CONSISTENT

Am hypothesis h is consistent with a set of training dataset D of target concept c if and only if $h(x) = c(x)$ for each training samples $(x, c(x))$ of D

$$\text{consistent}(h, D) \equiv \forall x \in D \; h(x) = c(x)$$

OVERFITTING

hypothesis h OVERFITS if exist another hypothesis h' given a data sample S if

1. $\text{error}_S(h) < \text{error}_S(h')$ and $\text{error}_D(h) > \text{error}_D(h)$

CLASSIFICATION METRICS

$$\text{PRECISION} = \frac{TP}{TP+FP} \qquad \text{RECALL} = \frac{TP}{TP+FN} \qquad \text{F1-SCORE} = \frac{2(\text{PRECISION} \cdot \text{RECALL})}{\text{PRECISION} + \text{RECALL}}$$

$$\text{ACCURACY} = \frac{TP+TN}{TP+FP+FN+TN}$$

REGRESSION METRICS

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\hat{f}(x_i) - t_i| \qquad MSE = \frac{1}{N}\sum_{i=1}^{N} (\hat{f}(x_i) - t_i)^2 \qquad RMSE = \sqrt{MSE}$$

ENTROPY → $\text{entropy}(S) = -\sum_{i=1}^{N} P_i \log_2(P_i)$

GAIN → $\text{entropy}(S) - \sum_{v \in \text{values}(A)} (|S_v|/|S|)\,\text{entropy}(S_v)$

REDUCE OVERFITTING (decision tree) →

1. REDUCED ERROR PRUNING: using validation dataset if a node doesn't reduce validation can prune

2. RULE POST PRUNE (C4.5): if accuracy is greater expect when we have prune we have overfitting

GAIN RATION → GAIN / SPLIT INFORMATION

SPLIT INFORMATION → $-\sum_{i=1}^{N} |S_i|/|S| \log_2 (|S_i|/|S|)$

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

$P(h|D)$: POSTERIOR; $P(D|h)$: LIKELIHOOD; $P(h)$: PRIOR

MAXIMUM A POSTERIORI h_{MAP} → $h_{map} = \text{argmax}(P(h|D)) =$

$= \text{argmax}((P(D|h)P(h))/P(D)) = \text{argmax}(P(D|h)P(h))$

MAXIMUM LIKELIHOOD h_{ML} → if we assume $P(h_i) = P(h_j)$ →

→ $h_{ML} = \text{argmax}(P(D|h))$

EXAMPLE useful

$h_1(x) = +$ $h_2(x) = -$ $h_3(x) = -$ $P(D|h_1) = 0.4$ $P(D|h_2) = 0.3$ $P(D|h_3) = 0.3$

$h_{ML} = \text{argmax}\{0.4, 0.3, 0.3\} = 0.4 \rightarrow$ class +

$P(+|x, h_1) = 1$ $P(+|x, h_2) = 0$ $P(+|x, h_3) = 0$

$P(-|x, h_1) = 0$ $P(-|x, h_2) = 1$ $P(-|x, h_3) = 1$

$V_{OC} = \text{argmax}\{(0.4 \cdot 1 + 0 + 0), (0 + 0.3 \cdot 1 + 0.3 \cdot 1)\} =$ class −

BAYESIAN OPTIMAL CLASSIFIER → $V_{OB} = \text{argmax} \sum_{v_j \in V} \sum_{h_j \in H} P(v_j|x, h_j) P(h_i|D)$

NAIVE BAYESIAN CLASSIFIER

① x CONDITIONALLY INDEPENDENT OF Y GIVEN z

$$P(x,y|z) = P(x|y,z)P(y|z) = P(x|z)P(y|z)$$

assume a target function $f: X \to V$ where each instance of x can be described $\langle a_1, \dots, a_n \rangle$

$$\underset{v_j \in V}{\operatorname{argmax}} P(v_j|x,D) = \underset{v_j \in V}{\operatorname{argmax}} P(v_j|a_1 \dots a_n, D)$$

② NAIVE BAYES ASSUMPTION

$$P(a_1, \dots, a_n | v_j, D) = \prod_{i=1}^{N} P(a_i|v_i, D)$$

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j|D) \prod_{i=1}^{N} P(a_i|v_i, D)$$

MULTINOMIAL NAIVE BAYES DISTRIBUTION

for ordinal feature vector $d = \langle a_1, \dots, a_n \rangle$ of generic $doc \in Docs$

$$P(d|c_j, D) = \frac{n!}{a_1! \cdots a_n!} \prod_{i=1}^{N} P(w_i|c_i, D)^{a_i}$$

maximum likelihood solution

$$\hat{P}(w_i|c_i, D) = \frac{\sum_{doc \in D} tf_{ij} + \alpha}{\sum_{doc \in D} tf_j + \alpha|V|}$$

GENERATIVE MODELS

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x)} = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} = \frac{1}{1 - \exp(a)}$$

$$a = \ln\left(\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}\right)$$

consider now parametric model:

$$a = \ln\left(\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}\right) = \ln\left(\frac{N(x;\mu_1;\Sigma)}{N(x;\mu_2;\Sigma)}\right) = \dots = \omega^T x + \omega_0$$

$$\omega_0 = \Sigma^{-1}(\mu_1 - \mu_2) \quad \text{and} \quad \omega = -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\left(\frac{P(C_1)}{P(C_2)}\right)$$

LOGISTIC REGRESSION

given a dataset $D = \{(x_n, t_n)\}_{n=1}^N$

LIKELIHOOD FUNCTION: $P(t|\tilde{\omega}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{(1-t_n)}$

CROSS ENTROPY ERROR FUNCTION (negative log LIKELIHOOD):

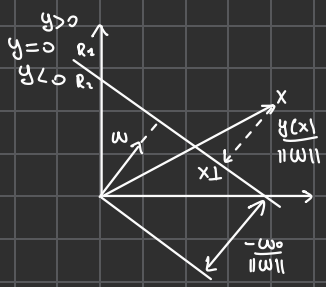
$$E(\tilde{\omega}) = -\log(P(t|\tilde{\omega})) = -\sum_{n=1}^{N} t_n \ln(y_n) + (1-t_n)\ln(1-y_n)$$

$$\tilde{\omega}^* = \text{argmin}(E(\tilde{\omega}))$$

LINEARLY SEPARABLE DATASET

a dataset is linearly separable if exist a surface that splits data in two regions such that different classified instances are separated

LINEAR DISCRIMINANT FUNCTIONS



LEAST SQUARES

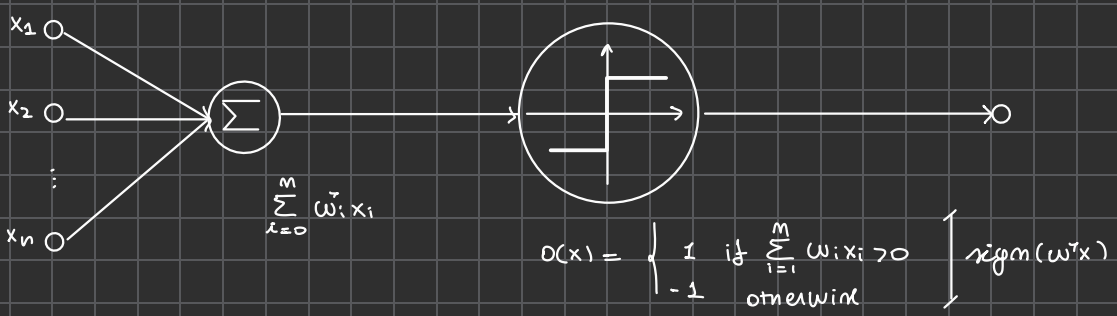
given a dataset $D = \{(x_n, t_n)\}_{n=1}^N$ find LINEAR DISCRIMINANT $y(x) = \tilde{\omega}^T\tilde{x}$

minimize the SUM OF ERROR FUNCTIONS:

$$E(\tilde{\omega}) = \frac{1}{2} tr\{(\tilde{X}\tilde{\omega} - T)^T(\tilde{X}\tilde{\omega} - T)\}$$

with $\tilde{\omega} = \underbrace{(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T}_{\text{PSEUDOINVERSE}}$

PERCEPTRON



on error function we use SQUARED ERROR

$$E(\tilde{\omega}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - o_n) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T x_n)^2$$

$$\frac{\partial E}{\partial \omega} = \sum_{n=1}^{N}(t_n - \omega^T x_n)(-x_{i,n})$$

PERCEPTRONE TRAIN RULE

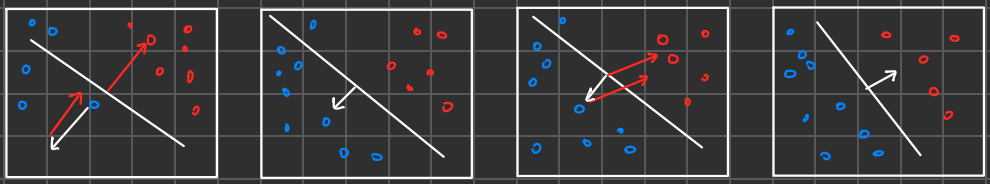
$\omega_i \leftarrow \omega_i + \Delta\omega_i$ where $\Delta\omega_i = -\eta \frac{\partial E}{\partial \omega} = \eta \sum_{n=1}^{N}(t_n - \omega^T x_n)(x_{i,n})$

BATCH MODE → $\Delta\omega_i = \eta \sum_{(x,t)\in D}(t - o(x_i))x_i$ high computational cost

MINIBATCH MODE → $\Delta\omega_i = \eta \sum_{(x,t)\in S}(t - o(x_i))x_i$ BEST ONE

INCREMENTAL MODE → $\Delta\omega_i = \eta (t - o(x_i))x_i$ high variance

PERCEPTRON TRAIN RULE



FISHER DISCRIMINANT RULE

consider two classes can determine $y = w^T x$ and classify $x \in C_1$ if $y \geq -w_0$
$x \in C_2$ otherwise
correspond to the projection on a line determine by w
we want adjust w in order to find direction that maximize separation

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

choose w that maximize $J(w) = w^T(m_2 - m_1)$ subject to $\|w\| = 1$

FISHER CRITERION:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

with $S_B = (m_2 - m_1)^T (m_2 - m_1)$ $\quad S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$

we need to choose w that maximize $J(w)$ so

$$\frac{d J(w)}{dw} = 0 \Rightarrow w^* \propto S_W^{-1} (m_2 - m_1)$$

MULTIPLE CLASSES $J(w) = \text{Tr}\{(w S_W w^T)^{-1} (w S_B w^T)\}$

SUPPORT VECTOR MACHINE SVM

SVM for classification aims at maximum margin providing for better accuracy

MARGIN (smallest distance from x_k to h) is defined $|y(x_k)| / \|w\|$

$\arg\max [\min (|y(x_k)| / \|w\|)] = \arg\max [1/\|w\| \min |y(x_k)|] =$

$= \arg\max \left[\frac{1}{\|w\|} \min \left(t_n (w^T x + w_0) \right) \right]$ assuming $|y(x_k)| = t_n y(x_k)$

so we want find

$$w^*, w_0^* = \arg\max \frac{1}{\|w\|} \min \left(t_n (w^T x + w_0) \right)$$

for solve this optimization problems we can use LAGRAGIAN MULTIPLIERS and for better estimation averaging over all support vector SV

$$SV = \{ x_k \in D \mid t_k (y(x_k)) = 1 \}$$

final we have:

$$w_0^* = \frac{1}{|SV|} \sum_{x_k \in SV} \left(t_k - \sum_{x_j \in S} a_j^* t_j x_k^T x_j \right)$$

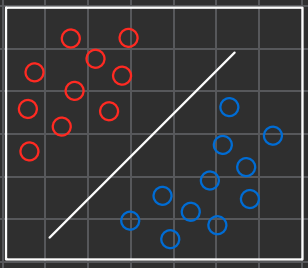
$$y(x, w) = \text{sign} \left(\sum_{x_k \in SV} a_k^* t_k x^{iT} x_k + w_0^* \right)$$

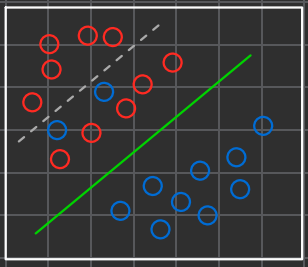
SOFT MARGINS CONSTRAINTS

$\xi_n = 0$ if point is ON OR INSIDE the correct margin boundary
$0 < \xi_n \leq 1$ if point is INSIDE the margin but correct side
$\xi_n > 1$ if point on wrong side of boundary

SVM with SOFT MARGIN CONSTRAINTS

$$w^*, w_0^* = \arg\max \frac{1}{\|w\|^2} + C \sum_{n=1}^{N} \xi_n$$





LINEAR REGRESSION

$$y(x,w) = \sum_{j=0}^{M} w_j \phi_j(x) \quad \text{is linear in } M$$

target value is effected by additive noise $\varepsilon \rightarrow t = y(x,w) + \varepsilon$

assuming gaussian noise $p(\varepsilon|\beta) = \mathcal{N}(\varepsilon|0, \beta^{-1})$ so we have

$$p(t|x,w,\beta) = \mathcal{N}(t|y(x,w),\beta^{-1})$$

we seek maximum likelihood function:

$$p(t_1,\dots,t_n|x_1,\dots,x_n,\beta) = -\frac{\beta}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2 - \frac{N}{2}\ln(2\pi\beta^{-1})$$

equivalent least square minimization

$$\text{argmin} \quad \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2$$

REGULARIZATION TECNIQUE: $\underset{w}{\text{argmin}} \; E_D(w) + \lambda E_W(w)$

KERNEL MODELS

consider a linear model

$$\begin{cases} E(w) = \frac{1}{2}\sum_{n=1}^{N}(w^T x_n - t_n)^2 + \lambda w^T w \\ E(w) = \frac{1}{2}(t - w^T x)^T(t - w^T x) + \lambda\|w\|^2 \end{cases}$$

where $X = [x_1, \dots, x_N]^T$ is design matrix $t = [t_1, \dots, t_N]$ output vector

the optimal solution is given by solving $\nabla E(w) = 0$

$$w^* = (X^T X + \lambda I_n)^{-1} X^T t = X^T (XX^T + \lambda I_n)^{-1} t$$

we can write $a = (X^T X + \lambda I_N)^{-1} t$ and we will have $y(x, w^*) = \sum_{n=1}^{N} a_n x_n^T x$

$k = X^T X$ is gram matrix $= \begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_N \\ \vdots & \ddots & \vdots \\ x_N^T x_1 & \cdots & x_N^T x_N \end{bmatrix}$

KERNEL TRICK if in algorithm appear an INNER PRODUCT $x^T x$ we can substitute them with kernel function $k(x, x)$

gram matrix will become $= \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$

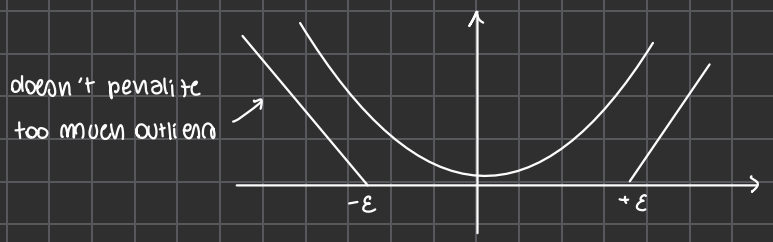
kernel families:
1. linear $k(x, x') = x^T x'$
2. polynomial $k(x, x') = (\beta x^T x' + \gamma)^d$
3. RBF $k(x, x') = \exp(-\beta |x - x'|^2)$
4. sigmoid $k(x, x') = \tanh(\beta x^T x' + \gamma)$

KERNELIZED VERSION OF SVM

$$y(x, a) = \text{sign}\left(\sum_{n=1}^{N} a_n t_n k(x_n, x) + w_0 \right)$$

KERNELIZED VERSION SVM REGRESSION

$$E_D(w) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} E_\varepsilon (y_n - t_n)$$



is not differentiable → difficult to solve

KNN

in an instance based model. considering a classification problem $f: X \to C$ where $C = \{+, -\}$ and a dataset $D = \{(x_n, t_n)_{n=1}^{N}\}$

classification with KNN:

1. find k nearest neighboor of new instance x
2. assign to new instance x the most common label among the majority of neighboor

likelihood function

$$P(C \mid x, D, K) = \frac{1}{K} \sum_{x_n \in N_K(x_n, D)} \mathbb{I}(t_n = C) \quad \text{with} \quad \mathbb{I}(e) = \begin{cases} 1 & e = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

MULTIPLE LEARNERS

VOTING:

given a dataset D

1. use D for train a set of models $y_m(x)$
2. make predictions

$$y_{VOTING}(x) = \sum_{m=1}^{M} w_m y_m(x)$$ REGRESSION

$$y_{VOTING}(x) = \text{argmax} \sum_{m=1}^{M} w_m \mathbb{I}(y_m(x) = C)$$ WEIGHTED MAJORITY OF CLASS

BAGGING → parallel method

1. generate M bootstrap dataset from D
2. train each model with M bootstrap dataset
3. make prediction using voting rule

$$y_{BAGGING} = \frac{1}{M} \sum_{m=1}^{M} y_m(x)$$

ADABOOST

given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

1. initialize $w_n = 1/N$, $n = 1, \dots, N$
2. for $m = 1, \dots, M$

train a weak learner by minimizing the weighted error function

- $J_m = \sum_{n=1}^{N} w_n^{(m)} \mathbb{I}(y_m(x_n) \neq t_n)$

- evaluate $\epsilon_m = \dfrac{\sum_{n=1}^{N} w_n \mathbb{I}(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n}$ and $\alpha_m = \ln\left(\dfrac{1-\epsilon_m}{\epsilon_m}\right)$

- update coefficients $w_n^{(m+1)} \leftarrow w_n^{(m)} \exp[\alpha_m \mathbb{I}(y_m(x_n) \neq t_n)]$

3. make prediction

$$Y_M = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

ANN

- REGRESSION → doesn't saturate
  - LINEAR UNITS → identity activation function $y = w^T h + b$
  - use a gaussian distribution noise model $p(t|x) = \mathcal{N}(t|y, \beta^{-1})$
  - LOSS-FUNCTION → MSE mean squared error $\frac{1}{N}\sum_{n=1}^{N}(y(x_n) - t_n)^2$
- BINARY CLASSIFICATION → SATURATE only correct answer
  - OUTPUTS UNITS → sigmoid activation function $y = \sigma(w^T h + b)$
  - LOSS-FUNCTIONS → Binary cross entropy $J(\theta) = E_{x,t\sim D}[-\ln[p(t|x)]]$
- MULTICLASS CLASSIFICATION → SATURATE minimal error
  - OUTPUTS UNITS → softmax activation functions $y_i = \text{softmax}(a^{(i)}) = \dfrac{\exp(a^{(i)})}{\sum_j \exp(a_j)}$
  - LOSS-FUNCTION → categorical cross entropy $J(\theta) = E_{x,t\sim D}[-\ln(\text{softmax})]$

BACKPROPAGATION

- is NOT a learning algorithm
- is ONLY compute gradient

FORWARD PASS

Require network depth ℓ
Require $W^{(i)}$, $i \in \{1, \dots, \ell\}$ weight matrices
Require $b^{(i)}$ $i \in \{1, \dots, \ell\}$ bias parameters
Require x input values
Require t target values
$h^{(0)} = x$
for $k = 1, \dots, \ell$ do

$$a^{(k)} = W^{(k)} h^{(k-1)} + b^{(k)}$$
$$h^{(k)} = f(a^{(k)})$$

end for
$y = h^{(\ell)}$
$J = L(t, y)$

BACKWARD PASS

$g \leftarrow \nabla_y J = \nabla L(t, y)$
for $k = \ell, \ell-1, \dots, 1$ DO
$g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^{(k)})$

$\nabla_{b^{(k)}} J \leftarrow g$
$\nabla_{W^{(k)}} J \leftarrow g (h^{(k-1)})^T$
propagate gradient to the next lower layer
$g \leftarrow \nabla_{h^{(k-1)}} J = (W^{(k)})^T g$
end for

STOCHASTIC GRADIENT DESCENT

Require learning rate $\eta > 0$
Require initial values of $\theta^{(1)}$
$k \leftarrow 1$
while stop criterion not met DO
sample a subset (minibatch) from D
compute the gradient estimate $g = \frac{1}{m} \nabla_\theta L(f(x^{(i)}, \theta^{(k)}), t^{(i)})$

apply update $\theta^{(k+1)} \leftarrow \theta^{(k)} - \eta g$
$k \leftarrow k+1$
end while

CNN

DIMENSIONS OF OUTPUT feature MAP:

$$w_{out} = \frac{w_{in} - w_k + 2p}{s} + 1 \qquad h_{out} = \frac{h_{in} - h_k + 2p}{s} + 1$$

NUMBER OF TRAINABLE PARAMS:

$$|\theta| = w_k \cdot h_k \cdot d_{in} \cdot d_{out} + \underbrace{d_{out}}_{\text{BIAS}}$$

GAUSSIAN MIXURE MODELS

mixed probability distribution p formed by k different Gaussian distributions

$$P(x) = \sum_{k=1}^{K} \pi_k N(x, \mu_k, \Sigma_k)$$

π_k = prior probability μ_k = mean Σ_k = covariance matrix

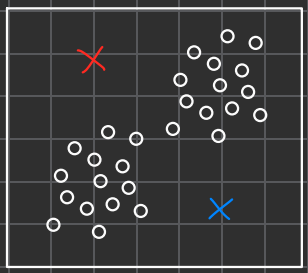
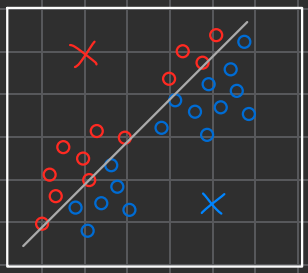
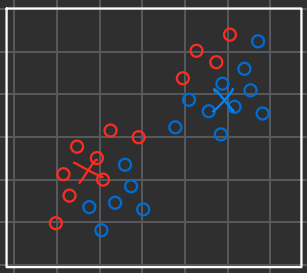
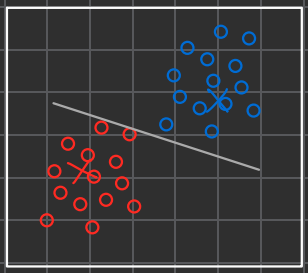
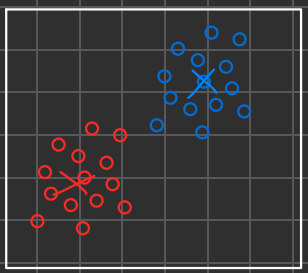
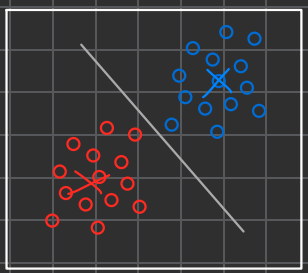
K-MEANS

is an unsupervised learning algorithm

1. begin with a decision value k = number of clusters
2. put any partition that classifies the data into k-clusters. this can be done randomly or systematically
   - consider k training samples as single element of clusters
   - assign to remain N-k training samples to the cluster with nearest centroid and after each assign recompute the centroide of cluster
3. take each sample in sequence and compute the distances from centroid of each clusters if sample is not in cluster with nearest centroid switch and update the centroid
4. repeat step 3 until convergence

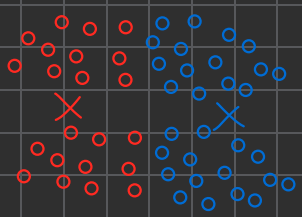
CONVERGENCE

1. for each culiten in step 2 the sum of distances for each training sample to centroid decrease
2. the are only finitely many partitions of training samples into k clusters

PROBLEM with K-MEAN

- NOT robust to outliers
- SENSITIVE to initial condition



!! this is a valid SOLUTION !!

EXPECTATION MAXIMIZATION

$$\gamma(z_k) = \frac{\pi_k N(x, \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x, \mu, \Sigma)} \rightarrow \text{POSTERIOR}$$

LATENT VARIABLE

we have two steps : 1. E-STEP → given π_k, μ_k and Σ_k compute $\gamma(z_{nk})$

2. M-STEP → given $\gamma(z_{nk})$ compute π_k, μ_k, Σ_k

PCA Principal Component Analysis

is widley technique used for various task:

1. dimensionality reduction
2. data compression
3. feature extraction
4. data visualization

VARIANCE MAXIMIZATION

goal maximize data variance after projection to some direction u_1

projected points $x_n^T u_1$

mean value of datapoints $\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$

variance of projected points $\frac{1}{N} \sum_{n=1}^{N} [u_1^T x_n - u_1^T \bar{x}]^2 = u_1^T S u_1$

covariance matrix of dataset $S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T = \frac{1}{N} X^T X$

we want maximize the projected variance: $\max_{u_1} u_1^T S u_1$

is equivalent to unconstraint maximization with lagrange multiplier:

$$\max_{u_1} u_1^T S u_1 + \lambda_1 (1 - u_1^T u_1)$$

solution derive w.r.t $u_1 \Rightarrow u_1^T S u_1 = \lambda_1 \rightarrow$ variance in maxima when λ_1 corresponds to the largest eigenvector

CALLED FIRST PRINCIPAL COMPONENT

ERROR MINIMIZATION

GOAL: approximate x_n using a lower dimensional representation:

$$\tilde{x}_n = \sum_{n=1}^{m} z_{ni} u_1 + \sum_{i=m+1}^{d} b_i u_i$$

↑ BIGGEST EIGENVALUE ↑ LOWEST EIGENVALUE

we will lose some information for this reason we compute the error we commit doing this approximation

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

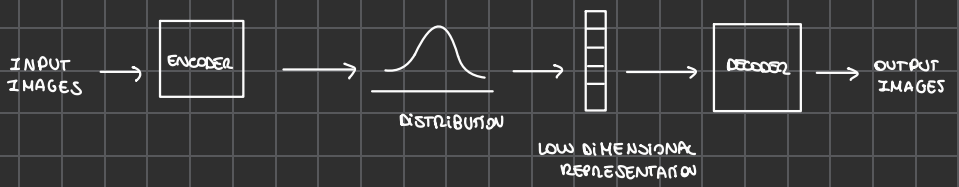
minimize the approximation error will become: $J = \sum_{i=m+1}^{N} u_i^T S u_i + \lambda_i (1 - u_i^T u_i)$

setting derivate of u_i to zero we have $Su_i = \lambda u_i$

the approximation error is then given by $J = \sum_{i=1}^{N} \lambda_i$

VAE Variational Auto-Encoders → focusing on learning latent variable space

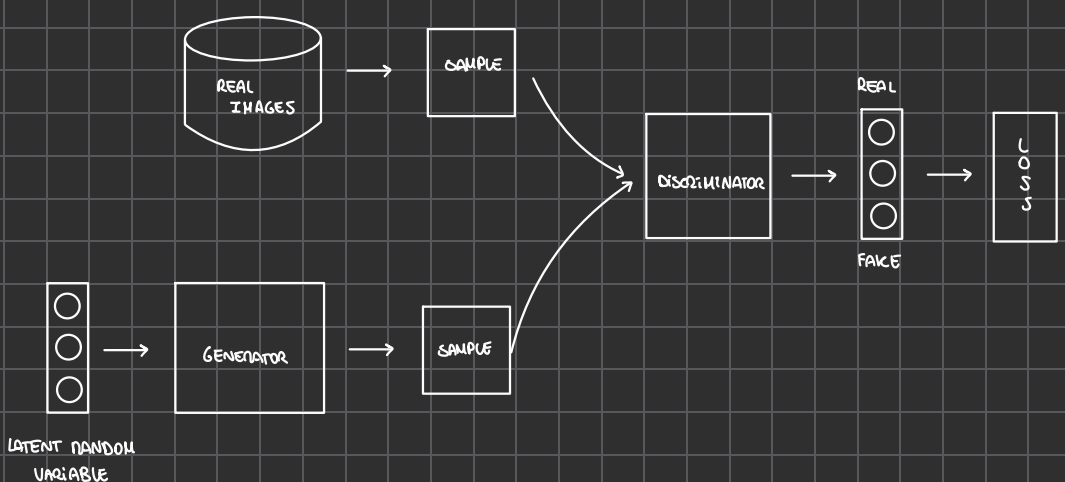
- modify data in specific directions
- identify meaningfull directions in latent space



HOW PRODUCE DISTRIBUTION → using parametric distribution

SAMPLING IS NOT DIFFERENTIABLE → re-parametrization

GAN Generative Adversarial Network



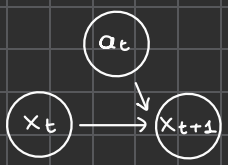
the loss of generator and the loss of discriminator "compete each other" the meaning is when one loss goes down the other goes up and viceversa the goal is when both stabilize

generator try to fool discriminator in believing that the images are real and discriminator try to discriminate which images is real and which no

LOSS OF GENERATOR AND DISCRIMINATOR DOESN'T DECREASE MONOTONIC

MARKOV PROPERTY

1. once the current state is known the evolution of the dynamic system doesn't not depend on the history of states, actions and observations
2. current state contain all information need to predict the future
3. future are conditionally independent of past states and observations given the current states



DETERMINISTIC TRANSITIONS

$\langle X, A, \delta, r \rangle$

X: finite set of states

A: finite set of actions

$\delta: X \times A \to X$ transition function

$r: X \times A \to \mathbb{R}$ reward function

NON-DETERMINISTIC TRANSITIONS

$\langle X, A, \delta, r \rangle$

X: finite set of states

A: finite set of actions

$\delta: X \times A \to 2^X$ transition function

$r: X \times A \to \mathbb{R}$ reward function

STOCHASTIC TRANSITIONS

$\langle X, A, \delta, r \rangle$

X: finite set of states

A: finite set of actions

$P(x'|x,A)$ probability distributions over transitions

$r: X \times A \times X \longrightarrow \mathbb{R}$ reward function

CUMULATIVE DISCOUNTED REWARD: $V^{\pi}(x_1) = \mathbb{E}[\bar{r}_1 + \gamma \bar{r}_2 + \gamma^2 \bar{r}_3 + \dots]$

if $r(a_i)$ is deterministic and know

$\pi^*(x_0) = \underset{a_i \in A}{\operatorname{argmax}}\ r(a_i)$

if $r(a_i)$ is deterministic and unknown

for each $a_i \in A$

- execute a_i and collect reward $r(i)$

$\pi^*(x_0) = a_i$ with $i = \underset{1 \dots |A|}{\text{argmax}}\ r(i)$

if $r(a_i)$ non-deterministic and known

$\pi^*(x_0) = \text{argmax}\ (\mathbb{E}(r_i))$

if $r(a_i)$ non-deterministic and un-known

I. initialize data structure Θ

II. for each time $t = 1, \dots, T$

- choose an action $a_{(t)} \in A$
- execute $a_{(t)}$ and collect reward $r_{(t)}$
- update data structure Θ

III. optimal policy $\pi^*(x_0) = \dots$ according with data structure Θ

assume $r(a_i) = \mathcal{N}(\mu_i, \sigma_i)$

1. initialize $\theta_{(0)}[i] \leftarrow 0$ and $c[i] \leftarrow 0$ $\quad i = 1, \dots, |A|$
2. for each time $t = 1, \dots, T$
   - choose an index $\hat{\imath}$ for action $a_{(t)} = a_{\hat{\imath}} \in A$
   - execute $a_{(t)}$ and collect reward $r_{(t)}$
   - increment $c[\hat{\imath}]$
   - update $\theta_{(t)}[\hat{\imath}] \leftarrow 1/c[\hat{\imath}]\ (r_{(t)} + (c[\hat{\imath}] - 1)\theta_{(t-1)}[\hat{\imath}])$
3. optimal policy: $\pi^*(x_0) = a_i$ with $i = \underset{i=1,\dots,|A|}{\text{argmax}}\ \theta_{(T)}[i]$

EXAMPLE K-ARM BANDIT → $r(a_i) = \mathcal{N}(\mu_i, \sigma_i)$

$$Q_n(a_i) \leftarrow Q_{n-1}(a_i) + \alpha\,[\,\bar{r} - Q_{n-1}(a_i)\,] \quad \text{and} \quad \alpha = 1 / 1 + V_{n-1}(a_i)$$

VALUE ITERATION

estimate value function and then compute π $\quad \pi^* = \underset{a \in A}{\text{argmax}}\ (r(x, a) + \gamma V^*(\underbrace{\delta(x, a)}_{\text{we don't know}}))$

RECURSIVE FORMULA OF $Q(x,a)$

$$V^*(x) = \max_{a \in A} \{ r(x,a) + \gamma V^*(\delta(x,a)) = \max_{a \in A} Q(x,a)$$

$$Q(x,a) = r(x,a) + \gamma V^*(\delta(x,a))$$

$$Q(x,a) = r(x,a) + \gamma \max_{a \in A} Q(\delta(x,a), a')$$

Q-LEARNING ALGORITHM → OFF POLICY

for each x, a initialize table entry $\hat{Q}_{(0)}(x,a) \leftarrow 0$

observe current state x

for each time $t = 1, \dots, T$

- choose an action a
- execute action a
- observe new state x'
- collect immediate reward $\bar{r}$
- update the table entry for $\hat{Q}(x,a)$

$$\hat{Q}_{(t)}(x,a) \leftarrow \bar{r} + \gamma \max_{a \in A} \hat{Q}_{(t-1)}(x',a')$$

$x \leftarrow x'$

optimal policy $\pi^*(x) = \arg\max_{a \in A} \hat{Q}_{(T)}(x,a)$

NON-DETERMINISTIC Q-LEARNING

$$\hat{Q}_n(x,a) \leftarrow \hat{Q}_{n-1}(x,a) + \alpha [\underbrace{r + \gamma \max \hat{Q}_{n-1}(x,a)}_{\text{WHAT I CAN COMPUTE WHEN I EXECUTE AN ACTION}} - \underbrace{\hat{Q}_{n-1}(x,a)}_{\text{THE KNOWLEDGE I HAVE BEFORE}}]$$

TEMPORAL DIFFERENCE LEARNING

Q-LEARNING reduce discrepancy between successive Q estimate

$$Q^{\lambda}(x_t, a_t) = r_t + \gamma [(1-\lambda) \max_{a \in A} \hat{Q}(x_t, a_t) + \lambda Q^{\lambda}(x_{t+1}, a_{t+1})]$$

SARSA → ON POLICY → according with current policy

is based on tuple $\langle s, a, r, s', a' \rangle$

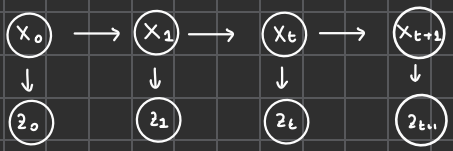
$$\hat{Q}(x,a) \leftarrow \hat{Q}_{(n-1)}(x,a) + \alpha\left[r + \gamma \hat{Q}_{n-1}(x',a') - \hat{Q}_{n-1}(x,a)\right]$$

POLICY ITERATION → we don't estimate Q function we have an parametric representation of policy $\pi_\theta(x)$ we estimate the policy gradient through experiment until terminal condition

MARKOV CHAIN

future depends only on the current state

HMM Hidden Markov Models



state x are discrete and not observable

$\langle X, Z, \pi_0 \rangle$

transition model $P(x_t \mid x_{t-1})$

observation model $P(z_t \mid x_t)$

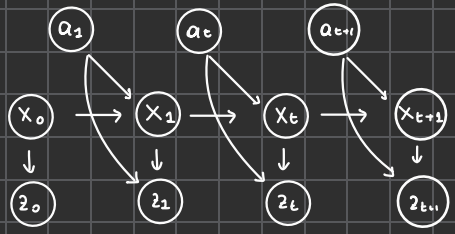
FILTERING → problem of estimating the state of a system at time t given all the past observations

$$P(x_T = k \mid z_{1:T}) = \alpha_T^k / \sum_j \alpha_T^j$$

SMOOTHING → given observation so far we want estimate what was the value of state at the past

$$P(x_t = k \mid z_{1:T}) = \alpha_t^k \beta_t^k / \sum_j \alpha_t^j \beta_t^j$$

POMDP = MDP + HMM



$\langle X, A, Z, \delta, r, O \rangle$

X : states

A : actions

Z : observations

$P(x_0)$: prob. initial state

$\delta(x,a,x') = P(x' \mid x,a)$: prob distr. over trans. function

$r(x,a)$: reward function

$O(x',a,z') = P(z' \mid x',a)$: prob distribution over observations

we want find policy but we don't know state → belief$_t(x)$ = prob. distribution over states