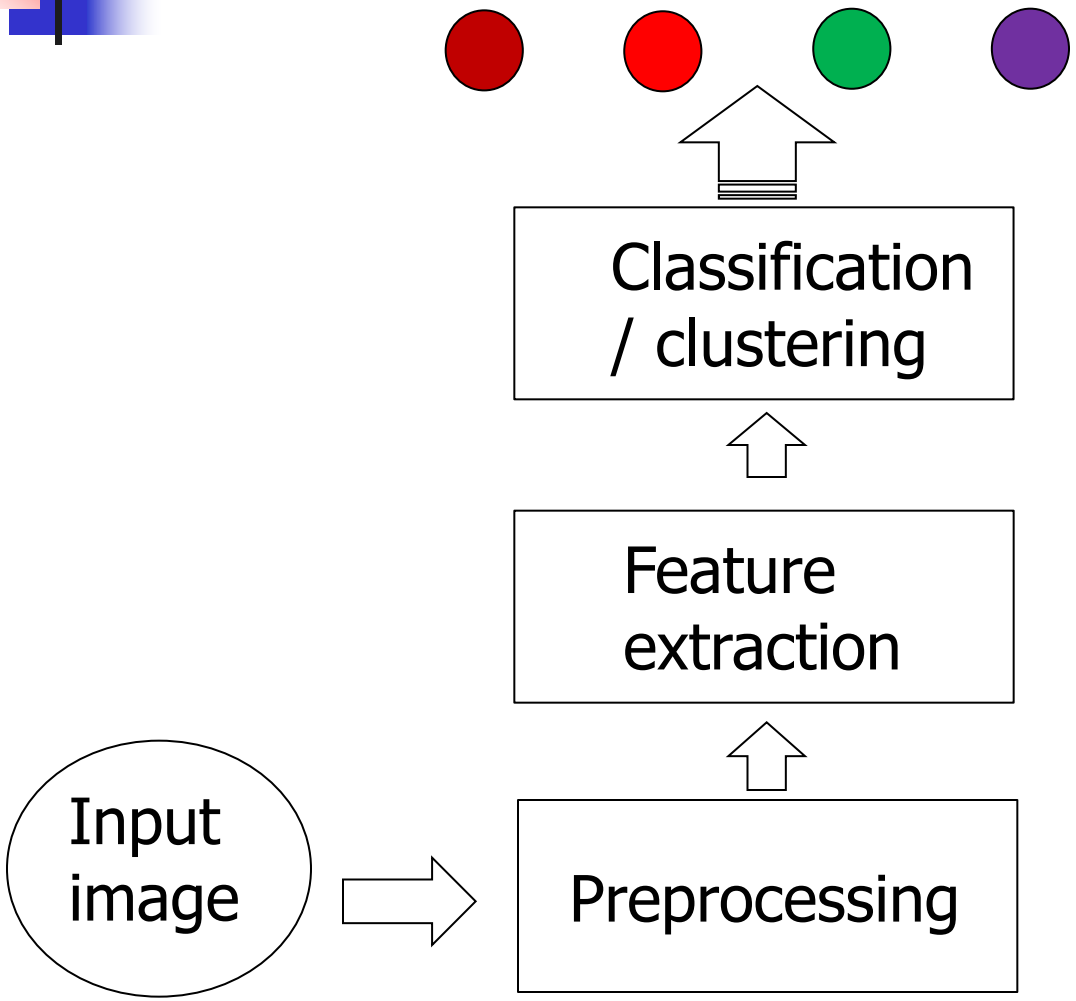
- Potential to form a class.

Clustering: Motivation

- finding representatives for homogeneous groups
  - to reduce data.
- discovering natural groups or categories.
  - to describe by their unknown properties.
- finding relevant groups.
  - major groups in the given context.
    - segments of an image.
- detecting unusual data objects
  - outliers.

https://www2.cs.duke.edu/courses/fall03/cps260/notes/lecture18.pdf

Processing pipeline



Supervised and unsupervised learning

Clustering and classification: learning problems.

- Learning about groups or categories in data.

Unsupervised learning: learns in the absence of almost any prior knowledge of groups.

- Sometimes the number of groups provided.
- clustering

Supervised learning: exploits knowledge about the classification problem, such as example instances of classes.

- Training samples with class labels.
- Finds features suitable for predicting classes.
- Classification

Other variations in learning frame work

Semi-supervised learning:
learns by making use of unlabeled data for training in conjunction with a small amount of labeled data.

- Usually sizes of unlabeled data large, and labeled data small.
- falls between unsupervised learning and supervised learning.

Reinforcement learning:
learning by feedback from a teacher or a critique, in the form of reward / punishment, yes / no, true / false, etc.

Clustering: Three major components

A. The similarity (distance) measure between two data samples.

- L_1, L_2, L_p norms.

B. Criterion function to evaluate clusters.

- Intra-cluster cohesion
  - Sum of Squares Error (SSE)
- Inter cluster separation

C. The clustering algorithms.

- K-means
- K-medoids
- Mixture of Gaussians

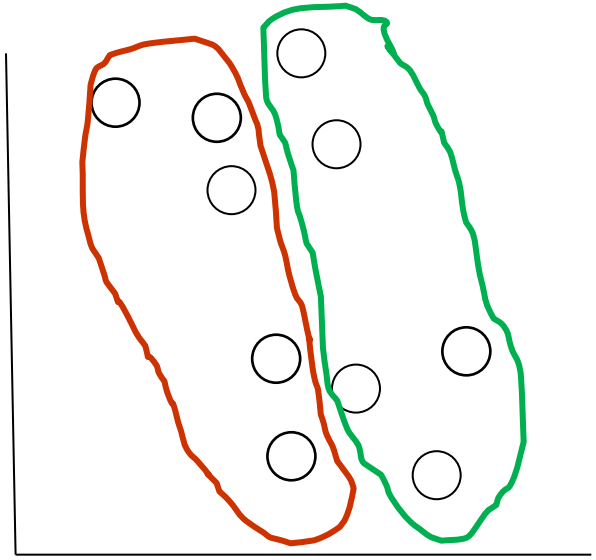
Other approaches

- Hierarchical clustering
- Graph based approaches, etc.

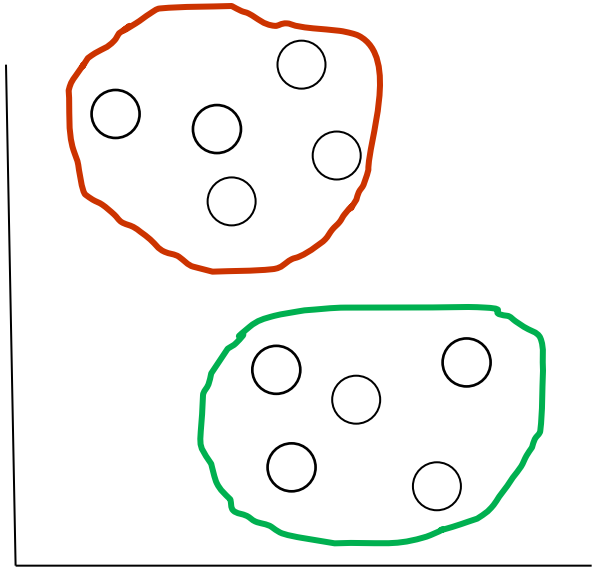
Homogeneity and Separation Principles

- **Homogeneity:** Elements within a cluster are close to each other.
  - e.g. compute the average distance of these elements from the cluster center.
    - Should be small.
- **Separation:** Elements in different clusters are further apart from each other.
  - e.g. compute average distance of pairs of cluster centers.
    - Should be large.

Several choices …



A bad clustering example



A good clustering example

K-means clustering

- Given n data points, compute k partitions (clusters) in them so that it minimizes the sum of square of distances between a data point and the center of its respective partition (cluster).

Optimization problem

Minimization of

$$E = \sum_{k} \sum_{\forall x \in c_k} \|x - c_k\|^2$$

where $$c_k = \frac{1}{|C_k|} \sum_{\forall x \in C_k} x$$

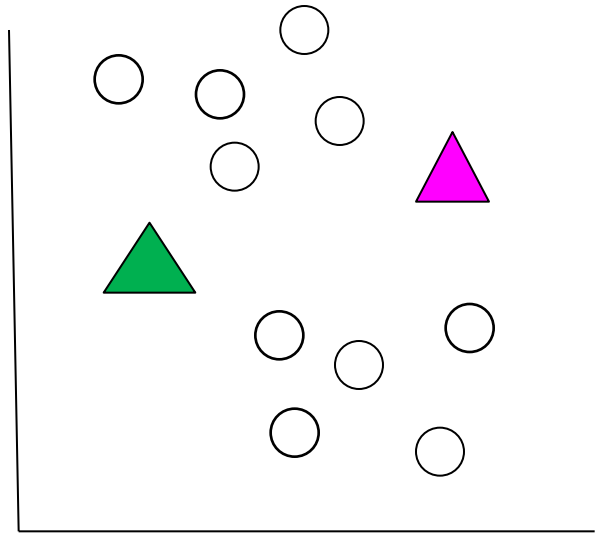
An NP-complete problem (K>1).

The Lloyd algorithm

- Given k initial centers, assign a point to the cluster represented by its center, if it is the closest among them.
- Update the centers.
- Iterate above two steps, till the centers do not change their positions.
- Trying to minimize the energy function defined by the sum of divergences of each cluster from its center.
- Convergence not guaranteed, but works well in practice.
- May get stuck at local minima.

$$E = \sum_{k} \sum_{\forall x \in c_k} \|x - c_k\|^2$$

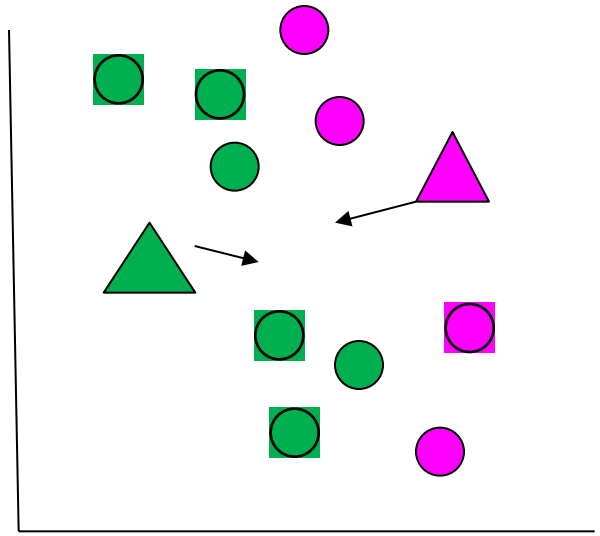
K-means: example (k=2)



Choose initial centers.

Compute partitions.

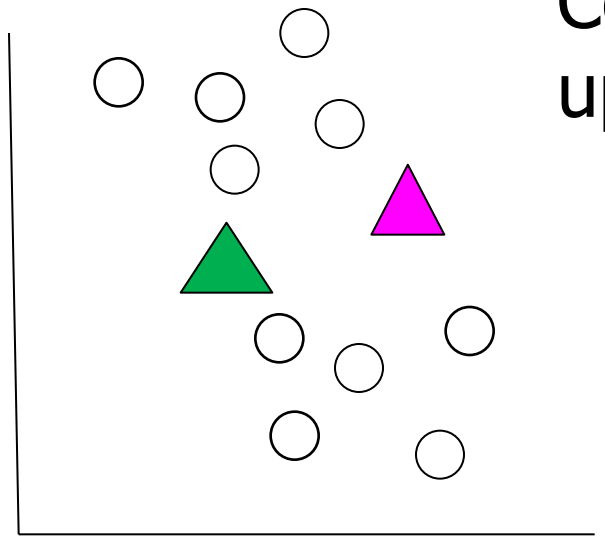
K-means: example (k=2)



Compute partitions.

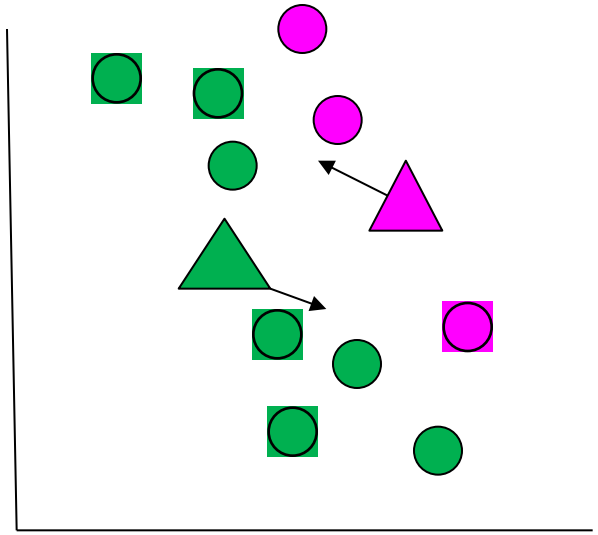
Update centers.

K-means: example (k=2)



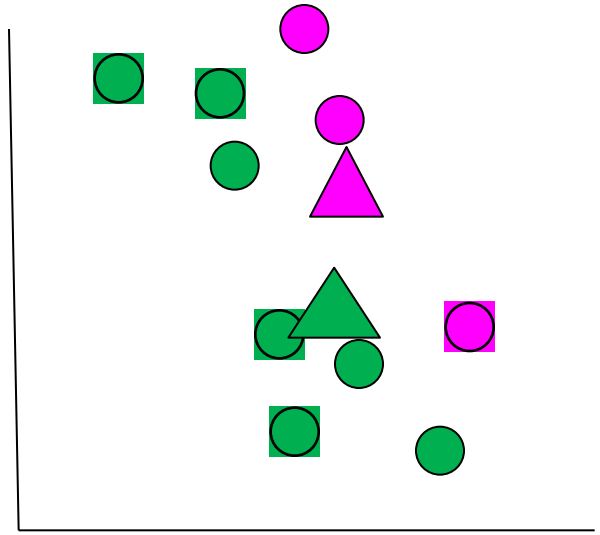
Compute new partitions with updated centers.

K-means: example (k=2)

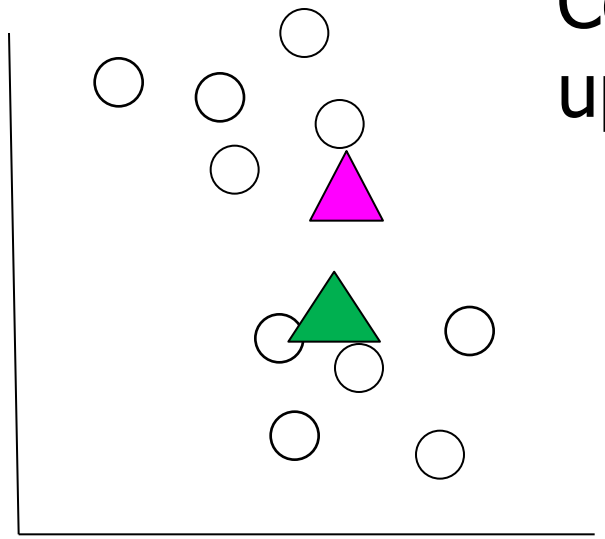


Update centers.

K-means: example (k=2)

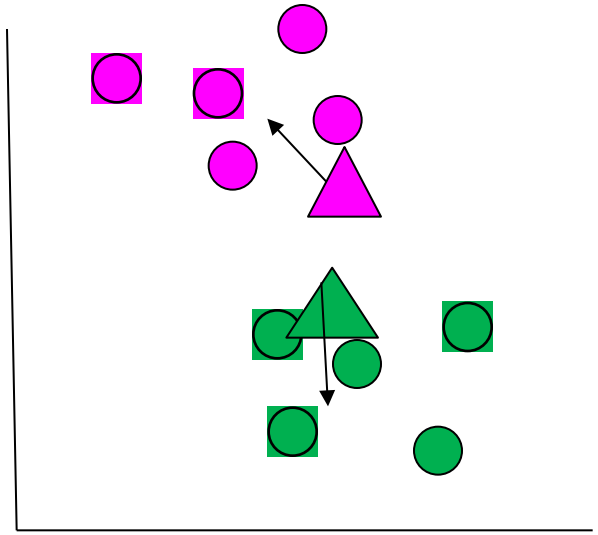


K-means: example (k=2)



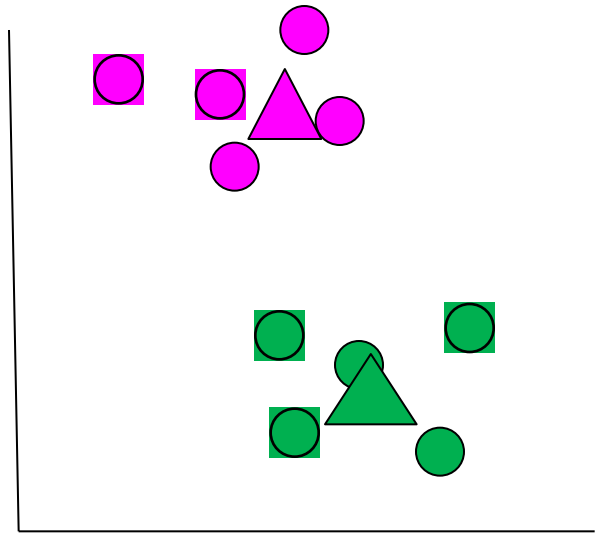
Compute new partitions with updated centers.

K-means: example (k=2)



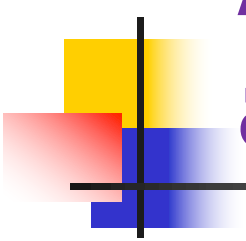
Update centers.

K-means: example (k=2)



Stop at no change (or a very little change in cluster centers).

A more conservative approach



- Lloyd algorithm fast but not necessarily causing better convergence.
- A more conservative approach to move one data point at a time provided overall cost gets reduced.
- A greedy approach by choosing the transfer of a data point from a class (say, i) to another class (say, j), which causes the best (maximal) cost reduction at that step.

Greedy K-Means Algorithm

Select an arbitrary partition P into k clusters.

Repeat the steps below till convergence of cost.

```
maxReduction=0;
for every cluster C do
for every element i not in C do
    reduction=cost(P) - cost(P_{i→C})
    if (reduction>0)
        if (reduction> maxReduction)
            maxReduction=reduction
            Update C and the cluster containing i.
        endif
    endif
endfor
endfor
```

k-Medoids clustering

- A medoid: the representative element of a set of data point with minimal average dissimilarity with other data points in the set.
  - always restricted to be a member of the data set.

Given a set X=$\{x_i | i=1,2,..,n\}$, its medoid is given by

$$medoid = \underset{x_k}{\operatorname{argmin}} \frac{1}{n-1} \sum_{\forall j, j \neq k} \|x_j - x_k\|$$

Cost to be minimized in k-Medoids clustering:

$$E = \sum_{k} \sum_{\forall x \in C_k} \|x - m_k\|^2$$

m_k: Medoid of C_k.

k-medoids clustering

- Given k initial centers, assign a point to the cluster represented by its center, if it is the closest among them.
- Update the medoids.
  - Very expensive!!
- Iterate above two steps, till the medoids do not change their positions.

$$E = \sum_{k} \sum_{\forall x \in C_k} \|x - m_k\|^2$$

- Randomly choose an element of a different cluster, swap with the medoid element and update the medoids.
- If the cost decreases, accept the swap.
- Continue till it converges.

Also known as Partitioning around medoids (PAM).

Mixture of Gaussians

- Each cluster center is augmented by a covariance matrix, whose values are re-estimated from corresponding samples.
  - Mahalanobis distance function:

$$d(x, \mu_k; \Sigma_k) = (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)$$

Cluster center

Covariance matrix

Technique could be refined by computing probabilities of belongingness to a cluster.

Parametric PDF:

$$p(x|\{\pi_k, \mu_k, \Sigma_k\}) = \sum_k \pi_k N(x|\mu_k, \Sigma_k)$$

$$N(x|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} d(x, \mu_k; \Sigma_k)}$$

Mixing coefficients

Expectation Maximization (EM) Algorithm

$$z_{ik} = \frac{1}{Z_i}\pi_k N(x_i|\mu_k, \Sigma_k)$$

Normalizing factor

$$Z_i = \sum_k z_{ik}$$

- Start with initial set :$\{\pi_k, \mu_k, \Sigma_k\}$.
- E-Step (Expectation stage)
  - Compute likelihood (z_{ik}) of x belonging to kth Gaussian cluster.
  - Assign x to the mth cluster whose *likelihood* is maximum.

Optional step. Decision to be taken at the end.

- M-Step (Maximization Stage)
  - Re-estimate parameters ($\{\pi_k, \mu_k, \Sigma_k\}$) from class distribution
- Iterate above two steps till it converges.

Parameter re-estimation

$$z_{ik} = \frac{1}{z_i} \pi_k N(x|\mu_k, \Sigma_k)$$

Normalizing factor

$$N_k = \sum_i z_{ik}$$

Expected number of pixels in class k.

$$\mu_k = \frac{1}{N_k} \sum_i z_{ik}\, x_i$$

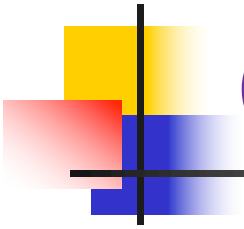
$$\Sigma_k = \frac{1}{N_k} \sum_i z_{ik}\, (x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}$$

Classification approaches

Classification:
Task of assigning a known category or class to an object.

- Probabilistic
  - Bayesian classification
- Distance based
  - K-Nearest neighbor
- Discriminant analysis
  - Linear discriminant analysis (LDA)
- Artificial neural network (ANN)
  - Feed-forward neural network.



Classification problem

- Given a labelled data sets:

  $\{(y_i,x_i)\}$, i=1,2,…,n such that x_i in R^n and y_i is the class of x_i, an element of the finite set of classes.

  - y_i could be +1 or -1 for a two class problem.

  Design a classifier C which assigns class y_i (output) to x_i (input).

Risk of Classification

- Given a labelled data sets:

  $\{(y_i,x_i)\}$, i=1,2,…,n such that x_i in R^n and y_i is the class of x_i, an element of the finite set of classes.

  - y_i could be categorical or {+1 , -1} for a two class problem.

  Design a classifier C which assigns class y_i (output) to x_i (input).

$$Risk(C) = \frac{1}{n}\sum_i 1_{y_i \neq C(x_i)}$$

Bayesian classification minimizes the risk.

Bayesian Classification

- Based on Bayes' Theorem.
- Probabilistic prediction of belongingness to a class.
- A simple Bayesian classifier: *Naïve Bayesian classifier.*
- Each training example can increase/decrease the probability of the predicted class.
- Prior knowledge can be combined with observed data

Courtesy: hanj.cs.illinois.edu/bk3/bk3_slides/08ClassBasic.ppt

Bayesian Inference

1. Conditional probability

$$P(A \text{ and } B) = P(A)\,P(B|A) = P(B)\,P(A|B)$$

Prior probability, the probability of the hypothesis on previous knowledge

2. Bayes' Theorem (Thomas Bayes (1763))

$$P(H|X) = \frac{P(H)\,P(X|H)}{P(X)}$$

Likelihood function, probability of the data given the hypothesis

Posterior probability, the probability of the hypoth. (assigning a class) given the data.

Unconditional probability of the data, a normalizing constant ensuring the posterior probabilities sum to 1.00

Bayes' classification rule

Given training data ***X***, *posteriori probability of a hypothesis* ***H***, P(***H***|***X***), follows the Bayes' theorem

$$P(H|X) = \frac{P(H)\,P(X|H)}{P(X)}$$

Bayes' classification rule:

Assign C_i to X iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *k* classes.

Challenges:

requires prior knowledge of probabilities of classes and their distributions in multidimensional feature spaces.

Problem statement

- Input: a training set of tuples and their associated class labels.
  - each tuple is represented by an n-D attribute vector $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$.
  - Let there be m classes $C_1, C_2, \ldots, C_m$.
- To derive the maximum posteriori, i.e., the maximal $P(C_i|X)$.

$$P(C_i|X) = \frac{P(C_i)\,P(X|C_i)}{P(X)}$$

- Since $P(X)$ is constant for all classes, only $P(C_i)\,P(X|C_i)$ needs to be maximized.

Naïve Bayes Classifier

Works on a simplified assumption:
attributes are conditionally independent (i.e., no dependence relation between attributes).

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

Significant reduction of the computation cost

- requires only the class distributions.

Convenient to estimate P($x_i|C_k$)

For a categorical or discrete variable:

- fraction of times the value occurred in a class.

For a continuous variable:

- may use parametric modeling of Gaussian distribution.

Likelihood estimation in Naïve Bayes Classifier

Likelihood: $$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

To estimate P(x_i|C_k)

For categorical or discrete variable:

- fraction of times the value occurred in a class.

For continuous variable:

- may use parametric modeling of Gaussian distribution.

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

An Example: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Computation of class prior

age	income	student	credit_rating	s_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

hanj.cs.illinois.edu/bk3/bk3_slides/08ClassBasic.ppt

Likelihood estimation: age =“<=30”

age	income	student	credit_rating	s_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6

hanj.cs.illinois.edu/bk3/bk3_slides/08ClassBasic.ppt

Likelihood estimation: income =“medium”

age	income	student	credit_rating	s_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

hanj.cs.illinois.edu/bk3/bk3_slides/08ClassBasic.ppt

Likelihood estimation: student =“yes”

age	income	student	credit_rating	s_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

hanj.cs.illinois.edu/bk3/bk3_slides/08ClassBasic.ppt

Likelihood estimation: credit_rating ="fair"

age	income	student	credit_rating	s_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

Likelihood estimation: $P(X|C_i)$

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$:

P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

Estimation of posterior: $P(C_i|X)$ and class assignment

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(C_i|X) = P(X|C_i)*P(C_i)$:

P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding Zero-Probability

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

All the conditional probabilities should be non-zero, else likelihood becomes zero.

Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10).

Use **Laplacian correction** (or Laplacian estimator)

Adding 1 to each case

Prob(income = low) = 1/1003
Prob(income = medium) = 991/1003
Prob(income = high) = 11/1003

Naïve Bayes Classifier: Pros and Cons.

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence
    - loss of accuracy
  - In real life, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier.

Nearest neighbor classification

- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - – To classify a new example by finding the training example (y_i, x_i) that is nearest to x.
  - – Guess the class $y = y_i$

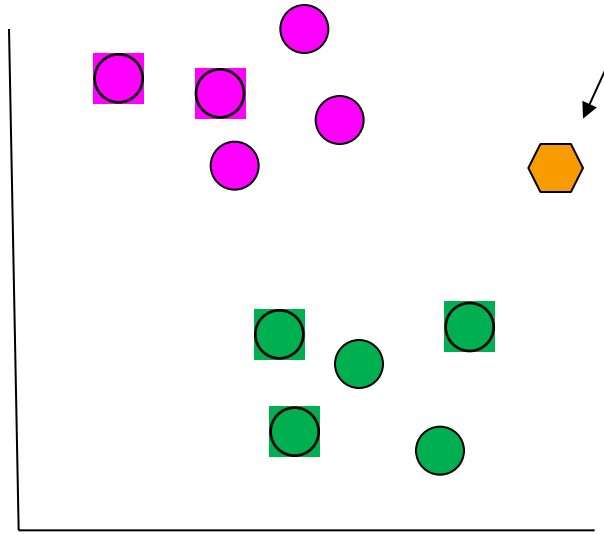
http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture10.pdf

K-Nearest Neighbor (K-NN) Method

- To classify a new input vector x,
  - Examine the K closest training data points to x.
  - Assign the object to the most frequently occurring class.

http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture10.pdf

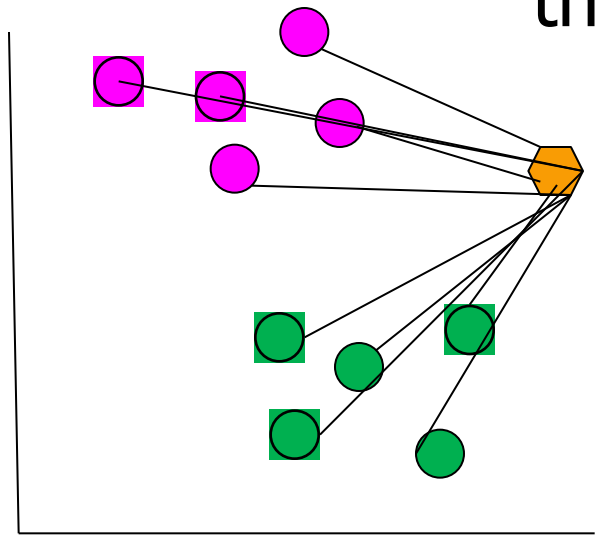
K-NN: example (K=3)



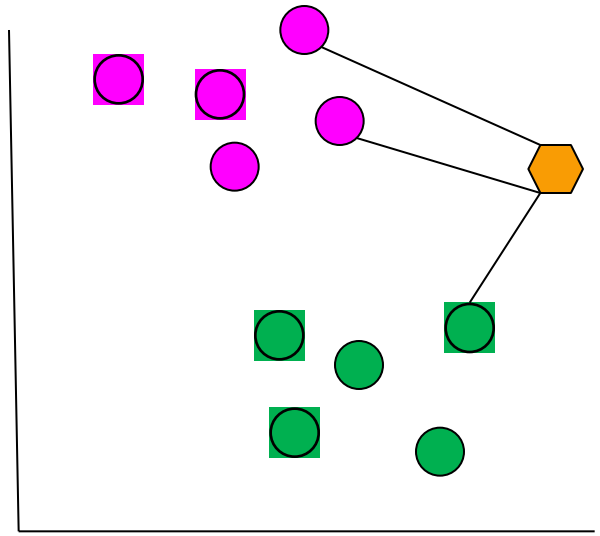
Query pattern for classification.

K-NN: example (K=3)

Compute distances with all the training samples.

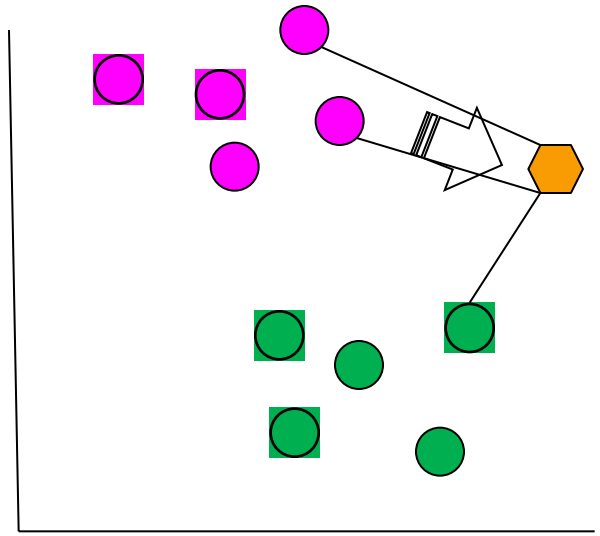


K-NN: example (K=3)



Find K=3 nearest neighbors

K-NN: example (K=3)



Assign the class which has maximum number of NNs.

K-NN: Probabilistic interpretation

- Non-parametric estimation of probability density at x given K neighbors.
- Number of training samples: N
- Assume the volume (hyper-volume) containing K neighbors: V
- Let prob. of a data point in the volume be P.

Following binomial distribution:
E(no. of data points in the volume)=N.P=K

Estimates of the Prob. that the volume V around x contains a data point, P= K/N

K-NN: Probabilistic interpretation

Estimates of the Prob. that the volume V around x contains a data point, P= K/N

Probability density at x, p(x) = P/V

Consider a class w_1 contains n_1 number of neighbors out of K neighbors.

$p(x, w_1) = (n_1/N)/V$

$p(w_1|x)=p(x,w_1)/p(x)=n_1/K$

Posterior prob. of the class w_1 given x.

Assign the class which has maximum posterior prob.

K-NN: Probabilistic interpretation

Estimates of the Prob. that the volume V around x contains a data point, P= K/N

Probability density at x, p(x) = P/V

Consider a class w_1 contains n_1 number of neighbors our of K neighbors.

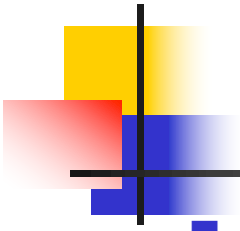
$$p(x, w_1) = (n_1/N)/V$$

$$p(w_1|x)=p(x,w_1)/p(x)=n_1/K$$

Posterior prob. of the class w_1 given x.

Assign the class which has maximum posterior prob.

Bayesian inferencing



Nearest neighbor

- When to consider
  - Instance map to points in R^n
  - Less than 20 attributes per instance.
  - Lots of training data.
- Advantages
  - Fast training.
  - Learn complex target functions.
  - Do not lose information
- Disadvantages
  - Slow at query time.
  - Easily fooled by irrelevant attributes.

http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture10.pdf

Issues

- Distance measure
  - Most common Euclidean
- Choosing K
  - Increasing k reduces variance, increases bias (class prior dominates!).
- For high dimensional space, the nearest neighbor may not be close at all!
- Memory based technique.
  - Must make a pass through data points for each classification.
  - prohibitive for a large data set.

Classification by discriminant function

- Consider two class problems.
- Input: Labelled data sets: $\{(y_i,x_i)\}$, i=1,2,…,n
  - x_i in R^n and y_i= +1 or -1.
- Design a discriminant function g(x) for class assignment as y_i=sign(g(x_i)).
  - Decision boundary: g(x)=0
  - Linear Discriminat Analysis (LDA): g in linear form.
    - Polynomial of x of degree 1.
  - Quadratic Disciminant Analysis (QDA): g being quadratic.
    - Polynomial of x of degree 2.

Linear discriminant analysis (LDA) from Bayes' classifier

- Bayes' classification steps:
  - Given an input x
    - Estimate $P(y_i|x)$, for all i.
    - Assign y_k to x if it is maximum.
  - Difficult to estimate $P(y_i|x)$ directly.
  - Compute the class likelihood ($P(x|y_i)$) and class prior $P(y_i)$ from the data.
    - $P(y_i|X)=(P(x|y_i).P(y_i) / P(x))$
    - Not required to compute P(x), as it is the same for all classes.

Linear discriminant analysis (LDA) from Bayes' classifier

- Assume likelihood distributions are normal.

$$N(x|m_k, S) = \frac{1}{\sqrt{(2\pi)^n |S|}} e^{-\frac{1}{2}(x-m_k)^T S^{-1}(x-m_k)}$$

C (a constant): Independent of class.

  - $N(x; m_k, S_k)$, k=1,2
  - Assume covariance matrices are the same
    - $S_k=S$, for all k.
  - log(Prob. of class k):

Assign class k having maximum value of log(.).

$$\log(p_k) + \log(C) - \frac{1}{2}(x-m_k)^T S^{-1}(x-m_k)$$

Ignore while comparing.

$$\Rightarrow \log(p_k) - \frac{1}{2}(x-m_k)^T S^{-1}(x-m_k)$$

Linear form of function

Maximize $\log(p_k) - \frac{1}{2}(x - m_k)^T S^{-1}(x - m_k)$

$$\Rightarrow \log(p_k) - \frac{1}{2}\left(x^T S^{-1} x - x^T S^{-1} m_k - m_k^T S^{-1} x + m_k^T S^{-1} m_k\right)$$

Independent of class.

$$\Rightarrow \log(p_k) - \frac{1}{2}\left(-x^T S^{-1} m_k - m_k^T S^{-1} x + m_k^T S^{-1} m_k\right)$$

As S is symmetric. $\Rightarrow$

$$\log(p_k) - \frac{1}{2}\left(m_k^T S^{-1} m_k\right) + m_k^T S^{-1} x$$

$g_k(x)$

Linear discriminant function: $g(x)=g_1(x)-g_2(x)$

LDA as Bayesian Classification

- Estimate class priors p_k. $p_k = \frac{N_k}{N}$
- Given training data estimate the means (m_k's) of classes and the covariance matrix (S) of the data.

$$m_k = \frac{1}{N_k} \sum_{x \in k} x$$

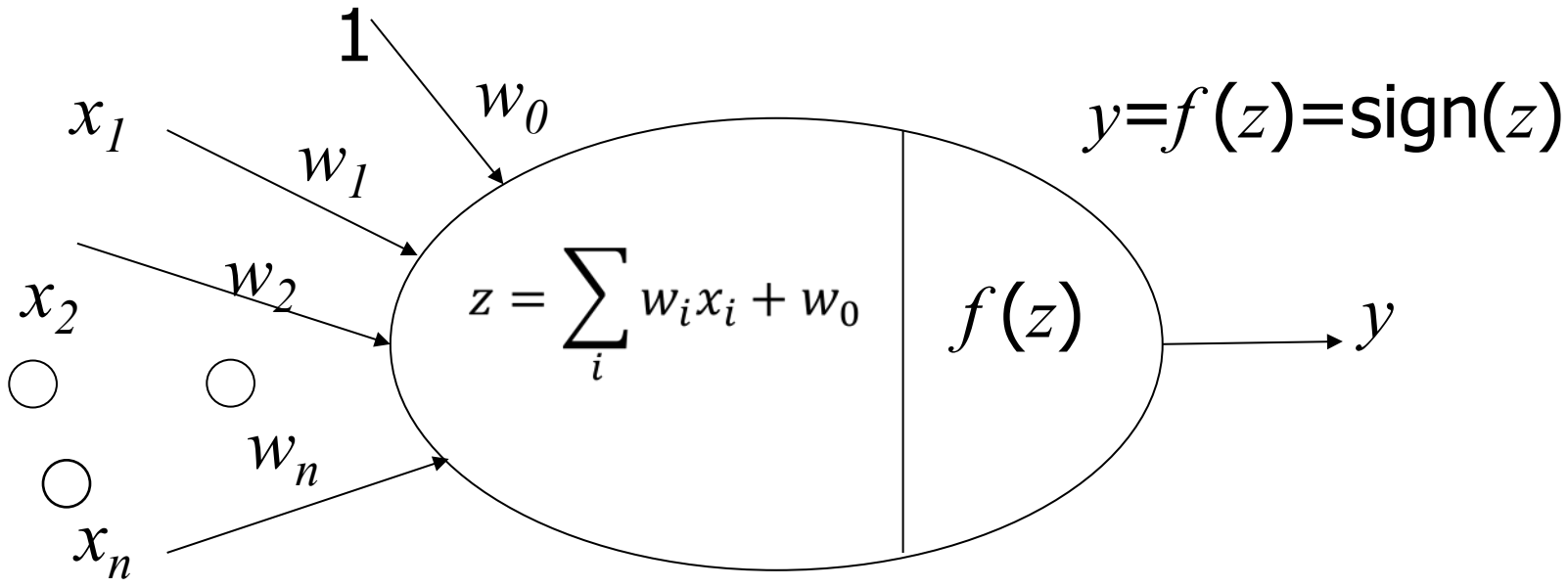
$$S = \frac{1}{N-1} \sum_{\forall x} (x - m)(x - m)^T$$

Mean of data.

Obtain the discriminant function g(x) and use it for classification.

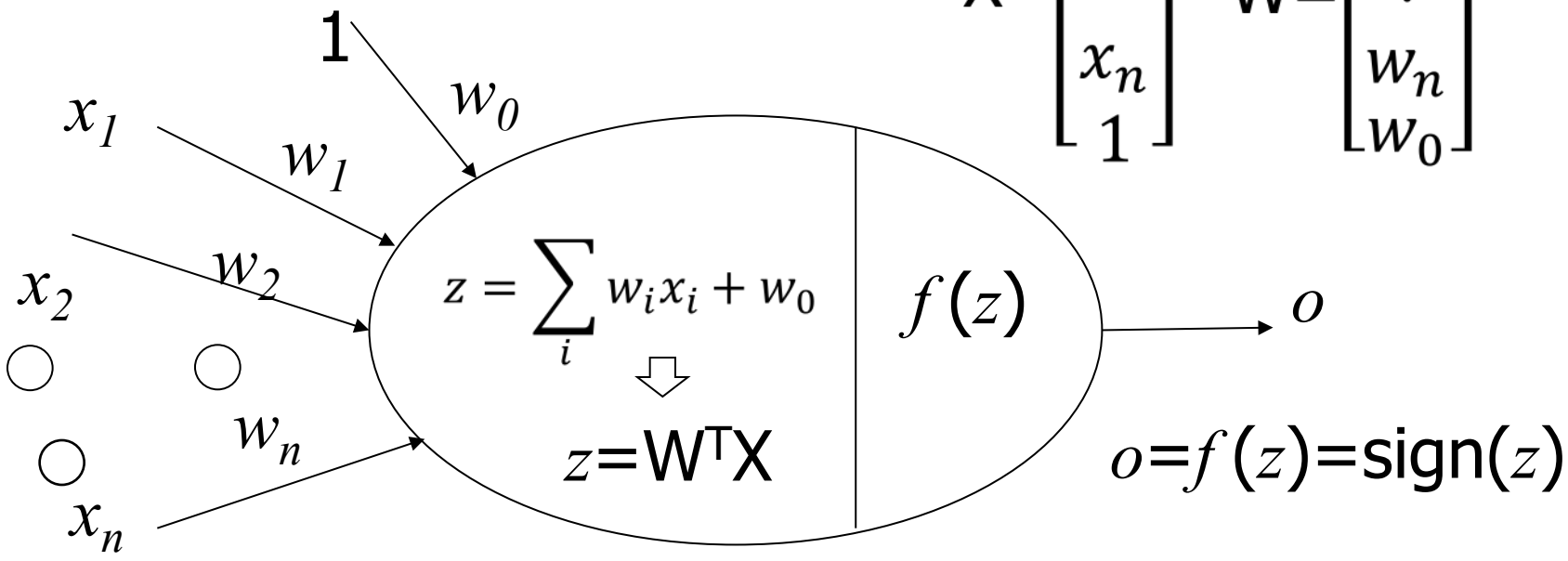
Perceptron classifier

- A linear classifier with a different perspective.



Augmented input and linear form

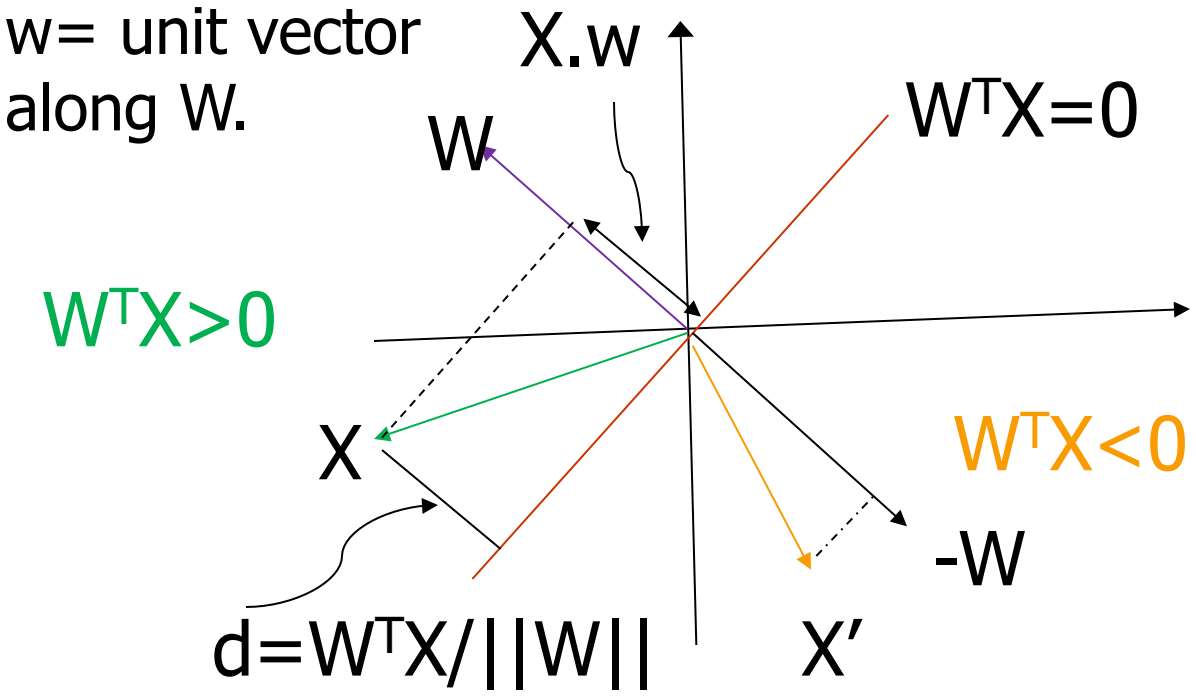
$$X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \quad W=\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_0 \end{bmatrix}$$



Given $\{y_i, X_i\}$, compute optimum W minimizing classification error.

Interpretation of W^TX

Consider the hyperplane $W^TX=0$ separating two samples X and X' of classes 1 and 2.

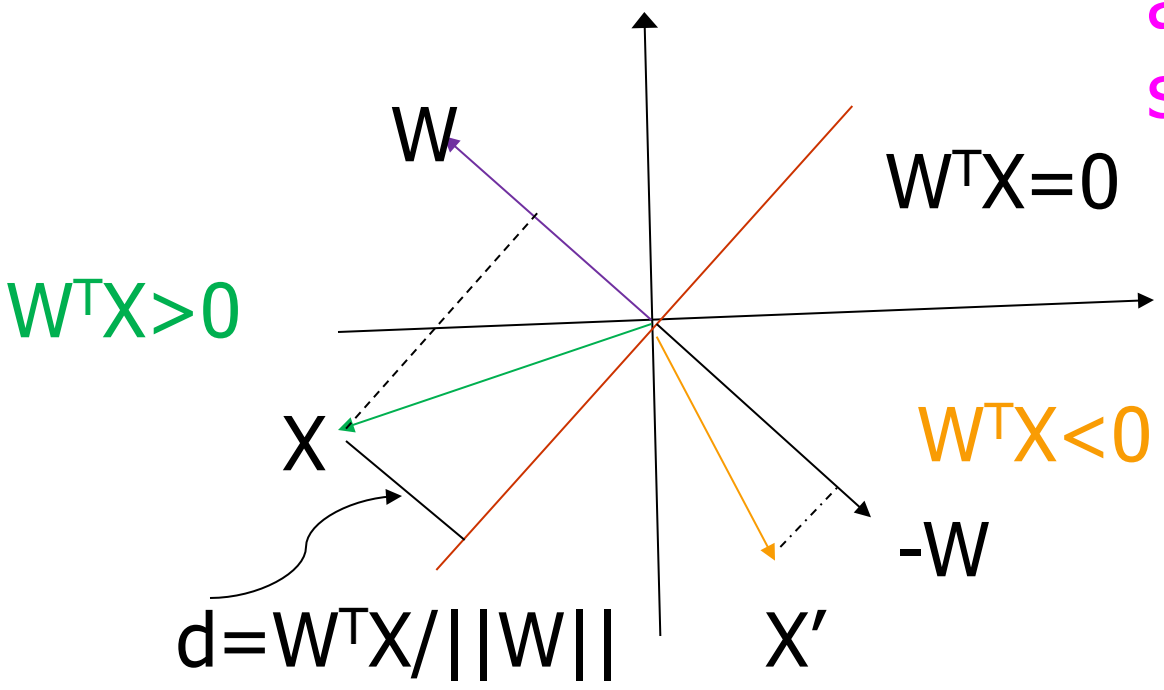


Distance of X from the hyperplane.

Linearly separable classes

To find a hyperplane separating data points of two classes.

If a solution exists, the classes are called linearly separable.



Distance of X from the hyperplane.

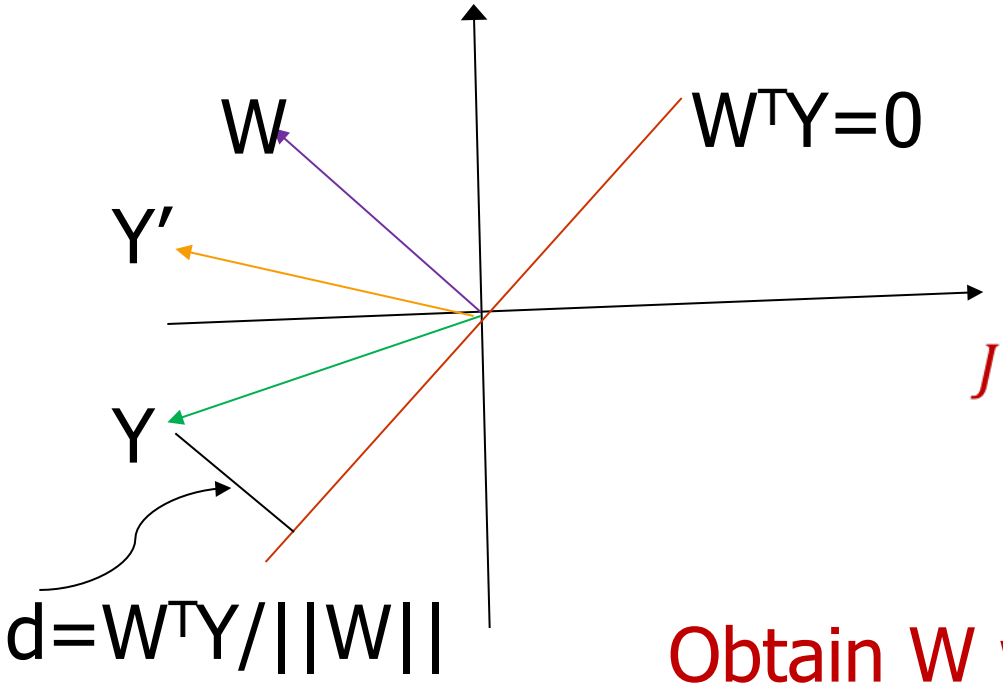
An error function

Data Normalization:

Y=X, if X in class 1 (o=1).

=-X, if X in class 2 (o=-1)

⇨ For correct classification, $W^TY > 0$, for all Y.



Error function (Perceptron Criterion):

$$J(W) = \sum_{Y\ misclassified} -W^TY$$

Always +ve

Obtain W which minimizes J(W).

Gradient descent method for iterative optimization

- To obtain W which minimizes J(W).
- Start with an initial vector $W^{(0)}$.
- Compute the gradient vector $\nabla J(W^{(0)})$
- Move closer to minimum by updating W.

$$W^{(i)} = W^{(i-1)} - \eta(i)\nabla J(W)$$

Positive scale factor (learning rate)

Iterative gradient descent Optimization

Data Normalization:

Y=X, if X in class 1 (o=1).

=-X, if X in class 2 (o=-1)

$$J_p(W) = \sum_{Y\ misclassified.} -W^T Y$$

Iterative Optimization using gradient descent

1. Start with $W^{(0)}$.
2. Update W

$$W^{(i)} = W^{(i-1)} + \eta(i) \sum_{Y\ misclassified} Y$$

$$W^{(i)} = W^{(i-1)} - \eta(i)\nabla J_p(W)$$

May be taken as a constant.

3. Continue step 2 till converges.

Other forms of the error function

- There could be other forms of the criterion function.
  - $J_p(W)$: not continuous
  - $J_q(W)$: continuous.
    - Very smooth in boundary.
    - May get stuck there.
    - Value dominated by long Y's.

$$J_q(W) = \sum_{Y\ misclassified.} (W^T Y)^2$$

Another error function (Relaxation criterion)

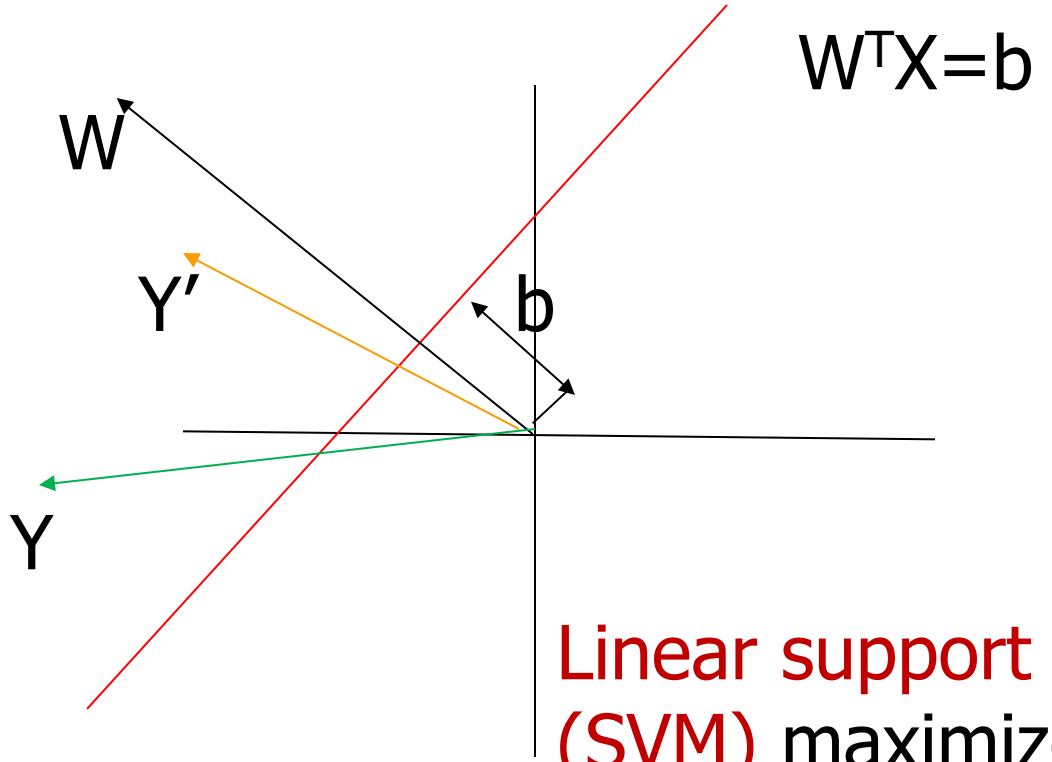
$$J_r(W) = \frac{1}{2} \sum_{\substack{Y\ \text{misclassified} \\ W^T Y \le b}} \frac{(W^T Y - b)^2}{\|Y\|^2}$$

Stronger linear separability

Gradient computation:

$$\nabla J_r(W) = \sum_{W^T Y \le b} \frac{Y(W^T Y - b)}{\|Y\|^2}$$

More stringent criteria of linear separability



Linear support vector machines (SVM) maximize b between two linearly separable data points of classes.

The algorithm (Batch relaxation with margin)

- Initialize W to $W^{(0)}$.
- Iterate till convergence
- Compute the set M of misclassified samples (with margin b), so that
  - $M=\{Y|W^TY<=b\}$
- Compute gradient.
- Update W.

$$\nabla J_r(W) = \sum_{W^TY \le b} \frac{Y(W^TY - b)}{\|Y\|^2}$$

$$W^{(i)} = W^{(i-1)} - \eta(i)\nabla J_r(W^{(i-1)})$$

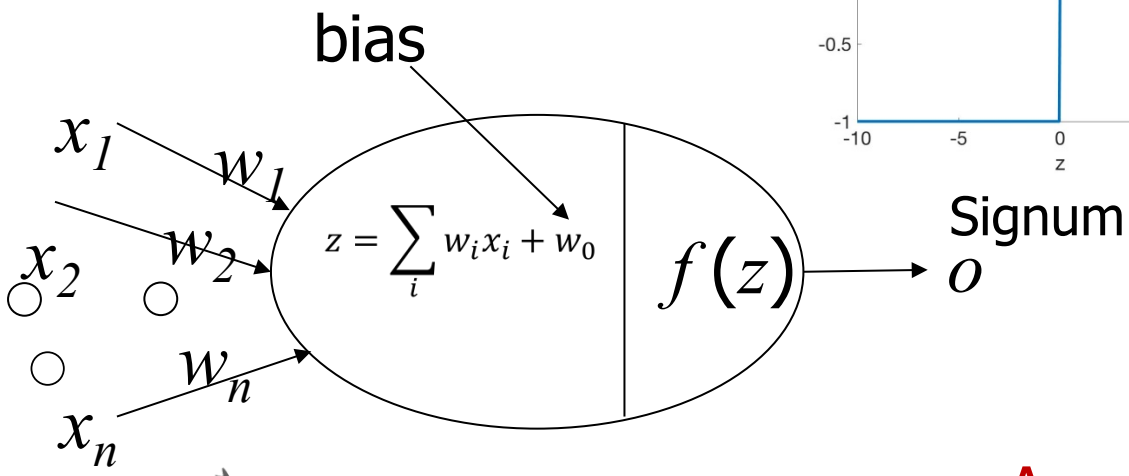
Single sample relaxation with margin

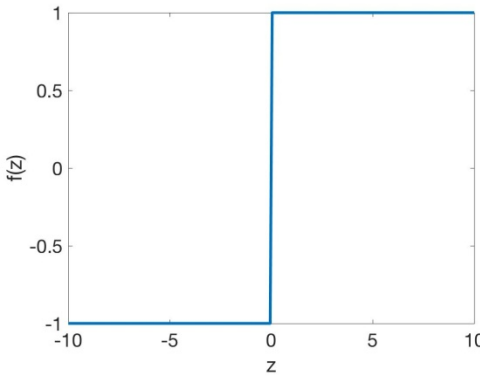
- Initialize W to $W^{(0)}$.
- Perform the update of W by considering samples one by one in every iteration.
- Consider an i th sample Y_i at k th iteration.
- If ($W^T Y_i$ <=b)
  - Update W.

$$W^{(k)} = W^{(k-1)} + \eta(k) \frac{b - W^T Y_i}{\|Y_i\|^2} Y_i$$

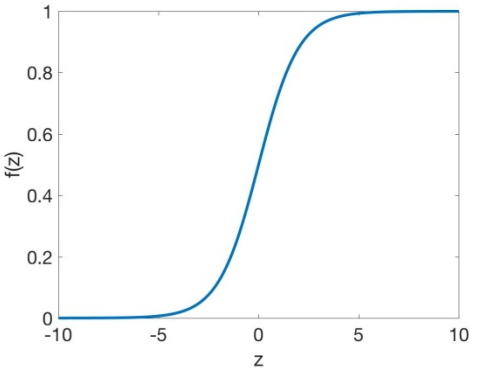
- Stop when very little change in updates at the end of an iteration.

Perceptron modelling a neuron



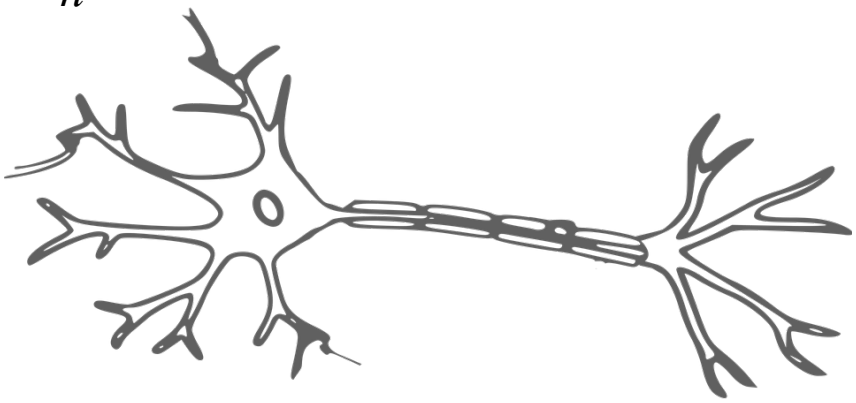


Signum



Logistic / Sigmoid

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

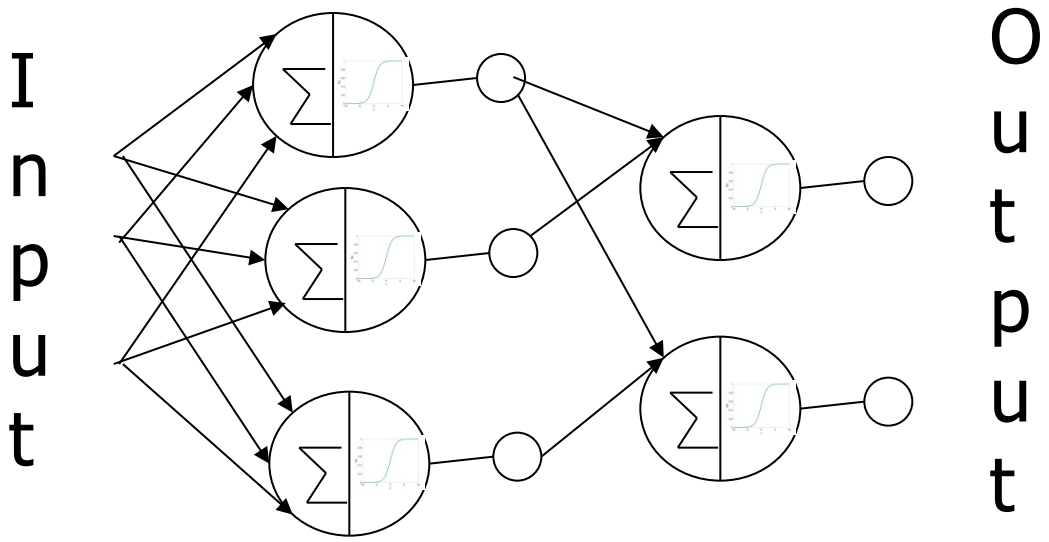


A network of perceptrons provides a powerful model describing input / output relations.

https://pixabay.com/vectors/brain-neuron-nerves-cell-science-2022398/

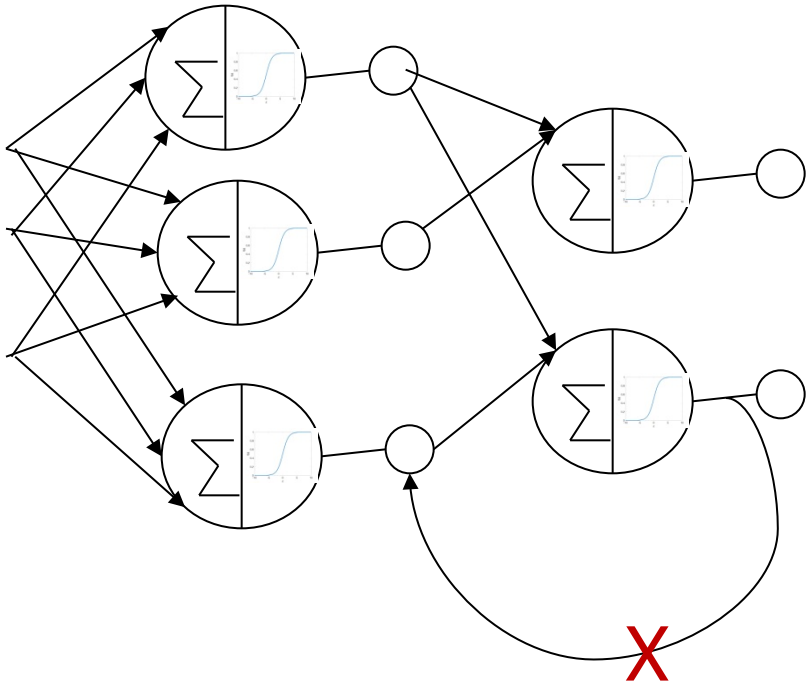
Artificial Neural Network

- A network of perceptrons.
  - Input: A vector
  - Output: A vector / A scalar

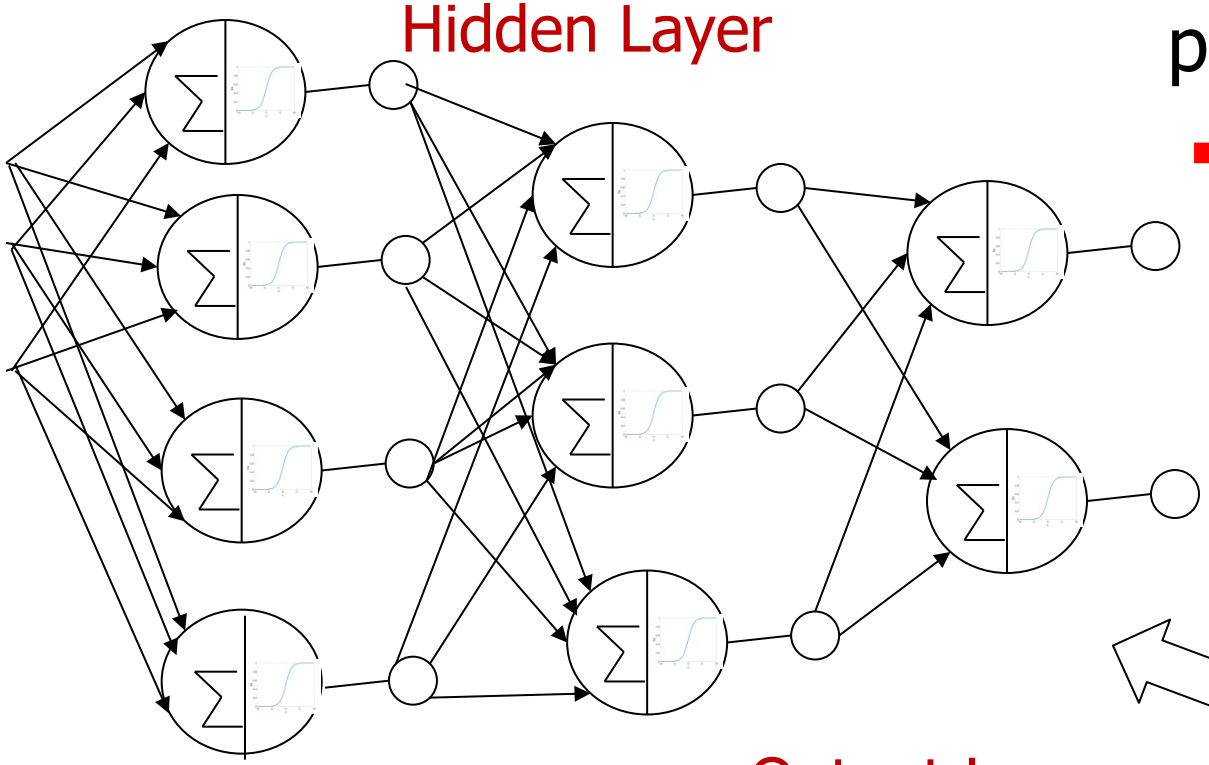


Feed-forward Network

- No feed back or loop in the network.



Multilayered feed-forward Network



- Layer-wise processing
  - i th layer takes input from (i-1)th layer and forwards its output to the input of next layer.

Fully connected (FC) feed-forward network.

Input Layer
- Accepts input

Output Layer
- Generates output.

Mathematical description of the model

- Let j th neuron of i th layer be $ne_j^{(i)}$.
- Its corresponding weights
  - $W_j^{(i)} = (w_{j1}^{(i)}, w_{j2}^{(i)}, \ldots, w_{jn\_(i-1)}^{(i)})$
  - Bias: $w_{j0}^{(i)}$
  - $n\_(i-1)$: Dimension of input to the neuron
  - n_i: Dimension of output at i th layer
- Output of the neuron:

$$y_j^{(i)} = f\left(W_j^{(i)^T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

Mathematical description of the model

- Output of j th neuron in i th layer:

$$y_j^{(i)} = f\left(W_j^{(i)^T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

- Input output relation in i th layer

$$Z^{(i)} = \begin{bmatrix} W_1^{(i)^T} \\ W_2^{(i)^T} \\ \vdots \\ W_{n\_i}^{(i)^T} \end{bmatrix} X^{(i-1)} + \begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n\_i0}^{(i)} \end{bmatrix}$$

$\mathbf{W^{(i)}}$ $\mathbf{b^{(i)}}$

Input output relation

$\mathbf{W^{(i)}}$ $\mathbf{b^{(i)}}$

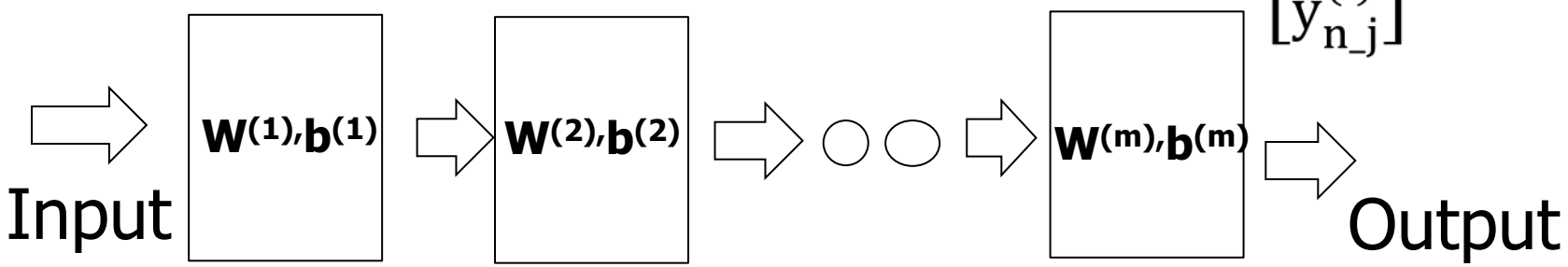
- Output of j th neuron in i th layer:

$$y_j^{(i)} = f\left(W_j^{(i)^T} X^{(i-1)} + w_{j0}^{(i)}\right)$$

$$Z^{(i)} = \begin{bmatrix} W_1^{(i)^T} \\ W_2^{(i)^T} \\ \vdots \\ W_{n\_j}^{(i)^T} \end{bmatrix} X^{(i-1)} + \begin{bmatrix} w_{10}^{(i)} \\ w_{20}^{(i)} \\ \vdots \\ w_{n\_j0}^{(i)} \end{bmatrix}$$

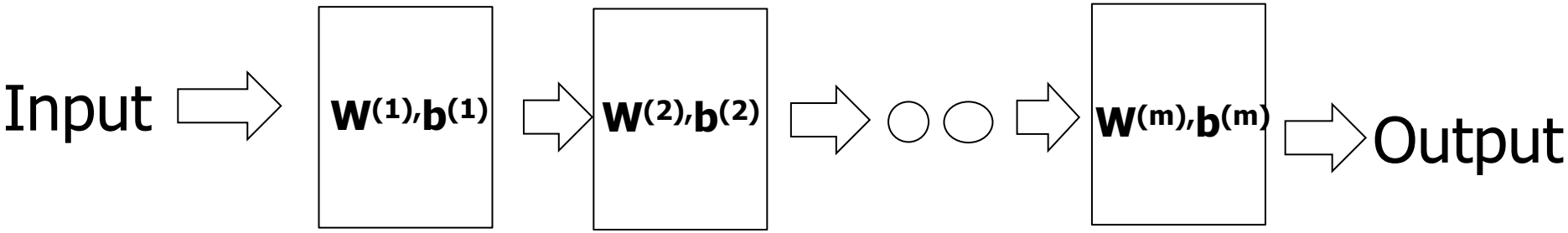
- Input output relation in i th layer

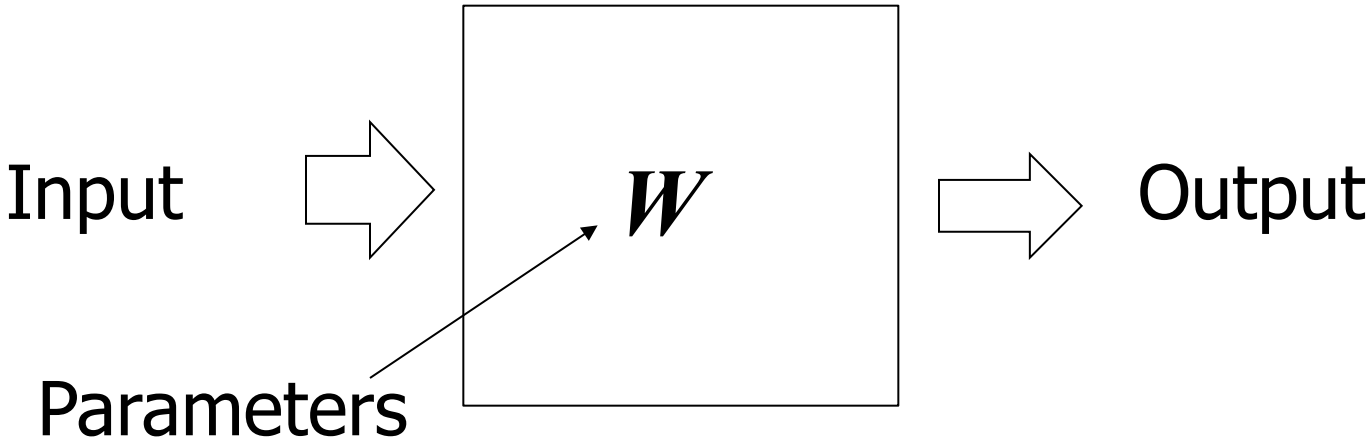
$$Y^{(i-1)} = \mathbf{f}\left(\mathbf{W^{(i)}} X^{(i-1)} + \mathbf{b^{(i)}}\right) \equiv \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_{n\_j}^{(i)} \end{bmatrix}$$



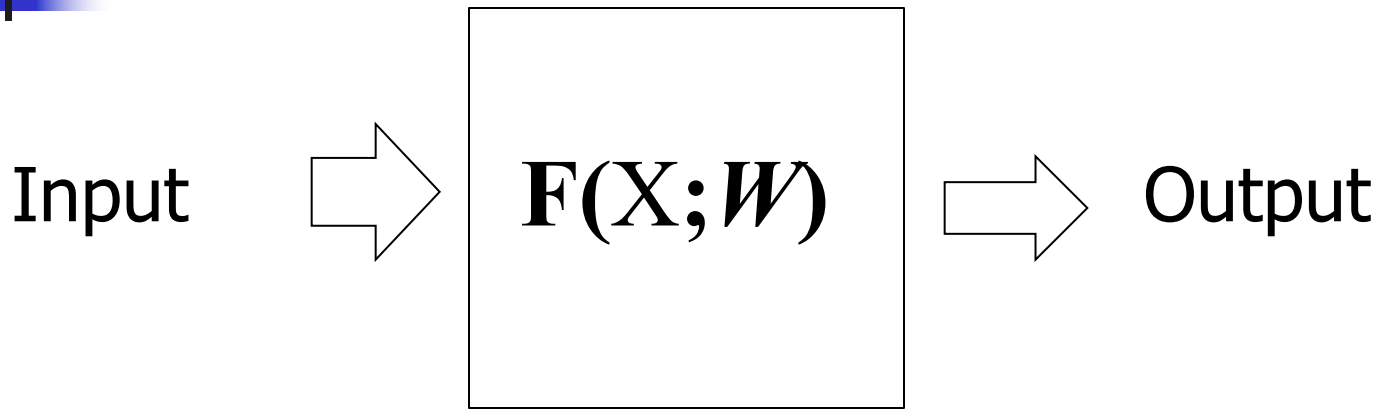
Input output relation

$$Y^{(i-1)} = f\left(W^{(i)}X^{(i-1)} + b^{(i)}\right)$$





Optimization problem

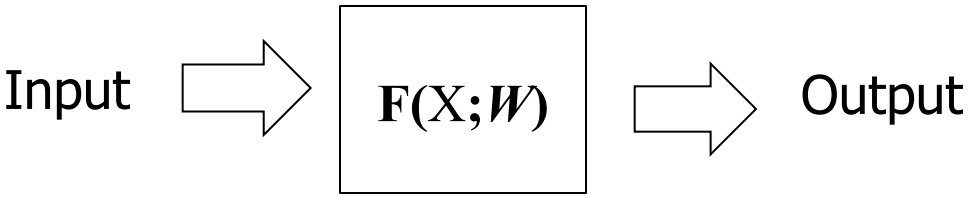


Given $\{(X_i, O_i)\}$, i=1,2,…,N, find $\boldsymbol{W}$ such that it produces O_i given input X_i for all i.

Minimize: $$J_n(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} \|O_i - F(X_i; \boldsymbol{W})\|^2$$

Apply the same gradient descent procedure to obtain the solution.

Optimization problem



Training samples:
$\{(X_i , O_i)\}$, i=1,2,…,N

Minimize: $J_n(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N} \|O_i - F(X_i; \boldsymbol{W})\|^2$

Apply the same gradient descent procedure to obtain the solution.

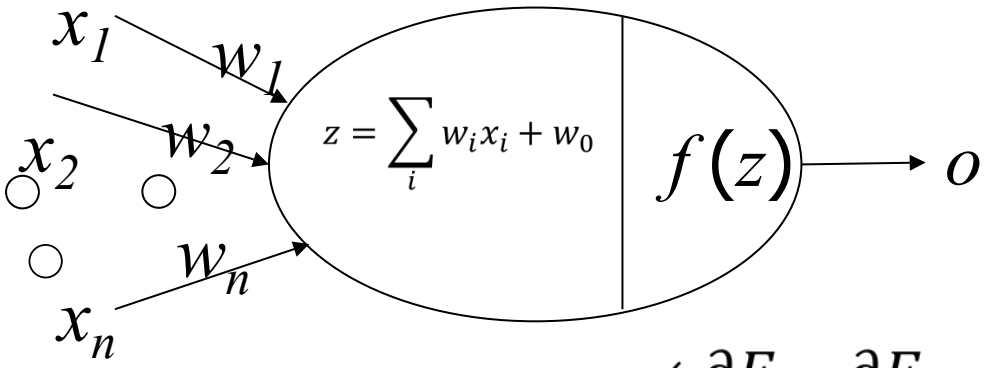
1. Start with an initial $\boldsymbol{W}_0$.
2. Update $\boldsymbol{W}$ iteratively.

$$\boldsymbol{W}_i = \boldsymbol{W}_{i-1} + \eta(i) \sum_k \left(O_k - F(X_k;\ \boldsymbol{W}_{i-1})\right) \nabla F(X_k; \boldsymbol{W}_{i-1})$$

Stochastic gradient descent:

$$\boldsymbol{W}_i = \boldsymbol{W}_{i-1} + \eta(i)(O_k - F(X_k; \boldsymbol{W}_{i-1}))\ \nabla F(X_k; \boldsymbol{W}_{i-1})$$

Chain rule of computing gradient of a single neuron



Target response: t

Error:

$E=(t-o)^2$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial w_i}$$

$-2(t-o)$ $\quad f'(z)$ $\quad x_i$

$$\nabla(W) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right)$$

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

$$\frac{\partial E}{\partial w_i} = -2(t-o)f'(z)x_i$$

$$f'(z) = \frac{e^{-z}}{(1+e^{-z})^2}$$

$$\frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right)$$

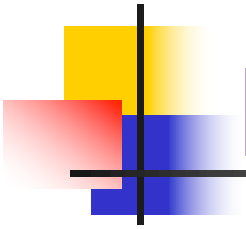
Analytical method!
Computed given the functional values.

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial x_i}$$

$$\frac{\partial E}{\partial x_i} = -2(t-o)f'(z)w_i$$

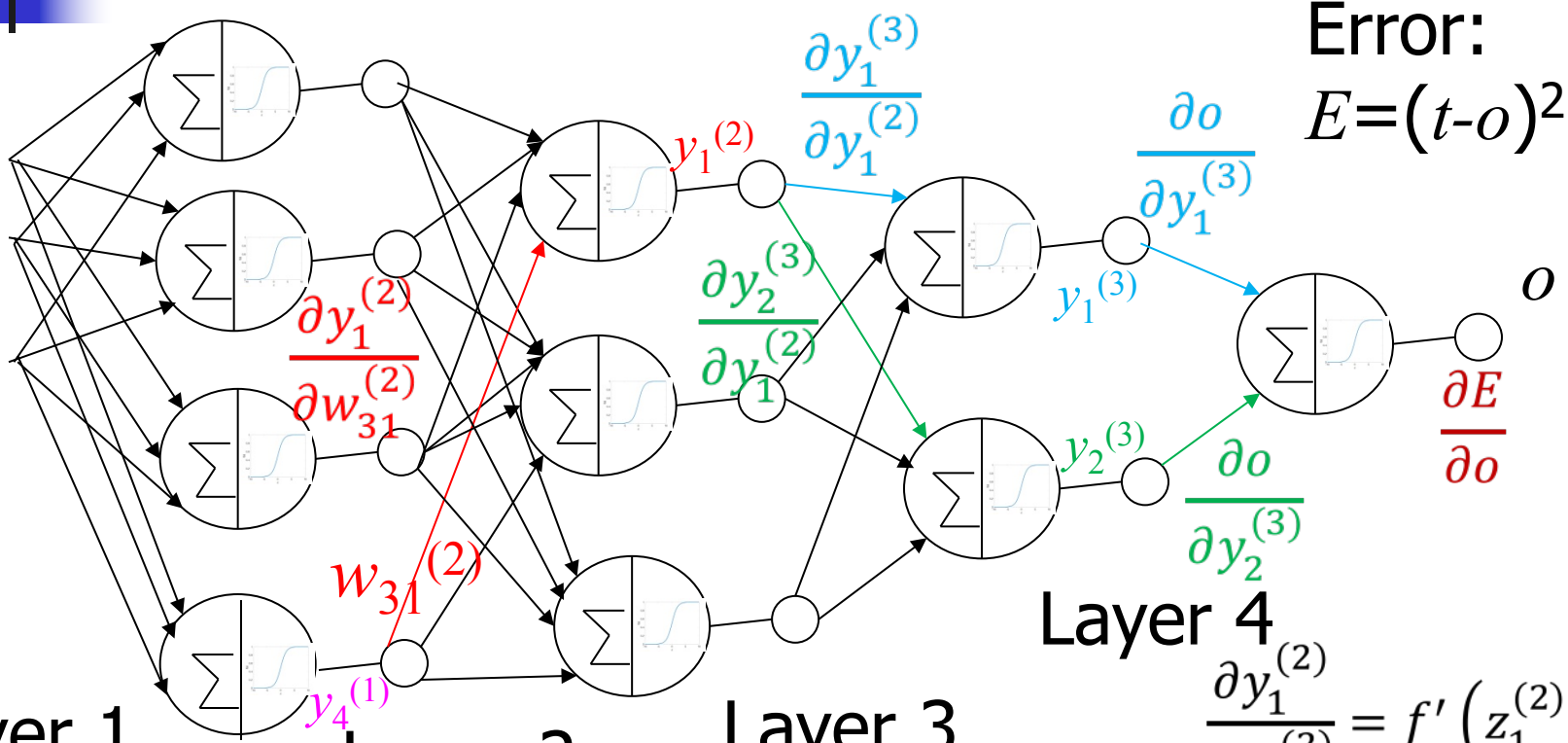
$f(z)(1-f(z))$

Computing gradient: Back propagation method



- For multi-layered feed forward network.
- Apply chain rule.
  - From output to toward input.
  - From output layer to toward input layer.
  - Compute partial derivatives of weights at (i-1)th layer from the i th layer.

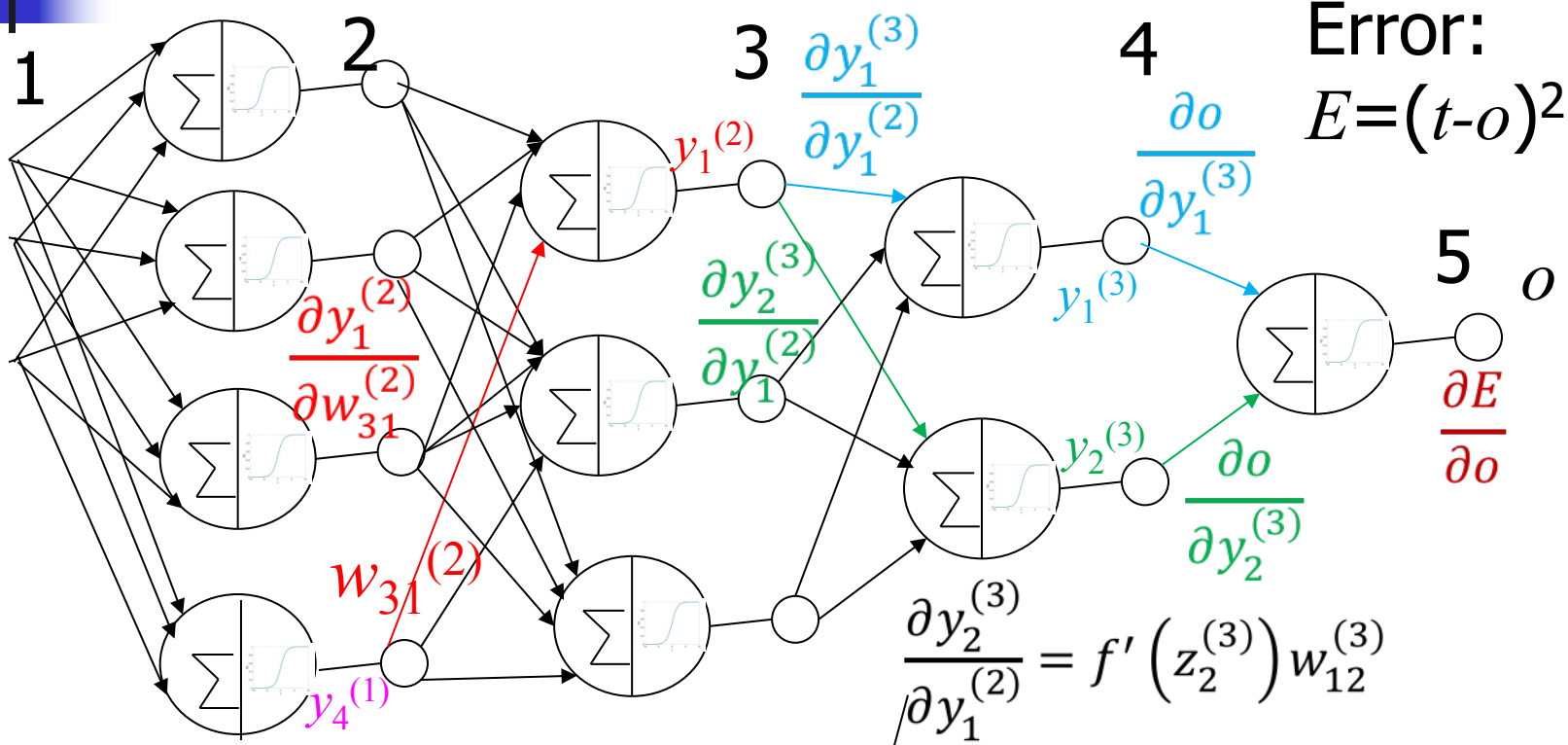
Back propagation: Concept



$$\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}} = f'\left(z_1^{(2)}\right) y_4^{(1)}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

Back propagation: Concept

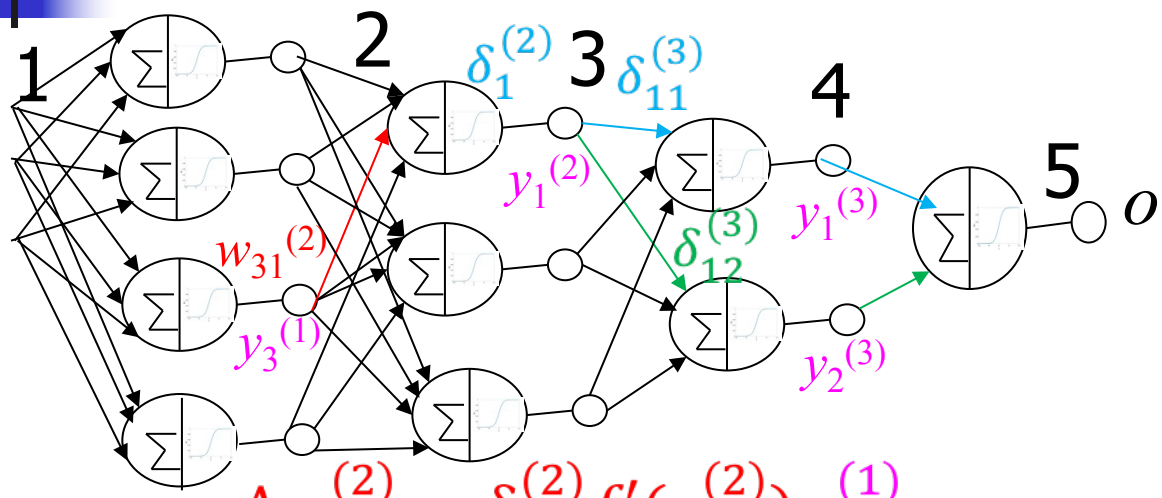


$$\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} = f'\left(z_2^{(3)}\right) w_{12}^{(3)}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

$$\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}} = f'\left(z_1^{(2)}\right) y_4^{(1)}$$

Back propagation: Delta rule



Error:

$E=(t-o)^2$

$$\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}} = f'\left(z_2^{(3)}\right) w_{12}^{(3)}$$

$$\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} = f'(z_1^{(3)}) w_{11}^{(3)}$$

$$\Delta w_{31}^{(2)} = \delta_1^{(2)} f'(z_1^{(2)}) y_3^{(1)}$$

$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\frac{\partial o}{\partial y_1^{(3)}}\frac{\partial y_1^{(3)}}{\partial y_1^{(2)}} + \frac{\partial o}{\partial y_2^{(3)}}\frac{\partial y_2^{(3)}}{\partial y_1^{(2)}}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

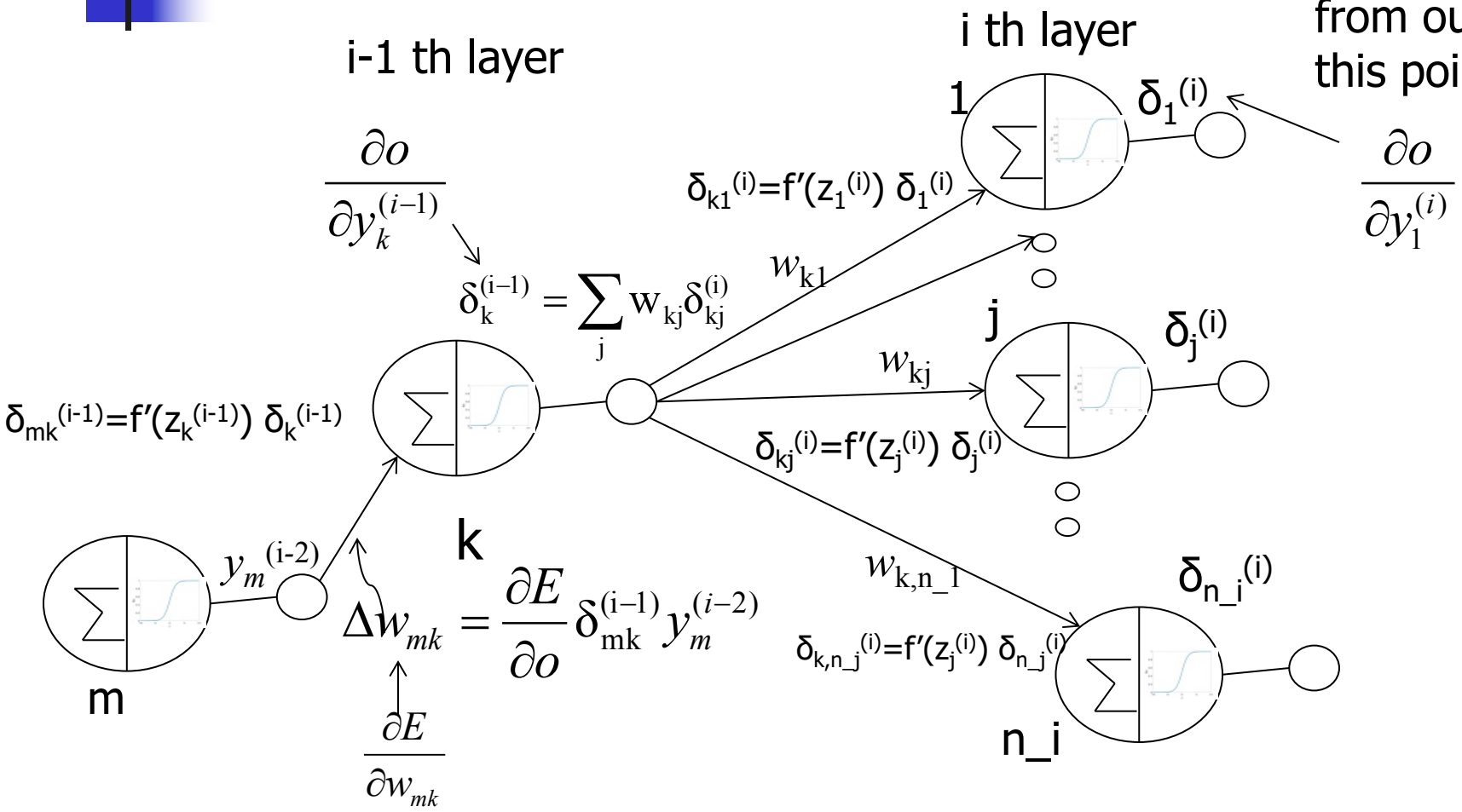
$$\Delta w_{31}^{(2)} = \frac{\partial E}{\partial w_{31}^{(2)}} = \frac{\partial E}{\partial o}\left(\underbrace{\frac{\partial o}{\partial y_1^{(3)}} f'\left(z_1^{(3)}\right)}_{\delta_{11}^{(3)}} w_{11}^{(3)} + \underbrace{\frac{\partial o}{\partial y_2^{(3)}} f'\left(z_2^{(3)}\right)}_{\delta_{12}^{(3)}} w_{12}^{(3)}\right)\frac{\partial y_1^{(2)}}{\partial w_{31}^{(2)}}$$

Delta rule:

$$\delta_1^{(2)} = \delta_{11}^{(3)} w_{11}^{(3)} + \delta_{12}^{(3)} w_{12}^{(3)}$$

Back propagation: Delta rule

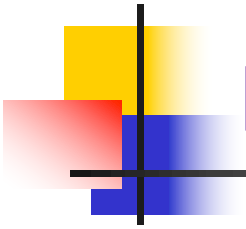
Accumulated gradient value from output to this point.



ANN training

- Initialize $W^{(0)}$.
- For each training sample (x_i, o_i) do
  - Compute functional values of each neuron in the forward pass.
  - Update weights of each link starting from the output layer using back propagation.
  - Continue till it converges.

Classification or regression?



- Primarily a regressor.
  - Build a model to predict functional value F(x) given input x.
- Can be converted to a classifier by appropriate encoding of classes (output vector o).
  - Two class problem
    - Binary encoding: 0 / 1
    - One hot encoding: (1 0) / (0 1)

Evaluation of a classifier

- Two class problems.
  - Positive class and Negative class
  - TP: Set of +ve samples predicted +ve.
  - FP: Set of –ve samples predicted +ve.
  - TN: Set of –ve samples predicted –ve.
  - FN: Set of –ve samples predicted +ve.

	AP	AN
PP	TP	FP
PN	FN	TN

Accuracy:
(TP+TN)/Total

Precision: TP/PP
Recall: TP/AP

Sensitivity / Recall:
TPR= TP/AP
Specificity:
TNR= TN/AN

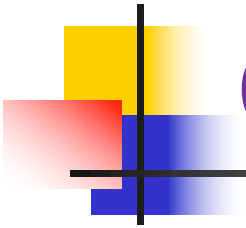
F-Score:
Harmonic mean of precision and recall

$$F = \frac{2}{\frac{1}{Prec} + \frac{1}{Recall}} = \frac{2 \times Prec \times Recall}{Prec + Recall}$$

Evaluation of a classifier

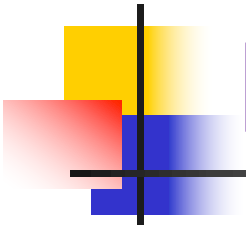
- Multi class problems.
  - Confusion matrix

Accuracy: (Sum of diagonal) / Total



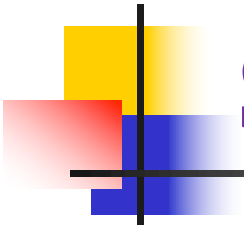
Cross validation

- For supervised classification.
- Separate training and test data.
- Train network using training data.
- Evaluate using test data.



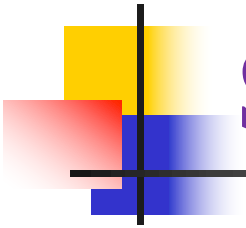
k-fold cross validation

- Divide the data in k sets of equal size.
- Train using (k-1) sets and test with the remaining.
- Do it for every set as a test set.
- Report the average performance.



Summary

- Classification
  - Task of assigning a known category or class to an object.
- Clustering
  - the task of organizing objects into groups whose members are *similar in some way*.
- Clustering techniques
  - K-Means
  - K-Medoids.
  - GMM



Summary

- Classification techniques
  - Naïve Bayesian
  - K-NN
  - Linear discriminant analysis
  - ANN