FORMULARIO ML:

SLIDES 1.

SUPERVISED LEARNING $D = \{(x_i, y_i)\}_{i=1}^{m}$

UNSUPERVISED LEARNING $D = \{(x_i)\}_{i=1}^{m}$

REINFORCEMENT LEARNING $\pi : S \to A$ $D = \{(\langle s_0, a_1, r_1, s_1, \ldots, a_n, r_n, s_n \rangle)\}_{i=1}^{n}$

$CONSISTENT(h, D) \equiv (\forall x \in D)\ h(x) = c(x)$

SLIDES 2.

$ERROR_D(h) \equiv \Pr_{x \in D}[f(x) \neq h(x)]$

$ERROR_S(h) \equiv 1/n \sum_{x \in S} \delta(f(x) \neq h(x))$

$BIAS \equiv E_S[error_S(h)] - ERROR_D(h)$

K FOLD CROSS VALIDATION:

for 1... k do

$T_i \leftarrow \{D - S_i\}$

$h_i \leftarrow L(T_i)$

$\delta_i \leftarrow error_{S_i}(h_i)$

$ACCURACY = 1 - error_{LD}$

RETURN $error_{L,D} = 1/k \sum_{i=1}^{k} \delta_i$

OVERFIT: $ERROR_S(h) < ERROR_S(h')$

$ERROR_D(h) > ERROR_D(h')$

$$RECALL = \frac{TP}{TP + FN}$$

$$PRECISION = \frac{TP}{TP + FP}$$

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

} CLASSIFICATION

$$MAE = \frac{1}{M} \sum_{i=1}^{M} |\hat{f}(x_i) - t_i|$$

$$MSE = \frac{1}{M} \sum_{i=1}^{M} (\hat{f}(x_i) - t_i)^2$$

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

SLIDES 3.

consider binary classification (+, -)

$$\text{ENTROPY}(S) = -P_{(+)} \log_2 P_{(+)} - P_{(-)} \log_2 P_{(-)}$$

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

$$\text{GAIN}(S, A) = \text{ENTROPY}(S) - \sum_{v \in \text{Values}(A)} (|S_v| / |S|) \, \text{ENTROPY}(S)$$

$$\text{GAIN RATIO}(S, A) = \frac{\text{GAIN}(S, A)}{\text{SPLITINFORMATION}(S, A)}$$

$$\text{SPLITINFORMATION}(S, A) = -\sum_{i=1} |S_i| / |S| \log_2 |S_i| / |S|$$

SLIDES 4.

~~BAYES'S RULE $P(\text{CAUSE} \mid \text{EFFECT}) = P(\text{CAUSE})$~~

$$\text{BAYES'S RULE} = P(\text{CAUSE} \mid \text{EFFECT}) = \frac{P(\text{EFFECT} \mid \text{CAUSE}) P(\text{CAUSE})}{P(\text{EFFECT})}$$

SLIDES 5.

THEOREM OF TOTAL PROBABILITY: $P(B) = \sum_{i=1}^{N} P(B \mid A_i) P(A_i)$

LEARNING AS PROBABILISTIC ESTIMATION

$$P(h \mid D) = \frac{P(D \mid h) P(h)}{P(D)}$$

$P(h)$: prior probability of hypothesis h (prior)

$P(D)$: prior probability of training data D (normalization factor)

$P(h \mid D)$: prior probability of h given D (posterior)

$P(D \mid h)$: probability of D given h (likelihood)

MAP hypothesis $h_{MAP} = \underset{h \in H}{\text{argmax}} \, P(h \mid D) = \underset{h \in H}{\text{argmax}} \left(\frac{P(D \mid h) P(h)}{P(D)} \right)$

↑ MAXIMUM APOSTERIRI HYPOTHESIS

ML hypothesis $h_{ML} = \underset{h \in H}{\text{argmax}} \, P(D \mid h)$

BAYES OPTIMAL CLASSIFIER:

$$V_{OB} = \underset{v_j \in V}{\text{argmax}} \sum_{h_i \in H} P(v_j \mid x, h_i) P(h_i \mid D)$$

target function $f: X \to V$ $V = \{v_1, \ldots, v_k\}$ dataset D

BERNOULLI DISTRIBUTION

$$P(X = k; \theta) = \theta^k (1 - \theta)^{1-k}$$

MULTIVARIATE BERNOULLI DISTRIBUTION

$$P(X_1 = k_1, \ldots ; \theta_1, \ldots, \theta_M) = \prod_{i=1}^{n} P(X_i = k_i \mid \theta_i) = \prod_{i=1}^{n} \theta_i^{k_i} (1-\theta_i)^{1-k_i}$$

BINOMIAL DISTRIBUTION

$$P(X = k; n, \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

MULTINOMIAL DISTRIBUTION

$$P(X_1 = k_1, \ldots, X_d ; n, \theta_1 \ldots \theta_d) = \frac{n!}{k_1! \ldots k_d!} \theta_1^{k_1} \ldots \theta_d^{k_d}$$

NAIVE BAYES CLASSIFIER

x is conditionally independent ~~to approximate the relation~~ to Y given Z

$$P(X, Y \mid Z) = P(X \mid Y, Z) P(Y \mid Z) = P(X \mid Z) P(Y \mid Z)$$

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j \mid D) \prod_i P(a_i \mid v_j, D) \quad \text{with assumption}$$

$$P(a_1, \ldots, a_n \mid v_j, D) = \prod_i P(a_i \mid v_j, D)$$

NAIVE BAYES ALGORITHM

target function $f: X \to V$ $\quad X = A_1 \times \ldots \times A_M$ $\quad V = \{v_1, \ldots, v_k\}$

new instance $x = \langle a_1, a_2, \ldots, a_n \rangle$

for each target value $v_j \in V$

$\hat{P}(v_j \mid D) \leftarrow$ estimate $P(v_j \mid D)$

for each attribute A_k

for each attribute $a_i \in A_k$

$\hat{P}(a_i \mid v_j, D) \leftarrow$ estimate $P(a_i \mid v_j, D)$

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j \mid D) \prod_{a_i \in x} \hat{P}(a_i \mid v_j, D)$$

NAIVE BAYES ESTIMATION

$$\hat{P}(a_i \mid v_j, D) = \frac{\#\langle \ldots, a_i, \ldots, v_j \rangle + mp}{\#\langle \ldots, v_j \rangle + m}$$

WORD REPRESENTATION

1. BOOLEAN FEATURE VECTOR $\begin{cases} d_i = 1 & \text{if } w_i \text{ appears in doc} \\ 0 & \text{otherwise} \end{cases}$

2. ORDINAL FEATURE VECTOR $d_i = k$ if w_i appears k times in DOC

(Multinomial Distribution) → (Bag of World)

3. tf-idf : $d_i = tf(w_i, doc)\, idf(w_i, D)$

term frequency inverse document frequency

MULTIVARIATE BERNOULLI DISTRIBUTION (NAIVE BAYES)

boolean feature vector $d = \langle d_1, \dots, d_n \rangle$ of generic doc $\in$ Docs

$$P(d \mid c_j, D) = \prod_{i=1}^{M} P(w_i \mid c_j, D)^{d_i} (1 - P(w_i \mid c_j, D))^{1-d_i}$$

MAXIMUM LIKELIHOOD SOLUTION

$$\hat{P}(w_i \mid c_j, D) = \frac{t_{ij} + 1}{t_j + 2}$$

t_{cj} = number of documents in D of class c_j containing word w_i

t_j = number of documents D of class c_j

1, 2 = LAPLACIAN TERM

MULTINOMIAL NAIVE BAYES DISTRIBUTION

$$P(d \mid c_j, D) = \frac{m!}{d_1! \dots d_m!} \prod_{i=1}^{m} P(w_i \mid c_j, D)^{d_i}$$

MAXIMUM LIKELIHOOD SOLUTION

$$\hat{P}(w_i \mid c_j, D) = \frac{\sum_{doc \in D} tf_{i,j} + \alpha}{\sum_{doc \in D} tf_{i,j} + \alpha |V|}$$

SLIDE 6.

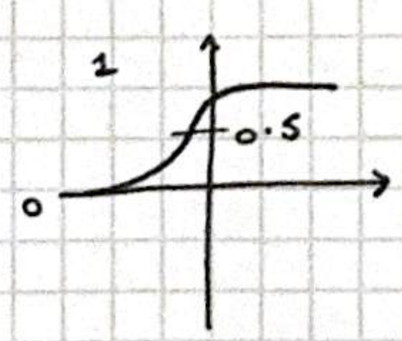
PROBABILISTIC GENERATIVE MODEL

$$P(C_1 \mid x) = \frac{P(x \mid C_1) P(C_1)}{P(x)} = \frac{P(x \mid C_1) P(C_1)}{P(x \mid C_1) P(C_1) + P(x \mid C_2) P(C_2)} =$$

$$= \frac{1}{1 + \exp(-a)} = \delta(a)$$

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

SIGMOID FUNCTION:



MAXIMUM LIKELIHOOD SOLUTION K CLASSES

$$P(C_k \mid x) = \frac{P(x \mid C_k) P(C_k)}{\sum_j P(x \mid C_j) P(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

SOFTMAX

NEWTON - RAPHSON:

gradient of error with respect to $\tilde{w}$

$$\nabla E(\tilde{w}) = \sum_{n=1}^{N} (y_n - t_n)\tilde{x}_n$$

gradient descent step

$$\tilde{w} \leftarrow \tilde{w} - H(\tilde{w})^{-1} \nabla E(\tilde{w})$$

$H(\tilde{w}) = \nabla\nabla E(\tilde{w})$ is the Hessian matrix of $E(\tilde{w})$

ITERATIVE METHOD

1. initialize $\tilde{w}$
2. repeat until termination

$$\tilde{w} \leftarrow \tilde{w} - (\tilde{X}^T R(\tilde{w}) \tilde{X})^{-1} \tilde{X}^T (y(\tilde{w}) - t)$$

with:

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_N^T \end{pmatrix} \quad t = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} \quad y(\tilde{w}) = (y_1 \dots y_n)^T$$

$R(\tilde{w})$: diagonal matrix with $R_{NN} = y_n(1 - y_n)$

MULTICLASS LOGISTIC REGRESSION

$$P(T | \tilde{w}_1, \dots, \tilde{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k | \tilde{x}_n)^{t_{nk}} = \prod_{n=1}^{n} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

cross entropy error function

$$E(\tilde{w}_1, \dots, \tilde{w}_K) = -\ln(P(T | \tilde{w}_1, \dots, \tilde{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

SLIDES 7.

- LINEAR DISCRIMINANT FUNCTION

$y : X \to \{C_1, \dots, C_k\}$

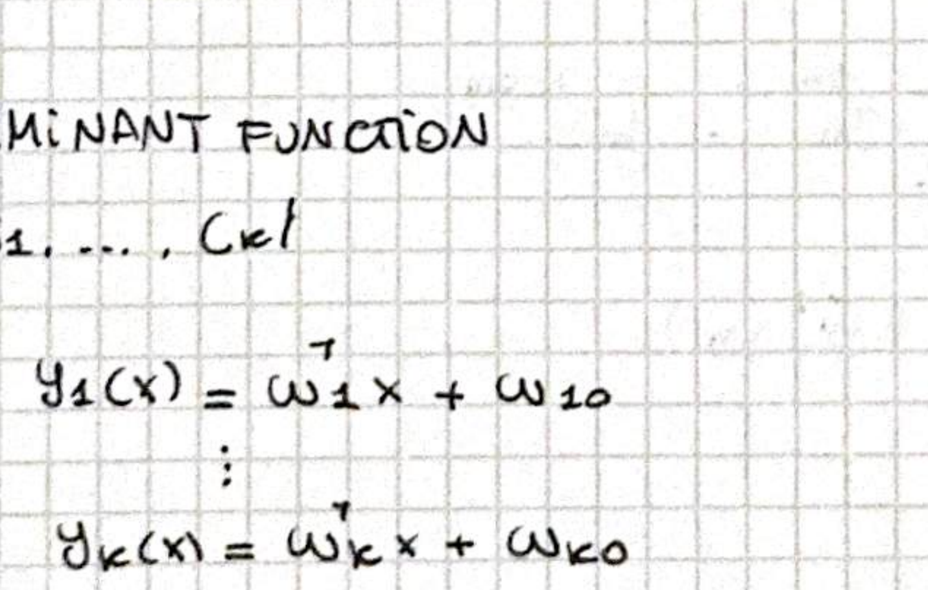
k classes $y_1(x) = w_1^T x + w_{10}$

$\vdots$

$y_k(x) = w_k^T x + w_{k0}$

COMPACT NOTATION

$$y(x) = \begin{pmatrix} y_1(x) \\ \vdots \\ y_k(x) \end{pmatrix} = \begin{pmatrix} w_1^T x + w_{10} \\ \vdots \\ w_k^T x + w_{k0} \end{pmatrix} = \begin{pmatrix} \tilde{w}_1^T \\ \vdots \\ \tilde{w}_k^T \end{pmatrix} \tilde{x} = \tilde{W}^T \tilde{x}$$



GAUSSIAN NAIVE MODEL

$P(C_k) = \pi_k \quad P(x|C_k) = N(x, \mu_k, \Sigma)$

TRAINING by estimating from D

$\hat{\pi}_k = \frac{N_k}{N} \quad \hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} x_n \quad \hat{\Sigma} = \sum_{k=1}^{K} \frac{N_k}{N} S_k$

$S_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$

PREDICTION

$\underset{C_k \in C}{\operatorname{argmax}} \hat{P}(C_k|x') = \underset{C_k \in C}{\operatorname{argmax}} \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \ln(\hat{P}(x'|C_k)P(C_k))$

POSTERIOR DISTRIBUTIONS WITH PARAMETRICS MODELS

K CLASSES

$P(C_i|x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = w^T x + w_0$

PROBABILISTIC DISCRIMINATIVE MODELS

estimate directly $P(C_k|\tilde{x}, D) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \tilde{w}^T \tilde{x}$

MAXIMUM LIKELIHOOD $\tilde{w}^* = \underset{\tilde{w}}{\operatorname{argmax}} \ln(P(t|\tilde{w}, x))$

LOGISTIC REGRESSION 2 classes dataset $D = \{(\tilde{x}_m, t_m)\}_{m=1}^{N}$ with $t_n \in \{0, 1\}$

likelihood:

$$p(t|\tilde{w}) = \prod_{m=1}^{N} y_m^{t_n} (1 - y_m)^{1-t_m}$$

with $y_m = p(C_1|\tilde{x}_m) = \sigma(\tilde{w}^T \tilde{x}_m)$

CROSS ENTROPY:

$E(\tilde{w}) = -\ln(P(t|\tilde{w}))$

optimitation → $\underset{\tilde{w}}{\operatorname{argmin}} E(\tilde{w})$

MULTICLASS

$$y(x) = \begin{pmatrix} y_1(x) \\ \vdots \\ y_K(x) \end{pmatrix} = \begin{pmatrix} \tilde{\omega}_1^T \tilde{x} \\ \vdots \\ \tilde{\omega}_K^T \tilde{x} \end{pmatrix} = \tilde{W}^T \tilde{x}$$

classify x as C_k with $k = \underset{j=1,\dots,K}{\operatorname{argmax}} \{ y_i(x) \}$

LEAST SQUARES

minimize sum of squares error function:

$$E(\tilde{W}) = \tfrac{1}{2} \operatorname{tr}\{ (\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T) \}$$

closed-form solution:

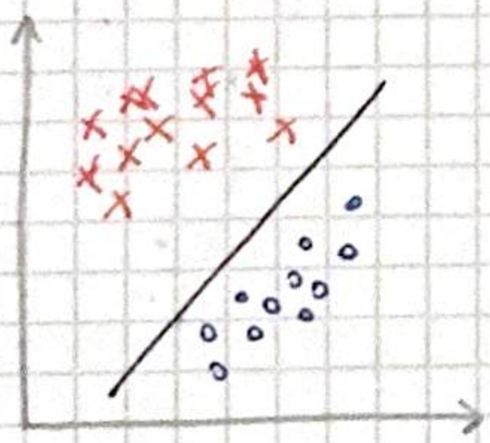
$$\tilde{W} = \underbrace{(\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T}_{\tilde{X}^\dagger} T$$

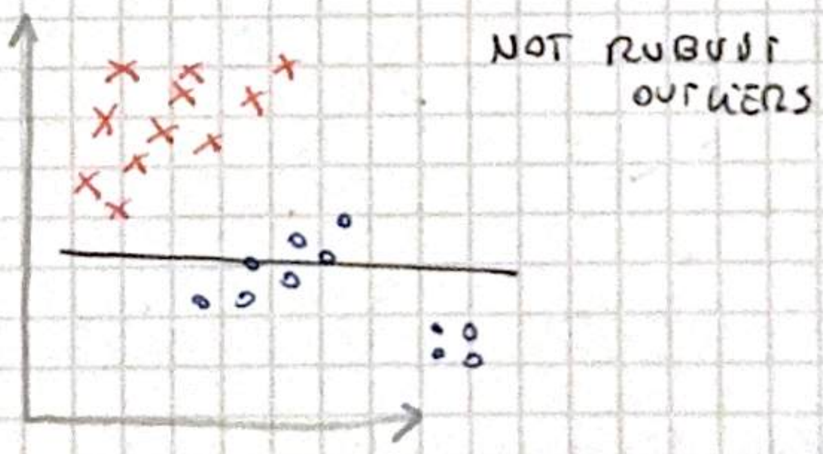
learned model

$$y(x) = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^\dagger)^T \tilde{x}$$

use $\tilde{W}$ to compute $y(x) = \tilde{W}^T \tilde{x} = \begin{pmatrix} y_1(x) \\ \vdots \\ y_K(x) \end{pmatrix}$

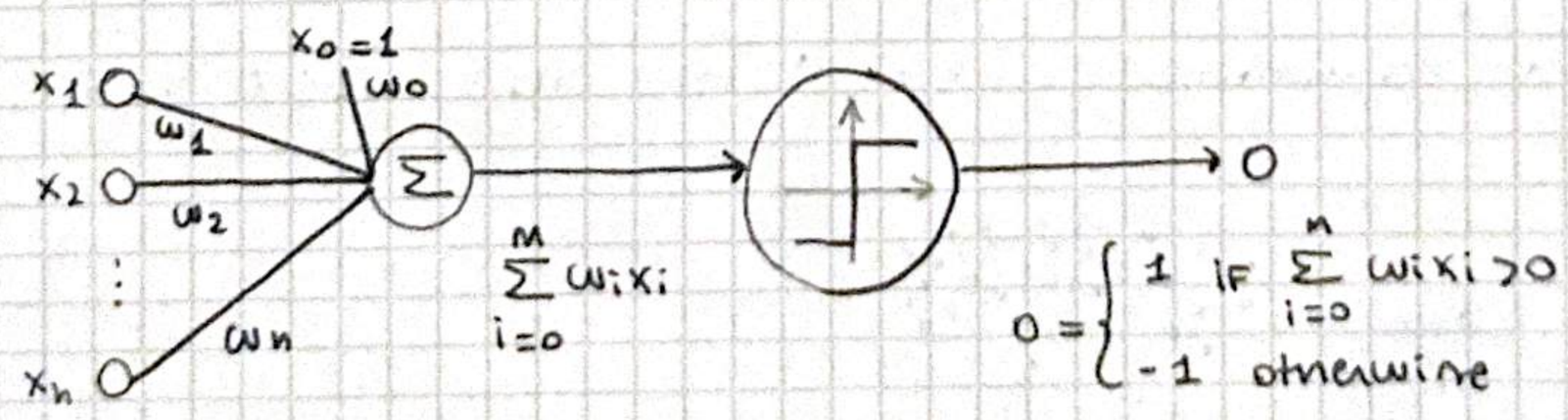
predict $k = \underset{j \in \{1,\dots,K\}}{\operatorname{argmax}} \{ y_i(x') \}$





PERCEPTRON

$f: \mathbb{R}^d \to \{-1, +1\}$



$$o(x_1, \dots, x_d) = \begin{cases} 1 & \text{if } \omega_0 + \omega_1 x_1 + \dots + \omega_d x_d > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$o(x) = \begin{cases} 1 & \text{if } w^T x > 0 \\ -1 & \text{otherwise} \end{cases} = \text{sign}(w^T x)$$

minimize squared error function:

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

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

Update of weights

$$w_i \leftarrow w_i + \Delta w_i$$

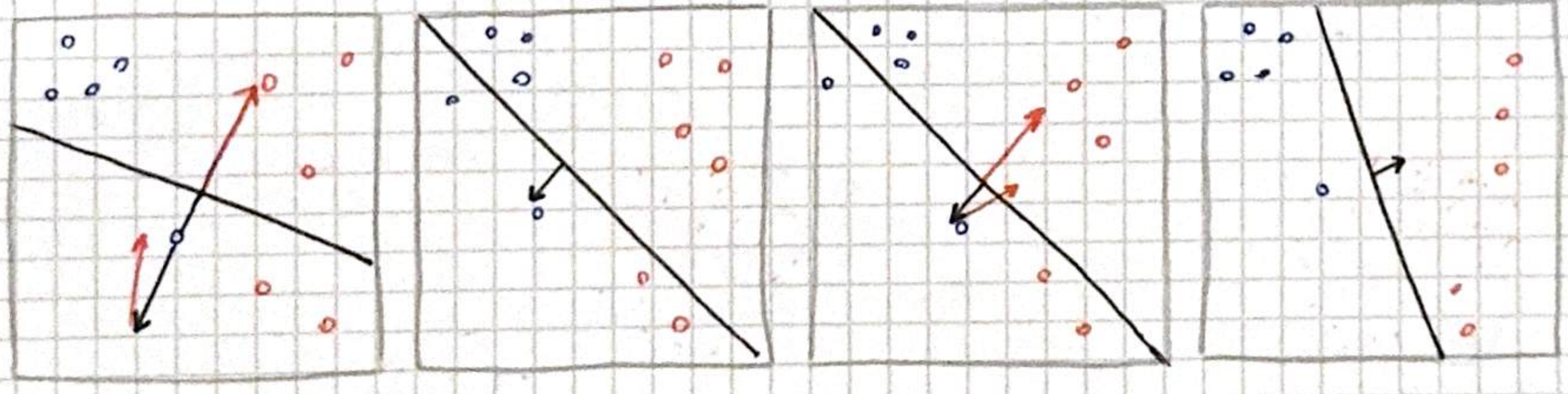
$$\Delta w_i = -\eta \frac{\Delta E}{\partial w_i} = \eta \sum_{n=1}^{N} (t_n - w^T x_n) x_{i,n}$$

BATCH-MODE → all dataset $\Delta w_i = \eta \sum_{(x,t) \in D} (t - o(x)) x_i$

MINI BATCH-MODE → choose small subset $\Delta w_i = \eta \sum_{(x,t) \in S} (t - o(x)) x_i$

INCREMENTAL-MODE → choose one sample $(x,t) \in D$

$$\Delta w_i = \eta (t - o(x)) x_i$$

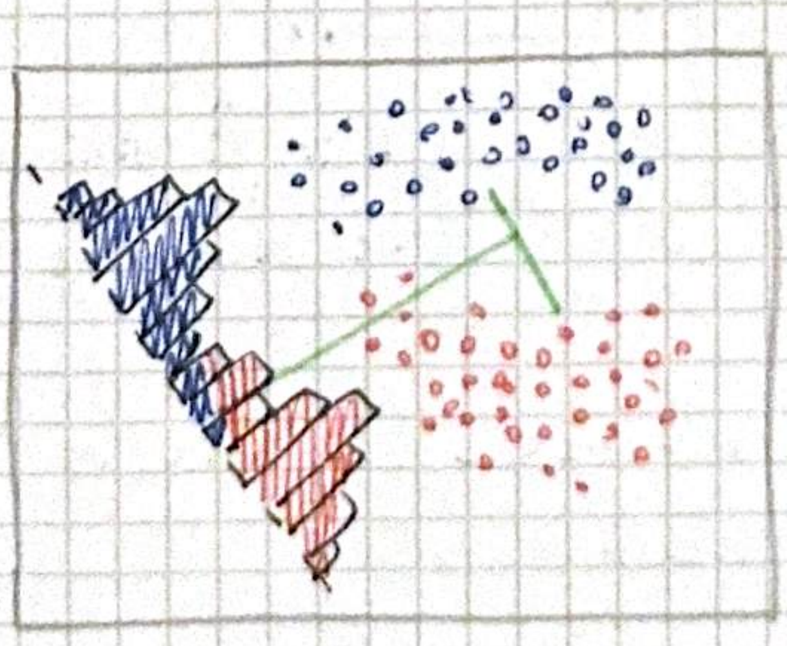


FISHER LINEAR DISCRIMINANT

dataset N_1 points in C_1 and N_2 points in C_2

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

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



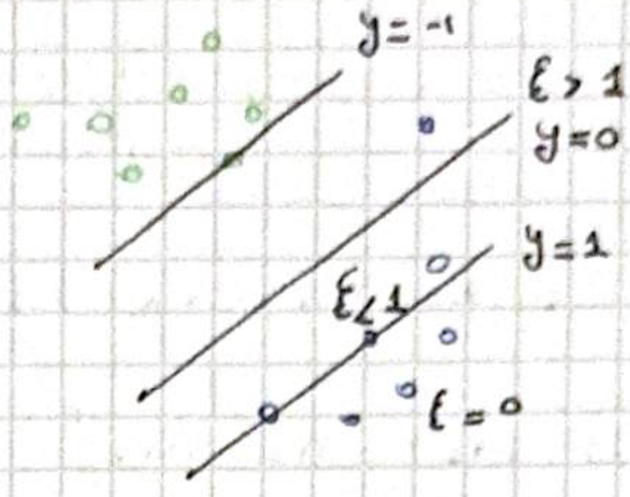
SVM SOFT MARGIN CONSTRAINT

$\xi_n = 0$ ON OR INSIDE CORRECT BOUNDARIES

$0 < \xi_n \leq 1$ INSIDE ~~BOUNDARIES~~ MARGIN CORRECT SIDE

$\xi_n > 1$ WRONG SIDE MARGIN

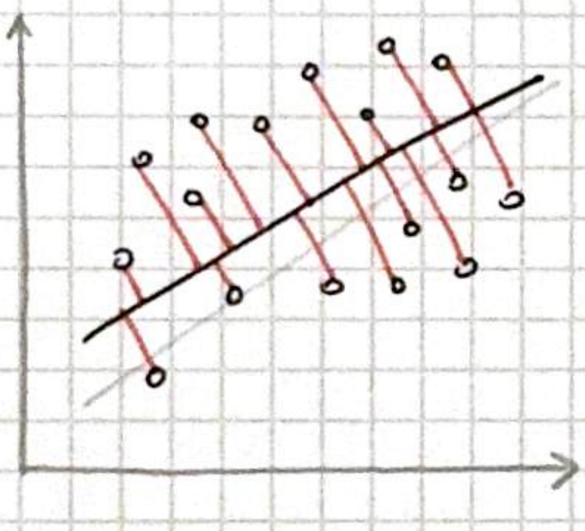
$$w^*, w_0^* = \text{argmin} \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \xi_n$$



SLIDES 8

LINEAR MODEL FOR REGRESSION:

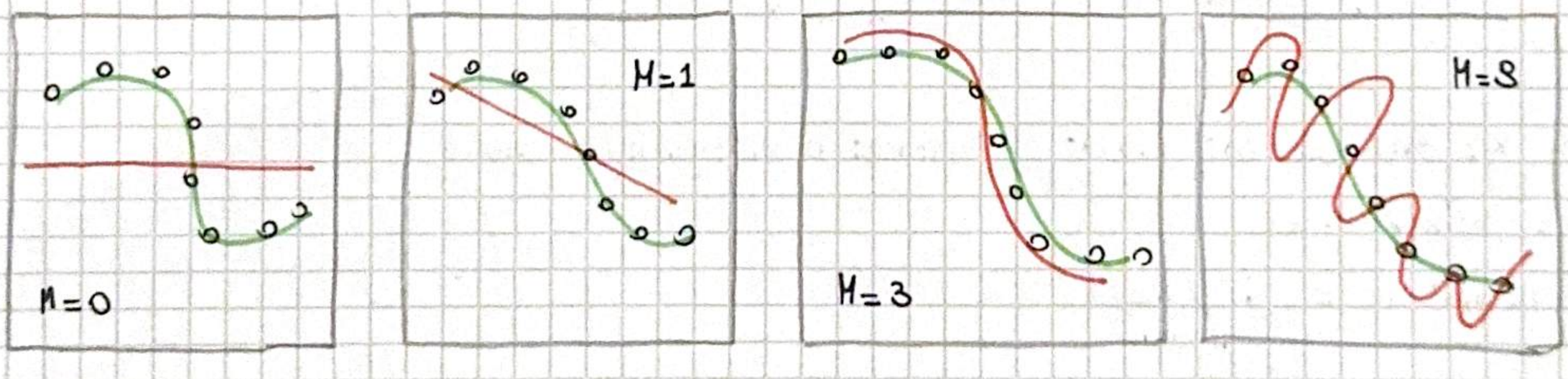
$$y(x;w) = w_0 + w_1 x_1 + \dots + w_d x_d = w^T x \quad \text{with} \quad x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$



$y = w_0 + w_1 x_1$

USING NON-LINEAR FUNCTIONS OF INPUT VARIABLES: $y(x,w) = \sum_{j=0}^{M} w_j \phi_j(x) = w^T \phi(x)$

POLYNOMIAL CURVE FITING:



UNDERFITTING

OVERFITTING

MAXIMUM LIKELIHOOD (ZERO-MEAN GAUSSIAN NOISE ASSUMPTION)

$$\underset{w}{\text{argmax}}\ P(\{t_1, \dots, t_N\} | x_1, \dots, x_N, w, \beta)$$

CORRESPOND TO LEAST SQUARE ERROR MINIMIZATION

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

$$E_D(w) = \frac{1}{2}(t - \Phi w)^T (t - \Phi w) \qquad \Phi = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_M(x_N) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_M(x_N) \end{bmatrix} \quad t = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$$

$$w_{ML} = \underbrace{(\Phi^T \Phi)^{-1} \Phi^T}_{\text{PSEUDO INVERSE}} t$$

FISHER CRITERION

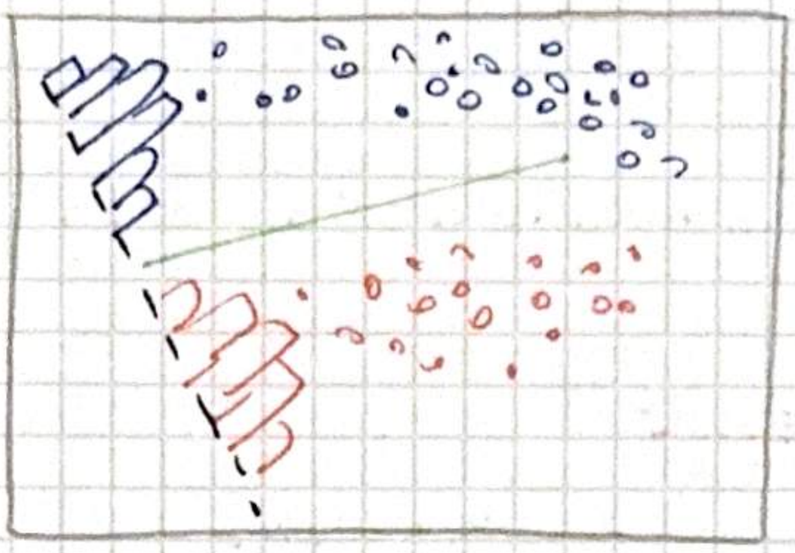
$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

S_W

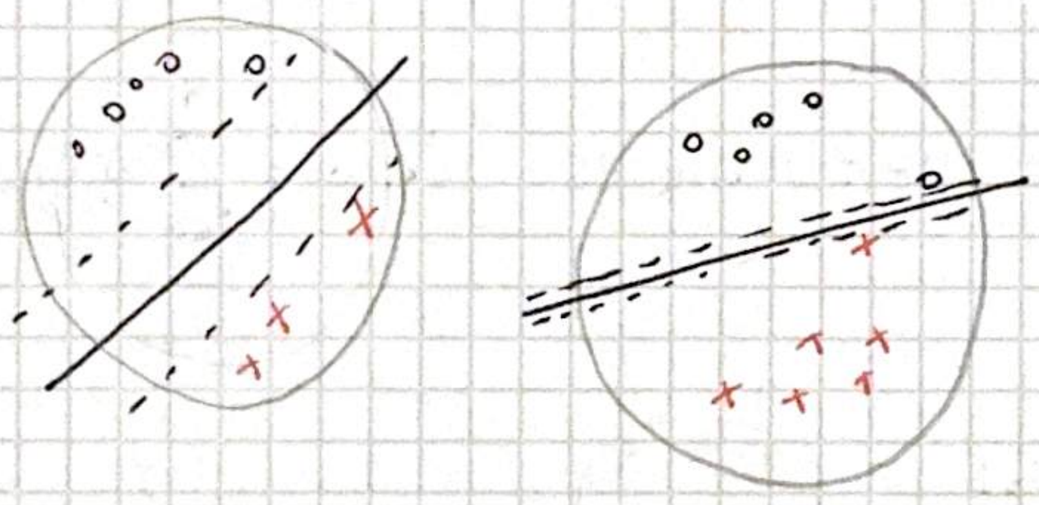
$S_B = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$

$S_B = (m_2 - m_1)(m_2 - m_1)^T$

maximite $J(w)$ by solving $\frac{d}{dw}(J(w)) = 0$



SUPPORT VECTOR MACHINE



let x_k the closest point of dataset D to the hyperplane

$h = w^T x + w_0 = 0$

MARGIN IS = $\frac{|y(x_k)|}{\|w\|}$

given dataset D and hyperplane h the margin is computed as

$$\min_{n=1,\dots,N} \frac{|y(x_n)|}{\|w\|} = \dots = \frac{1}{\|w\|} \min_{n=1,\dots,N} [t_n (w^T x_0 + w_0)]$$

given dataset D and hyperplan h^*: $w^{*T} x + w_0$ with maximum margin as

$$w^*, w_0^* = \arg\max_{w, w_0} \frac{1}{\|w\|} \min_{n=1,\dots,N} [t_n (w^T x_n + w_0)]$$

quadratic problem solved with LAGRAGIAN METHOD

$$w^* = \sum_{n=1}^{N} a_n^* t_n x_n$$

ai in result of LAGRAGIAN METHOD optimization problem

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m x_n^T x_m$$

SEQUENTIAL LEARNING: $\hat{\omega} \leftarrow \hat{\omega} - \eta \nabla E_n$

REGULARIZATION TECNIQUE $\arg\min_{\omega} E_D(\omega) + \lambda E_\omega(\omega)$ q with λ regularization factor

SLIDES 8

- LINEAR MODELS GRAM-MATRIX

consider a linear model $y(x,\omega) = \omega^T x$ with dataset $D = \{(x_n, t_n)_{n=1}^N\}$

① $E(\omega) = \frac{1}{2} \sum_{n=1}^{N} (\omega^T x_n - t_m)^2 + \lambda \omega^T \omega$

② $E(\omega) = (t - X\omega)^T (t - X\omega) + \lambda \|\omega\|^2$

optimal solution $\nabla E(\omega) = 0$ ①

$$\omega^T = -\frac{1}{\lambda} \sum_{n=1}^{N} (\omega^T x_n - t_m) x_m \qquad a_n = -\frac{1}{\lambda} (\omega^T x_n - t_n)$$

$$\omega^* = \sum_{n=1}^{N} a_n x_n$$

optimal solution $\nabla E(\omega) = 0$ ②

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

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

then $y(\omega, x) = \sum_{n=1}^{N} a_n x_n^T x \qquad a = (K + \lambda I_N)^{-1} t$

- GRAM-MATRIX → $K = \begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_N \\ \vdots & \ddots & \vdots \\ x_N^T x_1 & \cdots & x_N^T x_N \end{bmatrix}$

- KERNEL FUNCTION

similarity function $k: X \times X \to \mathbb{R}$

come prima ma con k

KERNEL TRICK

repo replace inner product $x^T x'$ with $k(x, x')$

KERNEL FUNCTION

a real valued function $k(x,x') \in \mathbb{R}$ for $x, x' \in X$ where X is some abstract space, usually:

- SYMMETRIC
- NON NEGATIVE

NORMALIZATION:

- min MAX → $\bar{x} = \frac{x - min}{MAX - min}$
- unit vector $\bar{x} = \frac{x}{\|x\|}$
- standardization → $\bar{x} = \frac{x - \mu}{\sigma}$

KERNEL FAMILIES:

- LINEAR $K(x, x') = x^T x'$
- POLYNOMIAL $K(x, x') = (\beta x^T x' + \gamma)^d \quad d \in \{2, 3, \dots\}$
- RADIAL BASIS FUNCTION (RBF) $K(x, x') = e^{(-\beta |x - x'|^2)}$
- SIGMOID $K(x, x') = \tanh(\beta x^T x' + \gamma)$

SVM kernelized → CLASSIFICATION

$$y(x; \alpha) = sign\left(w_0 + \sum_{n=1}^{N} \alpha_n x_n^T x\right) \rightarrow sign\left(w_0 + \sum_{n=1}^{N} \alpha_n K(x_n, x)\right)$$

REGRESSION

$$E(w) = \sum_{n=1}^{N} E(y_n, t_n) + \lambda \|w\|^2$$

$$y(w, x, w^*) = \sum_{n=1}^{N} \alpha_n K(x_n, x) \qquad \alpha = (K + \lambda I_N)^{-1} t$$

SLIDES 10

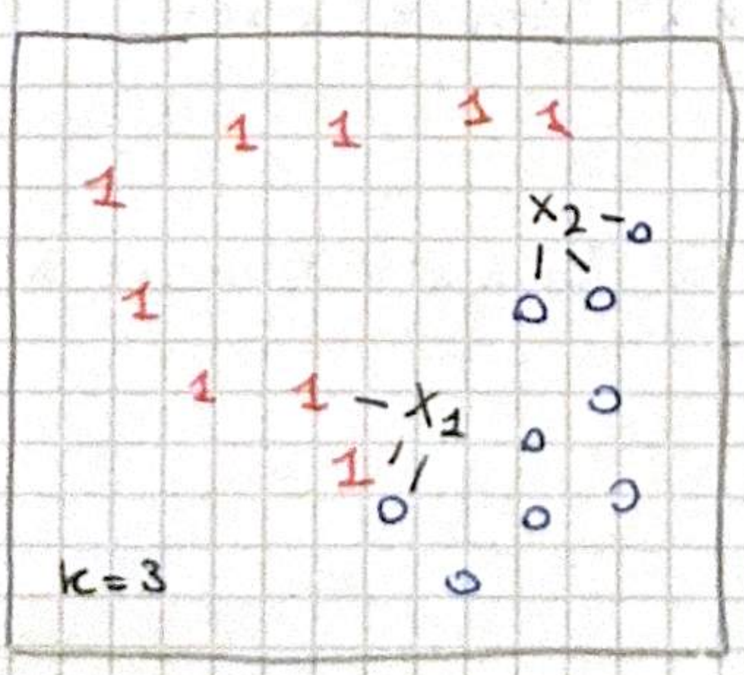
K-NEAREST NEIGHBORS

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

LIKELIHOOD OF CLASS C FOR NEW ISTANCES x

$$P(C | x, D, k) = \frac{1}{k} \sum_{x_n \in N_k(x_n, D)} \mathbb{I}(t_n = C)$$

$N_k(x_n, D)$ the k nearest point to x_n and $\mathbb{I}(e) = \begin{cases} 1 & \text{if } e = \text{true} \\ 0 & \text{otherwise} \end{cases}$



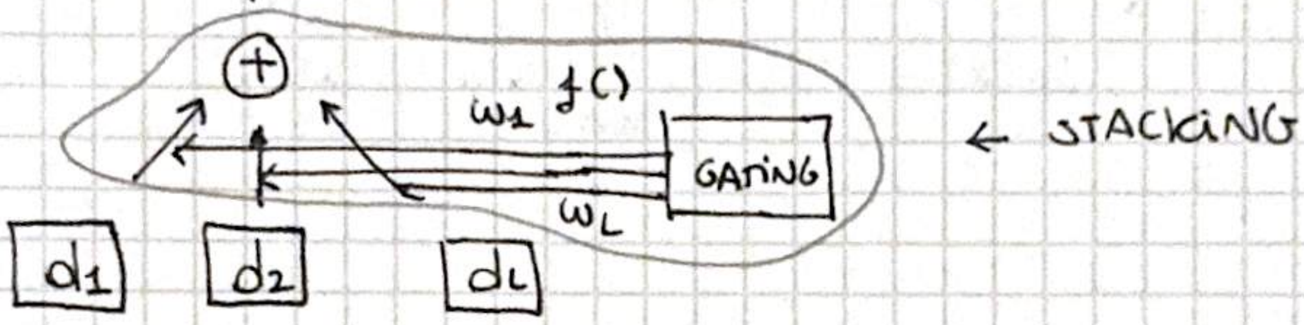
SLIDES 11

VOTING:

1. dataset D to train models $y_m(x)$
2. make predictions with:

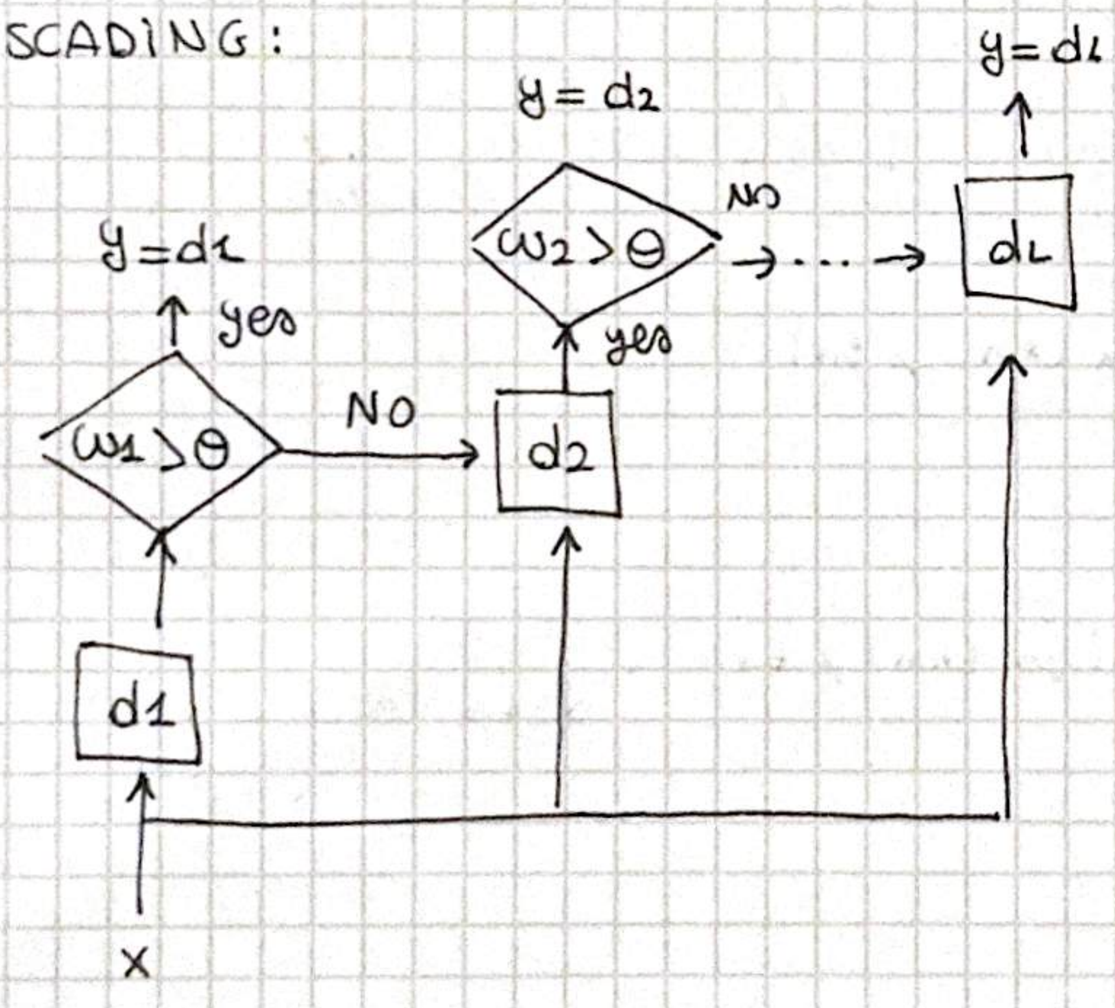
$$y_{VOTING}(x) = \sum_{m=1}^{M} w_m y_m(x) \qquad \text{REGRESSION}$$

$$y_{VOTING}(x) = \underset{c}{\text{argmax}} \sum_{m=1}^{M} w_m I(y_m(x) = c) \qquad \text{weighted majority (classification)}$$



← STACKING

CASCADING:



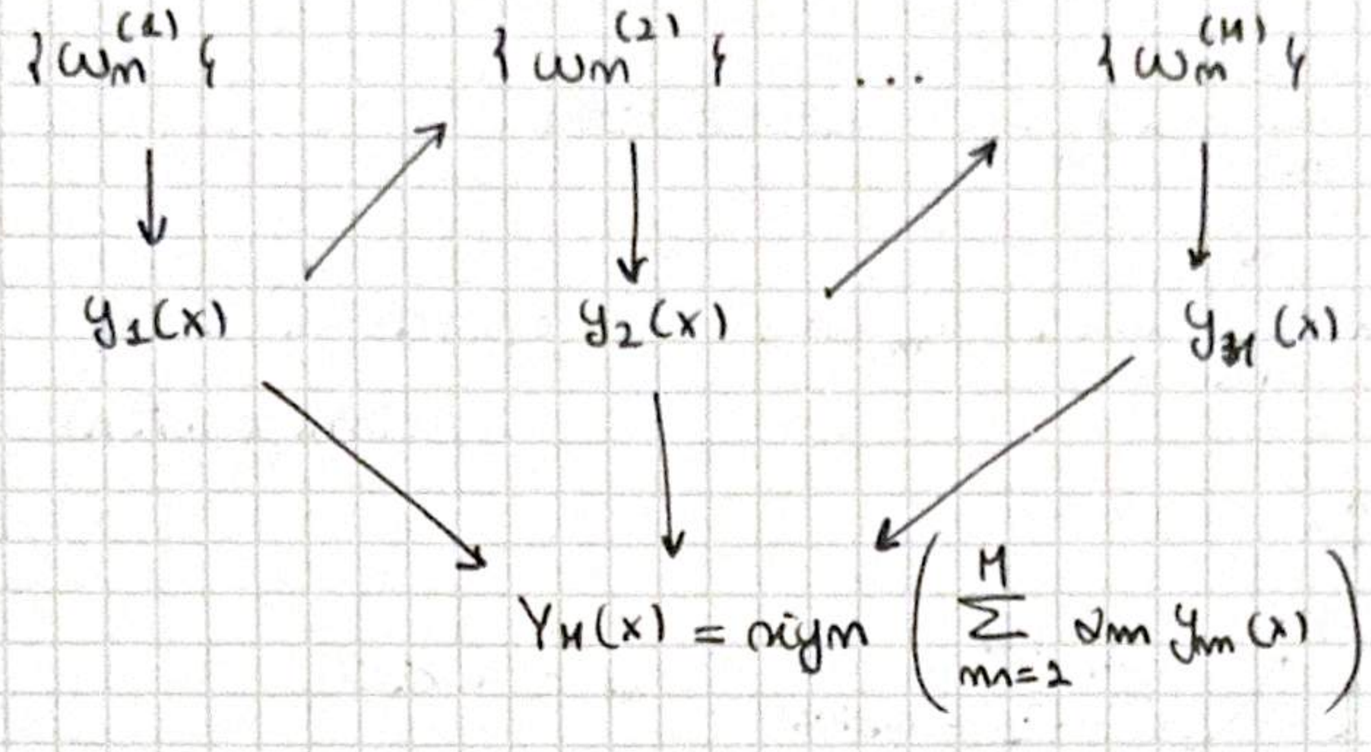
BAGGING

1. generate M bootstrap datasets $D_1, \ldots, D_M$
2. use them to train $y_m(x)$ model
3. make prediction with voting scheme

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

BOSTING:

~~we~~ ~~the~~ ~~use~~ train the first model and we ~~train~~ your weight for train the second model



ADA BOOST

1. train a week learner $y_m(x)$

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n) \qquad I(e) = \begin{cases} 1 & \text{if } e = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

2. evaluate $\varepsilon_m = \dfrac{\sum_{i=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{i=1}^{N} w_i^{(m)}}$

3. update $w_n^{(m+1)} = w_n^{(m)} e^{(\alpha_m I(y_m(x_n) \neq t_n)}$ $\qquad \alpha_m = \ln\left(\dfrac{1-\varepsilon_m}{\varepsilon_m}\right)$

4. $Y_M(x) = \text{sign}\left(\sum_{m=2}^{M} \alpha_m y_m(x)\right)$

SLIDES 12

WHY:

linear models cannot interaction between input variables

kernel method → require suitables methods

FNN → complex combination of many parametric functions

REGRESSION

linear units: $y = W^T h + b$

Gaussian distribution noise model: $p(t|x) = N(t|y, \beta^{-1})$

LOSS FUNCTION → MEAN SQUARED ERROR

OUTPUT UNITS : $y = \delta(w^T h + b)$

loss function: $J(\theta) = E_{x,t \sim D}[-\ln(p(t|x))]$ } binary classification

OUTPUTS UNITS: $y_i = \text{softmax}(a^{(l)}) = \frac{e^{a_i}}{\sum_j e^{a_j}}$

loss function: CATEGORICAL CROSS ENTROPY

↳ $J(\theta) = E_{x,t \sim D}[-\ln(\text{softmax}(a^{(l)}))]$

CHAIN RULE:

$y: \mathbb{R}^m \to \mathbb{R}^n$ and $f: \mathbb{R}^m \to \mathbb{R}$ $\partial z/\partial x_i = \sum_i \partial z/\partial y_i \; \partial y_i/\partial x_i$

FARWARD STEP

R: network step e

R: $W^{(i)}$, $i \in \{1, \dots, e\}$ weighted matrices

R: $b^{(i)}$, $i \in \{1, \dots, e\}$ bias parameter

R: x input value

R: t target value

$h^{(0)} = x$

for $k = 1, \dots, e$ DO

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

$h^{(k)} = f(a^{(k)})$

end for

$y = h^{(e)}$

$J = L(t, y)$

BACKWARD STEP

$g \leftarrow \nabla_y J = \nabla_y L(t, y)$

for $k = e, e-1, \dots, 1$ DO

propagate gradients to the pre-nonlinearity activations

$g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^{(k)})$ $\odot$ element wise product

$\nabla_{b^{(k)}} J = g$

$\nabla_{W^{(k)}} J = g (h^{(k-1)})^T$

propagate gradients to the next lower-level hidden layer

$g \leftarrow \nabla_{h^{(k-1)}} J = (W^{(k)})^T g$

end for

STOCHASTIC GRADIENT DESCENT

R: learning rate $\epsilon > 0$

R: Initial value of $\theta^{(1)}$

R

$k \leftarrow 1$

while stopping criterion not met do

SAMPLE SUB SET (mini batch) $\{x^{(1)}, \dots, x^{(m)}\}$ of m examples from D

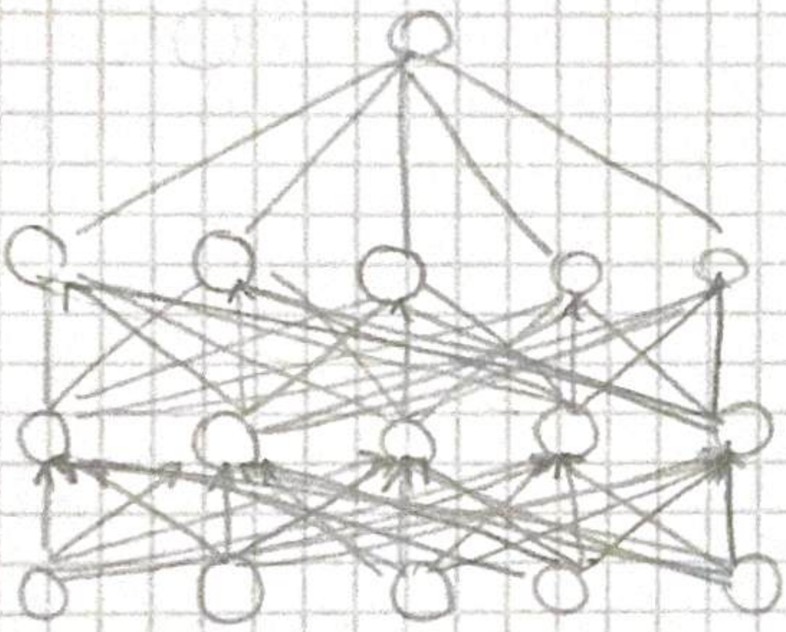
compute gradient estimate: $g = \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta^{(k)}), t^{(i)})$ ↓ backprop

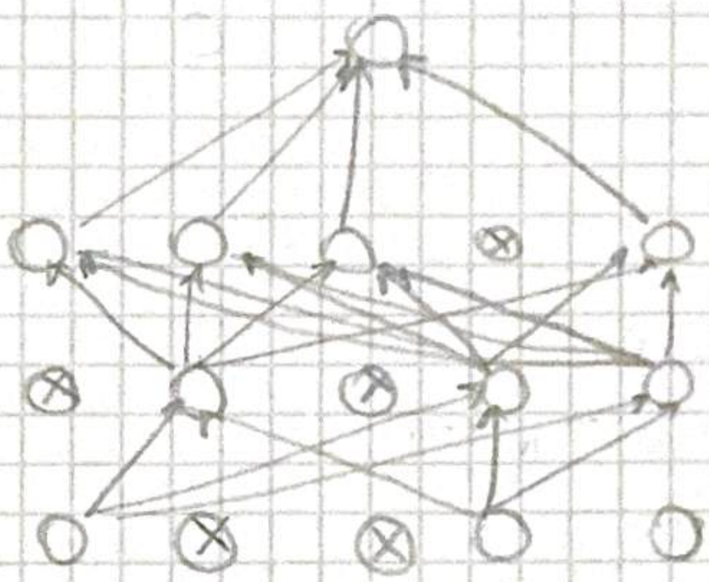
apply update $\theta^{(k+1)} \leftarrow \theta^{(k)} - \epsilon g$

$k \leftarrow k+1$

end while

DROP_OUT





SLIDES 13

CNN

CONVOLUTION: $(x * w)(t) \equiv \int_{a=-\infty}^{\infty} x(a) w(t-a) d(a)$ continuous

$(x * w)(t) \equiv \sum_{a=-\infty}^{\infty} x(a) w(t-a)$ discrete

2D convolution $(I * K)(i,j) = \sum_{m \in S_1} \sum_{n \in S_2} I(m,n) K(i-m)(j-n)$

cross correlation $(I * K)(i,j) = \sum_{m \in S_1} \sum_{n \in S_2} I(m+i, j+n) K(m,n)$

GENERAL

input: $w_{in} \times h_{in} \times d_{in}$

kernels: d_{out} of size $w_K \times h_K \times d_K$ $\quad d_K = d_{in}$

output: $w_{out} \times h_{out} \times d_{out}$

NUMBERS OF PARAMETERS stride s and padding p

consider input of size $W_{in} \times h_{in} \times d_{in}$, d_{out} kernel of size $W_K \times h_K \times d_{in}$

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 parameters:

$$|\theta| = \underbrace{W_K \cdot h_K \cdot d_{in} \cdot d_{out}}_{\text{kernel weights}} + \underbrace{d_{out}}_{\text{bias}}$$

SLIDES 14

INPUT DATA available $D = \{x_n\}$

GAUSSIAN MIXTURE MODEL (GMM): $P(x) = \sum_{k=1}^{K} \pi_k N(x, \mu_k, \Sigma)$

where π_k: prior probability Σ: covariance matrix

μ_k: mean

K-MEANS

STEP. 1 → decision value K = number of cluster

STEP. 2 → put initial partition that classifies the data into K cluster

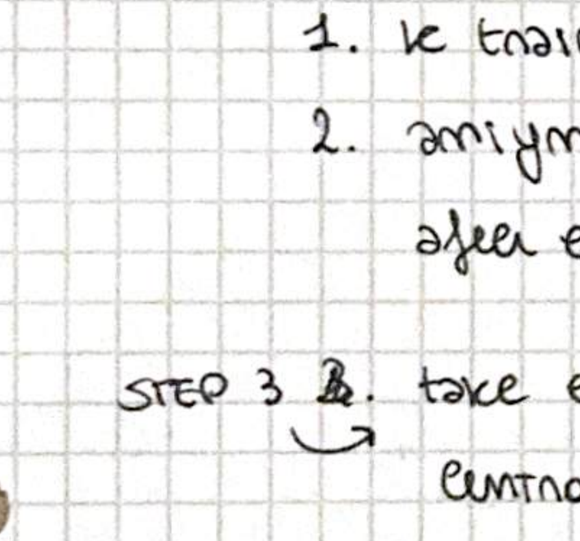
1. K training samples in single-element cluster
2. assign N-K to the cluster with nearest centroid

after each assignement recompute centroid

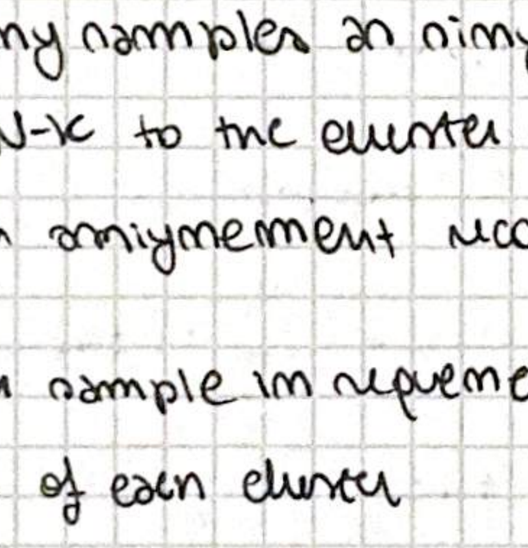
STEP 3 & ↳. take each sample in sequence and compute its distance from centroid of each cluster

IF NOT cluster → switch with that cluster and update centroids

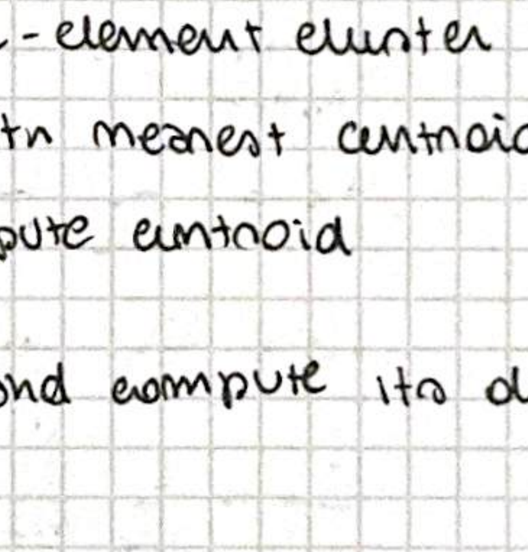
STEP 4 repeat until convergence



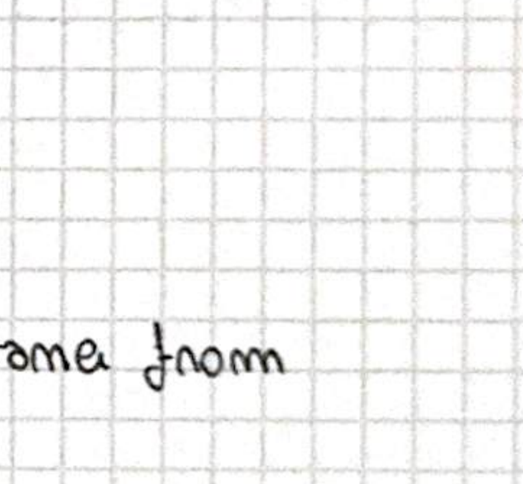
INITIALIZATION



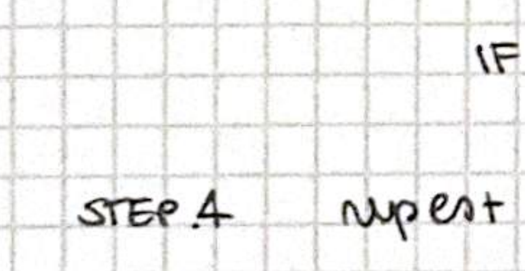
CREATE CLUSTER GIVEN THE MEAN



COMPUTE THE MEAN GIVEN THE CLUSTER



REPEAT UNTIL CONVERGENCE



LATENT VARIABLE

introduce $z_k \in \{0,1\}$ with $z = (z_1, \dots, z_K)^T$

$$P(z_k = 1) = \pi_k \quad \text{then} \quad P(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

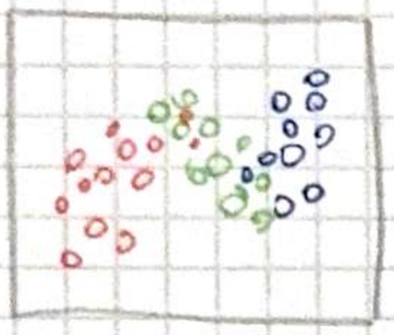
$$P(x|z) = \prod_{k=1}^{K} N(x; \mu_k, \Sigma_k)^{z_k}$$

$$P(x) = \sum_z P(z) P(x|z) = \sum_{k=1}^{K} \pi_k N(x; \mu_k; \Sigma_k)$$

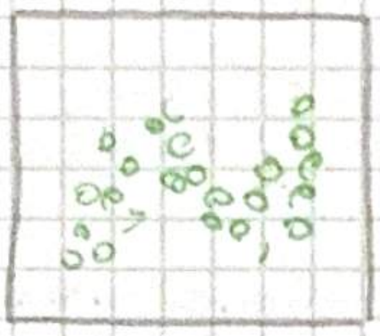
$$\gamma(z_k) = \frac{\pi_k N(x, \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x, \mu_j, \Sigma_j)}$$

π_k: prior probability

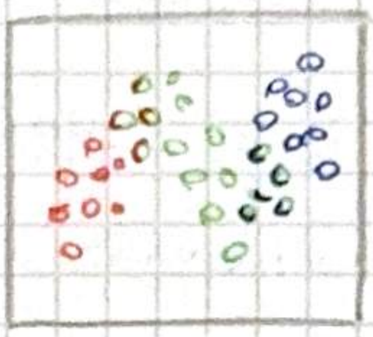
$\gamma(z_k)$: posterior probability after observation of x



$P(x,z)$ with 3 latent variable



$P(x)$ marginalized distribution



$\gamma(z_n, k)$ POSTERIOR DISTRIBUTION

EXPECTATION MAXIMIZATION (EM)

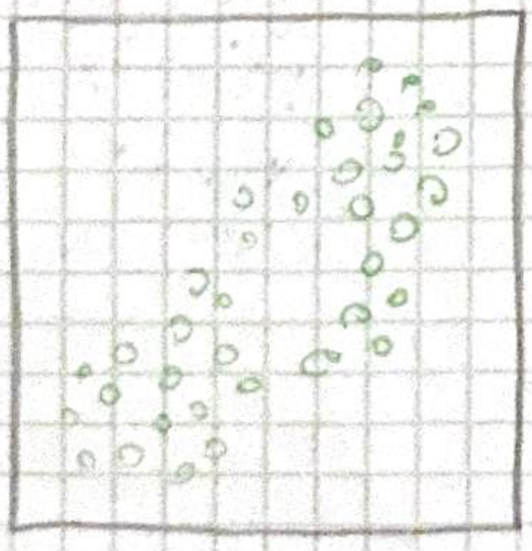
MAXIMUM LIKELIHOOD: $\underset{\pi,\mu,\Sigma}{\arg\max} \ln(P(X|\pi,\mu,\Sigma))$

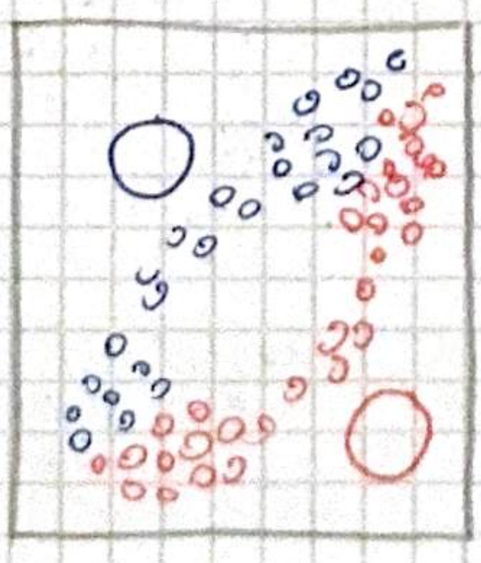
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

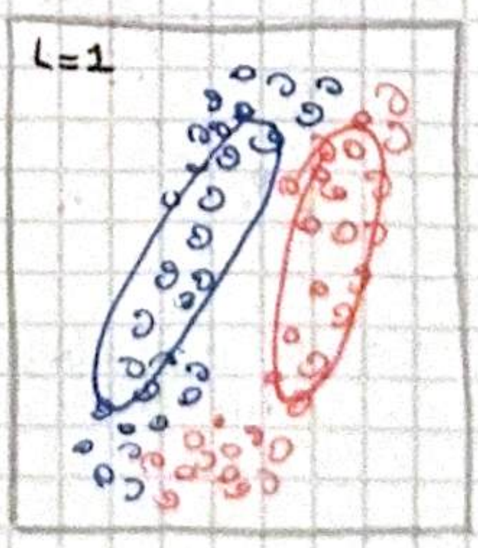
$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

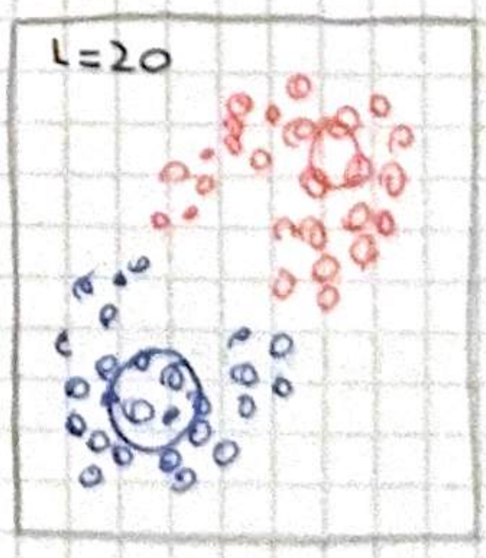
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^T$$

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









SLIDES 15

MEAN OF PROJECTED POINTS : $\bar{x}^T \mu_1$

VARIANCE OF PROJECTED POINTS: $\frac{1}{N} \sum_{n=1}^{N} [\mu_1^T x_n - \mu_1^T \bar{x}]^2 = \mu_1^T S \mu$

with S (dxd) covariance matrix of the dataset

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

PCA VARIANCE MAXIMIZATION

$$\max_{\mu_1} \mu_1^T S \mu_1$$

equivalent to unconstrainted maximization with LAGRAGIAN MULTIPLIER

$$\max_{\mu_1} \mu_1^T S \mu_1 + \lambda_1 (1 - \mu_1^T \mu_1)$$

solution:

$S\mu_1 = \lambda_1 \mu_1$ μ_1 must be an eigenvectors of S

left multiplying μ_1^T we have $\mu_1^T S \mu_1 = \lambda_1$

variance is maximal when μ_1 is the eigenvector corresponding to the largest eigenvalue λ_1 (called principal component)

PCA ERROR MINIMIZATION

$$x_n = \sum_{i=1}^{d} \alpha_{ni} \mu_i \quad \text{where} \quad \alpha_{ni} = x_n^T \mu_i \quad x_n = \sum_{i=1}^{d} (x_n^T \mu_i) \mu_i$$

goal: approximate x_n using a lower dimensional representation

$$\tilde{x} = \underbrace{\sum_{i=1}^{m} z_{ni} \mu_i}_{\text{HIGHEST EIGENVALUE}} + \underbrace{\sum_{i=m+1}^{d} b_i \mu_i}_{\text{LOWEST EIGENVALUE}}$$

evaluate Retained MSE

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

minimizing z_{ni} and b_i

$$z_{ni} = x_n^T \mu_i \qquad b_i = \bar{x}^T \mu_i$$

using this representation:

$$x_n - \tilde{x}_n = \sum_{i=m+1}^{d} [(x_n - \bar{x})^T \mu_i] \mu_i$$

the overall approximation:

$$J = 1/N \sum_{n=1}^{N} \sum_{i=m+1}^{d} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=m+1}^{d} u_i^T S u_i$$

$$\tilde{J} = \sum_{i=m+1}^{d} u_i^T S u_i - \lambda_i (1 - u_i^T u_i)$$

setting derivative of $u_i = 0 \Rightarrow S u_i = \lambda_i u_i$

approximation error $J = 1/N \sum_{i=m+1}^{d} \lambda_i$

HIGH DIMENSIONAL DATA:

- centered data matrix X
- compute the $N-1$ eigenvalue λ_i and eigenvectors v_i of XX^T
- let $u_i = \frac{1}{\sqrt{N\lambda_i}} X^T v_i$

PROBABILISTIC PCA

$$P(x) = \int P(x|z) P(z) d(z) = N(x, \mu, C) \quad C = WW^T + \sigma^2 I$$

posterior distribution

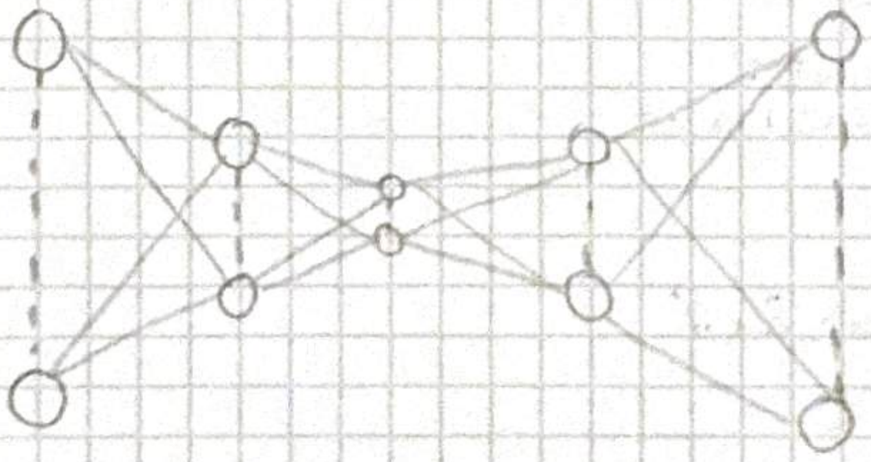
$$P(z|x) = N(z, M^{-1} W^T (x - \mu), \sigma^2 M) \quad M = W^T W + \sigma^2 I$$

maximum likelihood:

$$\underset{W,\mu,\sigma}{\operatorname{argmax}} \ln(P(x|W, \mu, \sigma^2)) = \sum_{n=1}^{N} \ln(P(x_n|W, \mu, \sigma^2))$$

AUTOENCODERS

reduce rate of hidden layers which learn to reconstruct their input by minimizing LOSS-FUNCTION



ANOMALY DETECTION:

LEARN $f: X \to \{n, a\}$ with dataset $D = \{(x_n, n)\}$ (labels are irrelevant so we can consider an UNSUPERVISED DATASET

GIVEN TEST $x' \notin D$ PREDICT $\hat{f}(x') \in \{n, a\}$ normal or abnormal

SOLUTION:

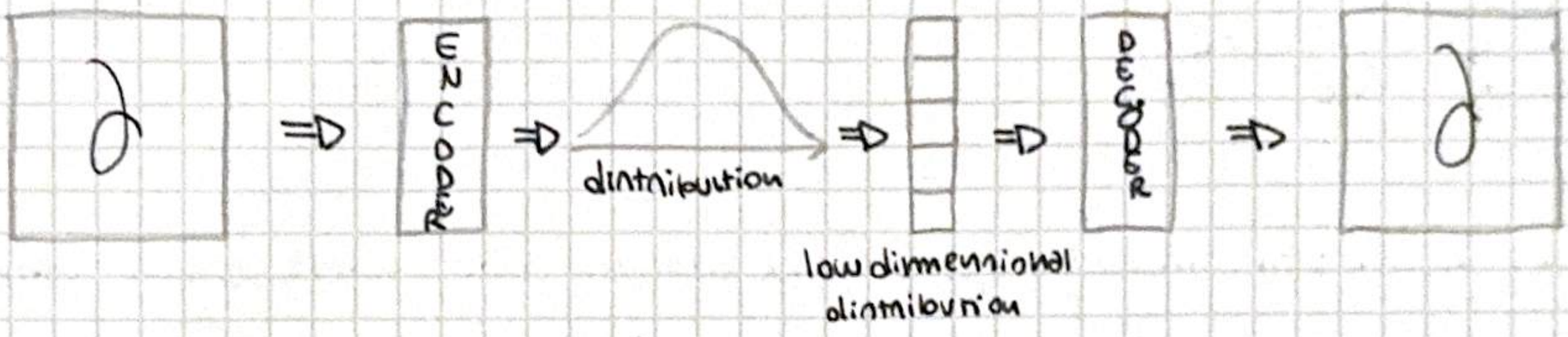
1. train AE with $\mathcal{D}_{train}$ compute final train loss
2. determine threshold δ = mean (loss) + std (loss)
3. given $x' \in \mathcal{D}$ reconstruct x' with AE and compute loss'
4. if loss' < δ → normal otherwise abnormal

GENERATIVE MODELS

- Variational AutoEncoders VAE → focus on learning latent space structure
  - modify data in specific direction
  - identify meaningful directions in latent spa

ENCODER → produces distribution instead of vector

DECODER → operate on sample on this distribution



PROBLEM → sampling operation NOT differentiable

↳ SOLUTION → RE-PARAMETRIZATION

- GAN Generative Adversarial Network
  - sample from the input data distribution X

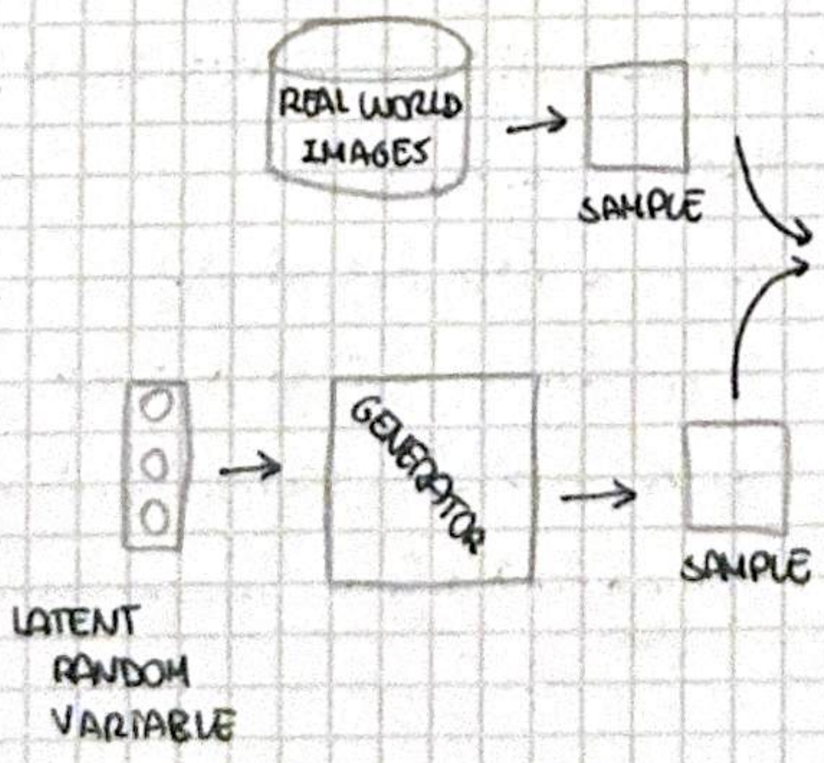
IDEA → invert convolutional neural network use adversarial training

ENCODER

- poses an image and produe an vector

DECODER

- receives a code and produces an images



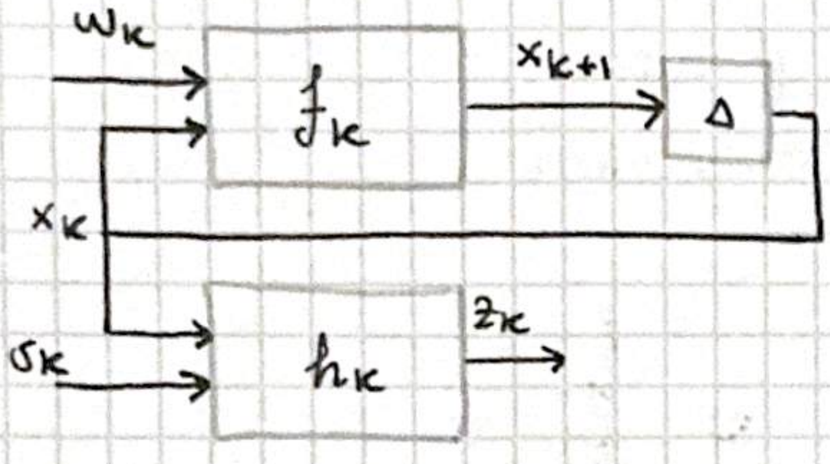
compete each other if loss generator up the discriminator goes down

TRAIN:

- train the discriminator with a batch of data $\{(x_n, \text{REAL}) \cup (\hat{x}_m, \text{FAKE})\}$ where x_n comes from the dataset while $\hat{x}_n$ is generated by GENERATOR with random value of the latent variable
- train the GENERATOR by using the entire model (generator + discriminator) with fixed layers (not trainable) $\{rx_k, \text{REAL}\}$ where rx_k are random values of LATENT VARIABLE

SLIDES 16

DYNAMIC SYSTEM



x: state — f: state transition model

z: observation — h: observation model

w, v: noise

REASONING: given the model (f, h) and current state predict the future

LEARNING: given past experience (z_0, k) determine the model

STATE FULLY OBSERVABLE $\rightarrow \pi : X \rightarrow A$ which action execute

MARKOV PROPERTY:

- once the current state is know, the evolution of dynamic system doesn't depend on the history of states, actions and observation
  - current state contain all information need to predict the future
  - future state are conditionally independent of past states and past observations given the current states
  - the knowledge of the current state makes past, present and future observations statistically indipendent

DETERMINISTIC TRANSITION:

MDP = $\langle X, A, \delta, rc \rangle$

X is finite set of states

A is finite set of actions

$\delta : X \times A \rightarrow X$ is transition function

$rc : X \times A \rightarrow \mathbb{R}$ is reward function

NON DETERMINISTIC TRANSITION:

MDP = $\langle X, A, \delta, rc \rangle$

X is finite set of states

A is finite set of actions

$\delta : X \times A \rightarrow 2^X$ is transition function

$rc : X \times A \times X \rightarrow R$ is reward function

STOCHASTIC TRANSITIONS:

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

X: set of states finite

A: finite set of actions

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

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

OPTIMAL POLICY

$\pi: X \to A$ for each $x \in X$ $\pi(x) \in A$ is the optimal action to be executed in such state

OPTIMALITY = MAXIMISE the CUMULATIVE REWARD

γ = discount factor

EXPECTED VALUE OF THE CUMULATIVE REWARD

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

↙ NON deterministic case

$$V^{\pi}(x_1) = \bar{r}_1 + \gamma \bar{r}_2 + \gamma \bar{r}_3 + \dots \to \text{deterministic case}$$

OPTIMAL POLICY $V^{\pi^*}(x_i) > V^{\pi}(x_i) \quad \forall i = \{1, \dots\}$

$r(a_i)$: DETERMINISTIC KNOWN

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

$r(a_i)$: DETERMINISTIC UNKNOWN

ALGORITHM:

1. for each $a_i \in A$
   - execute a_i and collect reward r_i

   |A| iterations are needed
2. OP: $\pi^*(x_0) = a_i$ with $i = \underset{i=1 \dots |A|}{\operatorname{argmax}} \; r_{(i)}$

$r(a_i)$: NON-DETERMINISTIC KNOW

OP: $\pi^*(x_0) = \underset{a_i \in A}{\operatorname{argmax}} \; E[r(a_i)]$

$r(a_i)$: NON DETERMINISTIC UNKNOWN

ALGORITHM:

1. initialite structure Θ
2. 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
3. optimal policy $\pi^*(x_0) = \dots$ according with data structure

   $T \gg |A|$ iterations

ε- GREEDY STRATEGY:

given $0 \leq \varepsilon \leq 1$

select an random action with probability ε

select best action with probability $1-\varepsilon$

ε decrease over time (FIRST exploration THEN exploitation)

SOFTMAX STRATEGIES:

actions with higher $\hat{Q}$ values are assigned higher probabilies but every action is assigned a non-zero probabilities

$$P(a_i|k) = \frac{k^{Q(x,a_i)}}{\sum_j k^{\hat{Q}(x,a_i)}}$$

$k>0$ select how atnomyly

the selection favors actions with high $\hat{Q}$

EXAMPLE k armed bandit

stochastic choice $r(a_i) = N(\mu_i, \sigma_i)$

ε-greedy strategies

training rule:

$$Q_n(a_i) \leftarrow Q_{n-1}(a_i) + \alpha [\bar{r} - Q_{n-1}(a_i)] \qquad \alpha = \frac{1}{1+V_{n-1}(a_i)}$$

$V_{n-1}(a_i)$ = number of execution

NOW consider also states X and state of evolution

$$Q^{\tilde{\pi}}(x,a) = r(x,a) + \gamma V^{\pi}(\delta(x,a))$$

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

OP: $\pi^*(x) = \underset{a \in A}{\operatorname{argmax}}\, Q(x,a)$

RECURSIVE FORMULA:

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

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

training rule

$$\hat{Q}(x,a) \leftarrow \bar{r} + \gamma \max_{a'} \hat{Q}'(x',a')$$

$\bar{r}$ is the immediate reward and x' is the state resulting from applying action a in x

NON DETERMINISTIC CASE

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

$$Q(x,a) = E[r(x,a)] + \gamma \sum_{x'} P(x'|x,a) \max_{a'} Q(x',a')$$

training rule:

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

$$\alpha = \alpha_{n-1}(x,a) = \frac{1}{1 + \text{visits}_{n-1}(x,a)}$$

TD TEMPORAL DIFFERENCE → OFF POLICY

n-steps time difference:

$$Q^{(n)}(x_t, a_t) = r_t + \gamma r_{t+1} + \dots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \hat{Q}(x_{t+n}, a)$$

using $0 \leq \lambda \leq 1$

$$Q^\lambda(x_t, a_t) \equiv (1-\lambda)[Q^{(1)}(x_t, a_t) + \lambda Q^{(2)}(x_t, a_t) + \lambda^2 Q^{(3)}(x_t, a_t) + \dots]$$

if $\lambda = 0$ Q-LEARNING

if $\lambda > 0$ emphasis on discrepancies based on more distant look heads

if $\lambda = 1$ only observed r_{t+i} are considered

SARSA → ON POLICY

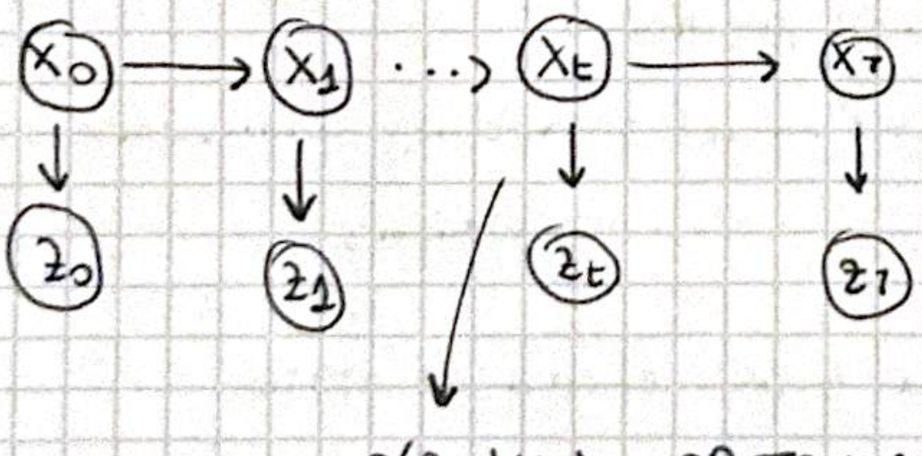
is based on tuple $\langle s, a, r, s', a' \rangle$

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

a' is choose based on current policy

SLIDES 17

STATE OF WORLD OF DYNAMIC SYSTEM IS NOT OBSERVABLE



I CAN ONLY OBSERVE z_t
(example robot with sensors)

$P(z_t|x_t)$ OBSERVATIONS CAN BE ALSO CONTINOUS

$P(x_t = \underbrace{1 \dots i}_{m} | x_{t-1} = \underbrace{1 \dots i}_{m})$ → can be discrete way → form a MATRIX

HMM Hidden Markov Model

- FILTERING: problem of estimating the state of system at time t given all the past observation
  ↓
  CURRENT STATE

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

- SMOOTHING: given all the observation given so far we want estimate what was the value of any state in the past

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

LEARNING in HMM

1. states can be observed at training time

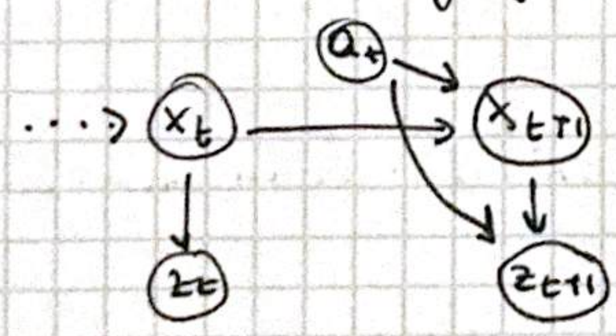
   transitions and observations models can be estimated with statistical analysis

2. states cannot be observed at training time

   compute local maximum like lihood with EM

POMDP Partially OBservable Markov Decision Process

combining decision making of MDP with HMM



POMDP $\langle X, A, \delta, r, Z, o \rangle$

X: finite states — $\delta(x, a, x') = P(x' \mid x, a)$ is probability distribution over transitions

A: finites actions — $r(x, a)$ is reward function

Z: finite set of observations — $o(x', a, z') = P(z' \mid x', a)$ probability distribution over observations

we need policy

- BELIEF STATES = probability distribution over the current state