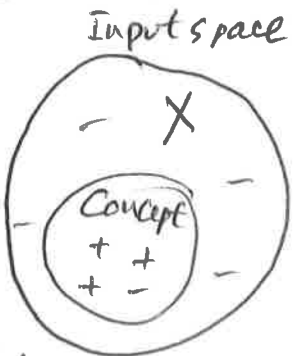
Lecture - II   Instance based learning



of KNN

Idea: i) Similar objects shares same concept
ii) Classify new examples like similar training examples.

Famous Similarity functions for Real value features:

1) Gaussian kernel $K(x', x) = e^{-(x'-x)}$ → depends how far is point x from x'

2) cosine $K(x', x) = \cos(x, x')$ → depends on angle of vector

Lecture - III   Decision Tree learning

Example object (comp, par, guess)	color	original	presentation	binder	A+
1					
2					
3					
4					

* Instance space = set of all possible objects described by attributes
  $|X| = 3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 48$
* Target function f: Maps each instance x to target label y. (hidden)
* Hypothesis h: function that approximates f
* Hypothesis space H: Set of functions we allow for approximating f
* Training Data S: Set of instances labeled with target function "f".

Q: How many functions $h: X \to Y$ exist? Ans $|H| = 2^{48}$

48 instances

	h_1	h_2	. . . . . .	h_n
(comp, y, y, c, y)	y	n		n
	y	y		n
⋮	⋮	⋮		⋮
(guess, no, no, unclear, no)	y	N		n

Just for 48 instances, we have 2^{48} hypothesis function. Therefore we should restrict hypothesis space.

Example we can restrict "h" that form conjunction P

$$h_p(x) = \begin{cases} \text{yes} & \text{if P is true} \\ \text{No} & \text{" " " false} \end{cases}$$

P = (correct = complete) ∩ (origin = yes)

correct	color	original	presentation	binder
complete	?	yes	?	?

Size of |H| for conjunctions $|H| = 4.3\,3.3\,3 = 324$
$+1 = 325$

i.e we need to search within 325 hypothesis functions instead of 2^{48}. This is syntactic

L#3 Prediction & overfitting

Learning Problem

$$P(X,Y) = P(Y|X)\,P(X)$$

STEP-1 P(X) is the "world" that produces data instance

Ex 1 X is homework description

$\vec{X_1}$ = (complete, yes, yes, clear, no) has some probability

$P(\vec{X_1}) = 0.2$

$\vec{X_2}$ = (guess, no, no, unclear, no)

$P(\vec{X_1}) = 0.0001$

Every instance ("48" instances) has some probability. We don't know what is that probability distribution

Ex2 Patient description ; Attributes (fever, age).

Step2 $P(Y|X)$ is "teacher" that labels X

(Before we were using function to label "x_i" but now its conditional distribution)

Ex1 Its A^+ homework ?

$$P(A^+ | x = \vec{x_1}) = 1 \longrightarrow P(\text{not } A^+ | x = \vec{x_1}) = 0$$

Its deterministic labeling.

Ex2 Develops complication in second example?

$$P(\text{complication} \mid X = (103, 5)) = 0.01$$
$$P(\text{complication} \mid X = (103, 90)) = 0.3$$

Its not deterministic.

$$P(S) = P(\underbrace{x_1, y_1, x_2, y_2 \ldots x_n, y_n}_{2n \text{ random variables}})$$

IID means :

$$P(S) = \prod_{i=1}^{n} P(X_i = x_i, Y_i = y_i)$$

i.e examples are independently drawn i.e sampling one sample would not effect sampling of other sample.

$$= \prod_{i=1}^{n} P(X = x_i, Y = y_i)$$

"identically distributed" i.e we can use same random variables for all samples. because they share same probability distribution.

If something changes overtime, then we will not have IID dataset.

$Err_P(h)$ is the probability of making an error $P(h(\vec{x}) \neq y)$ on randomly drawn example from $P(X,Y)$ (for 0-1 loss)

Overfitting: Hypothesis $h \in H$ overfits training data S if there exist $h' \in H$ such that $Err_S(h) < Err_S(h')$ and $Err_P(h) >> Err_P(h')$

Occam's Razor: Prefer the simplest hypothesis that fits data.

L#4

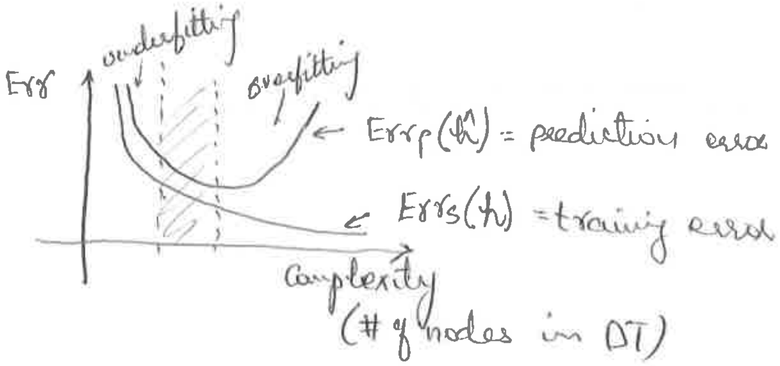
Model Selection And Assessment

(To prevent overfitting, we do model selection)

→ ~~~~ learning as prediction
- $P(x,y)$ is learning task
- i.i.d data $S = ((x_1,y_1) \cdots (x_n,y_n))$ drawn from $P(x,y)$
- Sample error $Err_S(h) = \frac{1}{n} \sum_{(x_i,y_i) \in S} \Delta(h(x_i), y_i)$
- Generalization Error $Err_P(h) = \sum_{(x,y) \in X} \Delta(h(x),y)\, P(X=x_i, Y=y_i)$

  It is not observed because $P(X=x_i, Y=y_i)$ is unknown

— Overfitting



In this lecture, we will talk to identify sweet spot in above curve, i.e in the middle of underfitting and overfitting.

L#5

Model Selection

Determine the parametre of learning algorithm
- In K.NN, we will find value of K, similarity measure using model selection
- Decision tree: depth, splitting criterion and many other.

→ In leave-one-out validation, we take $(n-1)$ examples & test it on the n^{th} example. We repeat this process "n" times and get unbiased estimator of prediction error. (42:00)

* what is the prediction error of h?

<u>Scenario</u> Is spam filtering rule h accurate enough $(Err_p(\hat{h}) \le 0.1)$

→ Binomial D's

$p = Err_p(h)$

n = no. of ex

X = number of errors

observation test error $Err_p(\hat{h}) = \frac{10}{500} = \frac{X}{n}$

Is there significant evidence that $Err_p(\hat{h}) \le 0.1$?

<u>Hypothesis test</u>:

Null hyp $p \ge 0.1$ (Generalization error $\ge 10\%$)

what is the probability to see at most 10 errors under null hypothesis?

OR $P(X \le 10 \mid p = 0.1, n = 500)$

$P(x \le 10 \mid 0.1, 500) = 1.1 \times 10^{-12}$ $\boxed{\le 0.05}$ This is famous threshold used in p-test

If null hypothesis is true, then it is extremely unlikely to see errors less than or equal to 10. Thus we reject null hypothesis.

L#6 <u>Linear Classifiers and Perceptrons</u>

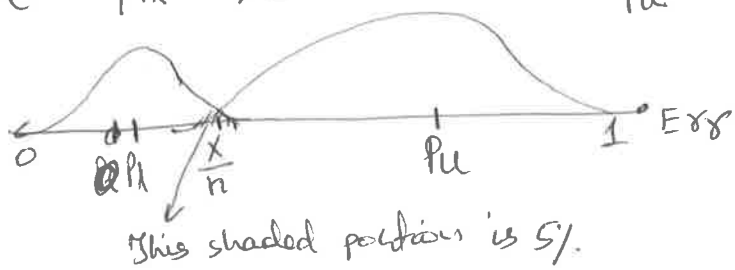
Perceptrons = invented by Rosenblatt from Psychology (at the same time in Soviet during 1960's).

What range does the true error rate lie in $[\underbrace{Err_p(\hat{h})}_{\text{true error rate}}]$?

→ Binomial confidence intervals

Given x errors on test set, what values $Err_p(\hat{h}) \in [p_l, p_u]$ could have plausibly (say 95% confidence) generated X

i.e $\min_{p_l} P(X \ge x \mid p_l, n) \le 0.025$ and $\max_{p_u} (X \le x \mid p_u, n) \le 0.025$



x = # of errors, we divide it by n so that our error ranges from 0 to 1

Example:

→ $n=40$ test examples, $Err_{test}(\hat{h}) = \frac{9}{40}$

→ Normal confidence interval

$$E(X) \in 9 \pm 1.96\sqrt{Var(x)}$$

We will use estimated variance

$$Var(X) = 40 \times \frac{9}{40}\left(1 - \frac{9}{40}\right)$$

$$Var(X) = 14.176$$

→ 95% normal confidence interval with estimated variance $Var(x)$

$$3.824 \leq np \leq 14.176$$

Divide by "n"

$$0.0956 \leq Err_p(\hat{h}) \leq 0.354$$

After seeing 9 test errors over 40 instances, we can conclude our test error lies in the ~~normal~~ confidence interval [0.0956, 0.354].

→ 95% exact binomial confidence interval

$$0.1084 \leq Err_p(\hat{h}) \leq 0.3844$$

McNemar Test

	(x_1, y_1)	(x_2, y_2)	(x_u, y_u) ...	(x_N, y_N)
h_1	✓	X	✓	X
h_2	✓	X	f	✓

d_1: no. of examples h_1 makes error h_2 correct

d_2: " " h_2 " " h_1 "

$Err_p(\hat{h}_1) = Err_p(\hat{h}_2) \Rightarrow E(d_1) = E(d_2) \Rightarrow d_1, d_2 \sim \text{Binomial}(p=0.5, n=d_1+d_2)$.

Example

1000 test examples

h_1 makes 101 error

h_2 makes 111 errors

$d_1 = 1$

$d_2 = 11$

$P(D_1 \leq 1 \mid p=0.5, n=12) = 0.003 \leq 0.5$ (Reject H_0)

* Very simple illustration of (Batch) Perceptron Algorithm. L#6 (50:00)
If the data is linearly separable, we can always find separating hyperplane using perceptron ~~to~~ algorithm.

L#7 Online Learning and Perceptron Mistake Bound

Worst-case Mistake Bound:

Given set C of example sequences and learning algorithm "L", what is the maximum number of mistakes that algorithm "L" ~~makes~~ with hypothesis space "H" makes on any $S \in C$ before learning a perfect rule? Ans: $\frac{R^2}{\delta^n}$
classification

→ Implicit Assumption: it exists $h \in H$ with zero error for all set $S \in C$.

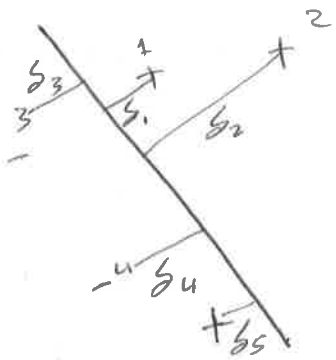
Margin (δ)

- If margin of example is positive then correct
- Geometric Intuition

Geometric margin is the signed distance of example to hyperplane.

$$d(h_{\vec{w},b}, \vec{x}) = \frac{1}{\|w\|} |wx+b|$$

- $\delta_{geometric} = \delta_{functional} \cdot \frac{1}{\|w\|}$



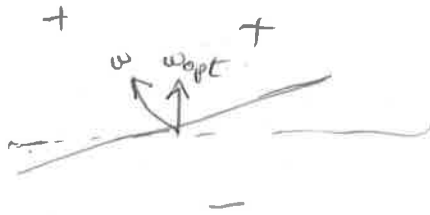
In perceptron, scaling $\|w\|$ will have no effect while classification,

Unbiased means there is no "b" in $\|w.x+b\|$.

To show: perceptron finds hyperplane close to w_{opt}

$\iff$ angle (w_{opt}, w_{K+1}) is small

$\iff$ $\cos(w_{opt}, w_{K+1}) = \frac{w_{opt} \cdot w_{K+1}}{\|w\|_{K+1} + \|w_{opt}\|}$ is almost 1.



$\rightarrow$ find lower bound on cos

a) lower bound on numerator : $w_{opt} \cdot w_{K+1}$

b) Upper bound on denominator : $\|w\|_{K+1}$. we already know that $\|w_{opt}\| = 1$.

Proof by contradiction

If for K, we can show that lower bound on the $\cos(w_{opt}, w_{K+1}) > 1$, then perceptron can not make K+1 errors.

a) Induction: $w_{K+1} \cdot w_{opt} \geq f(w_K \cdot w_{opt})$.

$$w_{K+1} \cdot w_{opt} = (w_K + y_i x_i) w_{opt}$$

$$= w_K \cdot w_{opt} + w_{opt} y_i \cdot x_i$$

$$= w_K \cdot w_{opt} + \underbrace{y_i \cdot x_i \cdot w_{opt}}_{\geq \delta}$$

$$\geq w_K \cdot w_{opt} + \delta$$

~~overall $w_{K+1} \cdot w_{opt} \geq$~~

$\rightarrow$ $w_K \cdot w_{opt} \geq K \cdot \delta$

b) Induction: $\|w_{K+1}\|^2 \leq g(\|w_K\|^2)$

$$\|w_{K+1}\|^2 = \| w_K + y_i x_i \|^2$$

$$= w_K \cdot w_K + 2 y_i x_i \cdot w_K + y_i x_i \cdot y_i x_i$$

$$= w_K \cdot w_K + \underbrace{2 y_i x_i \cdot w_K}_{\leq 0} + \underbrace{x_i \cdot x_i}_{\leq R^2}$$

b/c we are making update if $y_i (w_K \cdot x_i) \leq 0$

$$\leq w_K w_K + R^2$$

$$\|w_K\|^2 \leq K \cdot R^2$$

Putting together

$$1 \geq \frac{w_K \cdot w_{opt}}{\|w_K\|\|w_{opt}\|} \geq \frac{\delta \cdot K}{(\sqrt{R^2 \cdot K}) \cdot 1} = \frac{\delta\sqrt{K}}{R}$$

Solve for K:

$$K \leq \frac{R^2}{\delta^2}$$

→ Perceptron can not make more than $\frac{R^2}{\delta^2}$ updates/errors.

This bound is independent of no. of features.

Remarks

- Perceptron mistake bound is independent of N (# of features).
- Independent of ordering
- Mistake bound is independent of scaling and rotation of data.
- same for actual behavior of perceptron
- Downside is that this bound is valid for separable datasets.

L#8 Ensemble methods: Bagging

L#9 Ensemble methods: Boosting

Ensemble method: A class of "meta" learning algorithms because it combines set of classifiers to produce good classification.

* KNN with K=1 is a learning algorithm because a learning algorithm takes training set as an input and produces $h(x)$. If you change training set with K=1, you will have different decision boundary. Hence ~~KNN~~ KNN with K=1 is learning algorithm.

* $E[X+Y] = E[X] + E(Y)$
* $E[KX] = K\,E[X]$
* $E[X \cdot Y] = E[X]E[Y]$ if $X \perp Y$
* Variance:
$$E[(X - E[X])^2] = \underbrace{E[X^2]}_{\text{second moment of } x} - (E[X])^2$$

$$\underset{\text{over subset of training sets}}{E_S}[(\overset{\text{fixed}}{y_i} - h_i(x_i))] = E_S[y_i^2 - 2y_i h_s(x_i) + h_i^2(x_i)]$$

$$= E_S[y_i^2] - 2E[y_i h_s(x_i)] + E_S[h_i^2(x_i)]$$

y_i is constant b/c it will have no label noise

$$= y_i^2 - 2y_i E[h_s(x_i)] + E_S[h_i^2(x_i)]$$

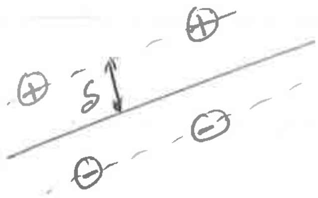
$$= \underbrace{y_i^2 - 2y_i E[h_s(x_i)]}_{(y_i - E[h_i(x_i)])^2} + \underbrace{E_S[(h_i(x_i) - E_S[h_i(x_i)])^2]}_{\text{Variance}}$$

Bagging: You create bag of training set and train set of classifiers over them.

$$h_S = \frac{1}{T}\sum_{i=1}^{T} \alpha_S h_i(x)$$

During test time, we ~~averaged~~ get classification by linear combination $h_i(x)$.

L#10 SVM: Optimal hyperplane & soft margin



Compute the optimal hyperplane:

Training sample $S=\{(x_1,y_1)\cdots(x_n,y_n)\}$

① $\max\limits_{(w,b)} \delta$ with $\delta = \min\limits_{(x_i,y_i)\in S}\left\{\frac{y_i(w.x_i+b)}{\|w\|}\right\}$

Geometric dist $= \frac{y_i}{\|w\|}(w^Tx+b)$

Simplify ② Write min interms of constraints

$$\max_{w,b} \delta \text{ with } \forall (x_i,y_i)\in S: \frac{y_i}{\|w\|}(wx_i+b)\ge\delta$$

③ normalize $\|w\| = \frac{1}{\delta}$, since ~~closest~~ classifier is invariant to $\|w\|$

$$\underset{w,b}{\text{maximize}} \ \frac{1}{\|w\|} \text{ with } \forall (x_i,y_i)\in S \ \frac{y_i}{\cancel{\|w\|}}(wx_i+b)\ge\frac{1}{\cancel{\|w\|}}$$

④ minimize $\|w\|$ s.t $y_i(wx_i+b)\ge 1$

⑤ minimize $\|w\|^2$ s.t

$\forall (x_i,y_i)\in S \quad y_i(wx_i+b)\ge 1$

L#10 (continued)

Soft-Margin SVM

* Slack variables at solution of QP:

$$\xi_i \geq 1 \iff y_i(w^T x_i + b) \leq 0 \text{ (error)}$$
$$0 \leq \xi_i < 1 \iff 0 < y_i(w^T x_i + b) < 1 \text{ (correctly classified but inside margin)}$$
$$\xi_i = 0 \iff y_i(w^T x_i + b) \geq 1 \text{ (correct with sufficient margin)}$$

Claim: $\sum \xi_i$ is an upper bound to number of training error.

Opt:

$$\frac{1}{2}\|w\|^2 + C\sum \xi_i$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

HP1: Error $= 2$, $\sum \xi_i = 8$, $\delta = \frac{1}{\|w_1\|} = \frac{1}{\sqrt{2}}$

HP2: Error $= 0$, $\sum \xi_i = 0$, $\delta = \frac{1}{\|w_2\|} = \frac{1}{\sqrt{4}} = \frac{1}{2} = 0.5$

HP3: Error $= 0$, $\sum \xi_i = 0$, $\delta = \frac{1}{\|w_3\|} = \frac{1}{\sqrt{3}}$

HP4: Error $= 0$, $\sum \xi_i = 2$, $\delta = \frac{1}{\|w_4\|} = \frac{1}{\sqrt{1/2}}$

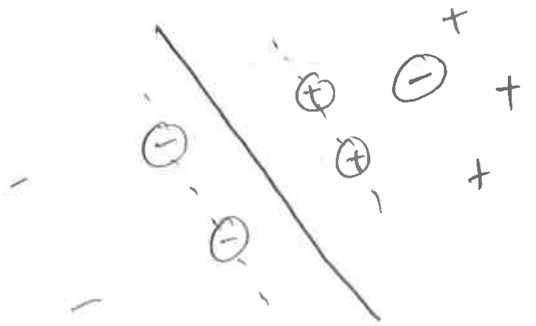
HP5: Error $= 0$, $\sum \xi_i = 0$, $\delta = \frac{1}{\|w_5\|} = \frac{1}{\sqrt{2}}$

HP6: Error $= 0$, $\sum \xi_i = 0$, $\delta = \frac{1}{\|w_6\|} = \frac{1}{\sqrt{1.8}}$

Intuition of SVM Margin: It puts weight on those features of $\vec{w}$ that are decisive for classification.

L#11 : Duality and Leave-one-out

- Support vectors have functional margin ≤ 1
- Margin is reasonable criterion for assigning weights based on evidence.



All training points in circles are support vectors.

If $n=10$ in (Batch) perceptron Algorithm

$$w_{final} = \alpha_1 y_1 x_1 + \alpha_2 y_2 x_2 + \cdots + \alpha_{10} y_{10} x_{10}$$

α = no of times perceptron makes update over x_i, y_i

$$= \sum_{i=1}^{n} \alpha_i y_i x_i$$

Theorem: The w_{final} by perceptron is always the linear combination of training examples

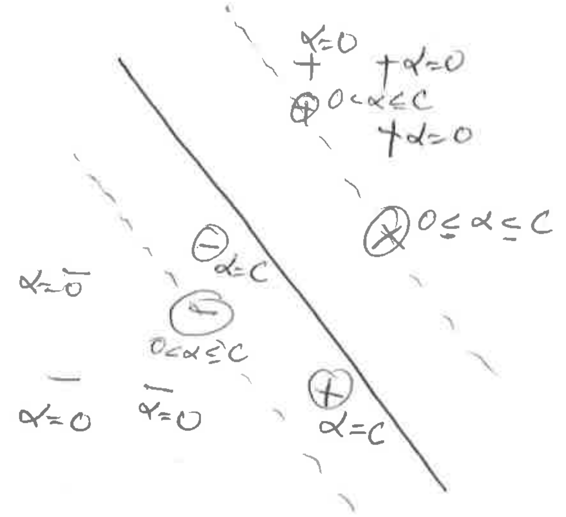
We should check primal & dual perceptron's equivalence

$$(w \cdot x_j) = \left(\sum_{i=1}^{n} \alpha_i y_i x_i \right) \cdot x_j$$

$$= \sum_{i=1}^{n} \alpha_i y_i (x_i^T x_j)$$

There are two different representation of linear classifier:

- Primal : weight vector "w" & threshold "b"
- Dual : training examples $(x_1, y_1) \cdots (x_n, y_n)$ and dual coefficient $\alpha_1 \ldots \alpha_n$.



$\alpha=0$ are those traing examples that do not disturb decision boundary.

Dual optimization

- $(\underbrace{\#}_{\vec{w}}$ of features $+ \overset{b}{1}$ param + # of $\overset{\xi_i}{\text{examples}})$ free variables in Primal
- $(\underbrace{\text{\# of examples}}_{\text{vector of } \alpha})$ free variables in the dual.
- It is convex & QP
- Training examples enter dual only through ~~in~~ inner product.

In dual, $\sum_{i=1}^{n} \alpha_i y_i = 0$ when we have "b" in primal

Corollary: If $P(w^*, b^*, \xi^*)$ is solution of primal & $D(\alpha^*)$ is the solution of dual, then $D(\alpha^*) = P(w^*, b^*, \xi^*)$

Leave-one-out error

Lets train SVM for whole traing set. Then we know:

Lemma

If $(h_i(x_i) \neq y_i)$ means $2\alpha_i R^2 + \xi_i \geq 1$

$\left.\begin{matrix}\alpha_i \\ \xi_i\end{matrix}\right\}$ from full traing set traing of SVM

$R \geq \|x_i\|$

As a result, we don't need to ~~ret~~ retrain those examples whos $2\alpha_i R^2 + \xi_i < 1$. An immediate case would be for those examples that have $\alpha=0$. If we remove those examples, they decision boundary would be unchanged & we dont need to retrain our classifier.

<u>Theorem</u>: $Err_{leave(S)}(h_{SVM}) \leq \dfrac{\# SV}{\# \text{ of training examples}}$

i.e leave one out error is bounded by number of support vectors.

<u>Bound on the expected generalization error</u>

Lemma (unbiased): If (x_i, y_i) is leave-one-out error then $\alpha_i R^2 \xi_i \geq 1$

<u>Assumption</u>: 1) we have hard margin $\longrightarrow \xi_i = 0$

If $\Delta(h_i(x_i), y_i) = 1$, then $\alpha R^2 \geq 1$

$\Rightarrow Err's_{leave}(h_{SVM}) \leq \underbrace{\frac{1}{m} |\# \alpha_i R^2 \geq 1| \leq \frac{1}{n} \sum_{i=1}^{n} \alpha_i R^2}_{\text{check it}}$

$h_i(x_i)$ = classifier retrained by leaving example "i" out.

$= \dfrac{R^2}{m} \sum_{i=1}^{n} \alpha_i$ $\quad \boxed{\sum_{i=1}^{1} \alpha_i = w^T w}$ in Hard margin case

$= \dfrac{1}{n} R^2 \cdot \|w\|^2$

$= \dfrac{1}{n} R^2 \cdot \dfrac{1}{\delta^2}$ $\quad \delta$ = geometric margin

$= \dfrac{R^2}{n \delta^2}$ (The # of leave-one-out error is upper bounded by $\frac{R^2}{\delta^2}$ which is exactly same as perceptron algorithm)

Lemma [Luntz]: The leave-one-out-error is almost an unbiased estimate of the generalization error.

$$\underset{\substack{S \text{ with } (n-1) \\ \text{examples}}}{E} [\text{True error of } Alg] = \underset{\substack{S \text{ of size} \\ \text{"}n\text{" example}}}{E} [Err_{\text{leave one out}} Alg]$$

Theorem If learning task $P(x,y)$ is separable by unbiased hyperplane, then the expected generalization error of an unbiased hard margin SVM is bounded by

$$\underset{\text{with } |S|=n-1}{E_S} [Err_P(h_{SVM})] \leq \frac{1}{n} \underset{\substack{S \text{ with} \\ \text{size}=n}}{E} \left[\frac{R^2}{\delta^2}\right]$$

where δ = sample margin of hyperplane computed by SVM.

L#12 SVM: Kernels

Review:

- Duality for linear classifiers
  - Primal: w, b ($N+1$ Params) $N \to$ # of features
  - Dual: α, b ($n+1$ Params) $n \to$ # of examples
  - $\Leftrightarrow w = \sum \alpha_i y_i x_i$ (connect primal & dual)
- DUAL Perceptron: $\alpha \in \mathbb{Z}^+$
- DUAL SVM: $\alpha \in \mathbb{R}$
- Leave-one-out
  - leave-one-out-error $\Rightarrow 2\alpha_i R^2 + \xi_{S_i} \geq 1$
  - $Err_{loo}(h_{SVM}) \leq \frac{1}{n} \frac{R^2}{\delta^2}$
  - $Err_{loo}(h_{SVM}) < \frac{\#SV}{n}$

Mapping $\phi: X_I \longrightarrow X_F$

Input Space: X_I with dimensionality N_I (low)

Feature Space: X_F with " N_F (high)

$\rightarrow a, b, c$ (3D) $\longrightarrow a, b, c, \underbrace{aa, ab, ac, bb, bc, cc}_{\text{monomials of 2 term}}$ (9D)

$$w = \begin{pmatrix} -0.7 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad b = 0 \xrightarrow[\text{into 2-attribute space}]{\text{Project it back}} h(x) = \text{sign}\left(\underbrace{-0.7x_1^2 + x_2}_{\text{we will see that we are actually learning non-linear classifier}}\right)$$

6-attributes problem on slide 5

* # of monomials of N_i features and degree less or equal "d" is $= \binom{N_i + d}{d}$

* $\phi(a) \cdot \phi(b) = \phi\left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}\right) \cdot \phi\left(\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)$

$$= \begin{pmatrix} a_1^2 \\ a_2^2 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \sqrt{2}a_1a_2 \\ 1 \end{pmatrix} \begin{pmatrix} b_1^2 \\ b_2^2 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \sqrt{2}b_1b_2 \\ 1 \end{pmatrix}$$

$$= a_1^2b_1^2 + a_2^2b_2^2 + 2a_1b_1 + 2a_2b_2 + 2a_1a_2b_1b_2 + 1$$

$$= (a_1b_1)^2 + 2(a_1b_1a_2b_2) + (a_2b_2)^2 + \cdots +$$

$$= (a_1b_1 + a_2b_2 + 1)^2$$

$$= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} + 1\right)^2$$

$$= (a \cdot b + 1)^2$$

i.e computing inner product of $\phi(a) \cdot \phi(b)$ can actually be obtained by $(a \cdot b + 1)^2$

$\forall N_I \forall d$: $\phi(a)\phi(b) = (a \cdot b + 1)^d$ → hold for any d

Kernel: $K(a,b)$

* $\boxed{\text{Kernel is like a similarity measure.}}$

* SVM with Kernel

How to compute b?

- Pick some (x_i, y_i) with $0 < \alpha_i < C$
- We know that $1 = y_i(w \cdot \phi(x_i) + b) = y_i\left[\left(\sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j)\right) + b\right]$
- Solve for b:

$$b = y_i - \sum_{j=1}^{n} \alpha_i y_i K(x_i, x_j)$$

- If we (x_i, y_i) whose $C = 1$, then its margin will be less than 1.

* $\|\phi(a) - \phi(b)\| = \sqrt{(\phi(a) - \phi(b))^2} = \sqrt{\phi(a)\phi(a) - 2\phi(a)\phi(b) + \phi(b)\phi(b)}$
$= \sqrt{K(a,a) - 2K(a,b) + K(b,b)}$

i.e Any linear learning algorithm that I can express interms of distances or interms of inner products, I can just plug in kernel and make it non-linear.

* Any boolean function can be represented by ~~boolean~~ kernels as well.

L#13 Learning ranking functions with SVMs

Why kernel?
- make linear learner non-linear
- make linear learner with non-vectorial data

what is kernel?
- ~~Itsea~~ kernel correspond to inner product in feature space
i.e $K(a,b) = \phi(a)\,\phi(b)$
- Kernel is efficient to compute even if $\phi(a)$ is inefficient.

Constructing kernel?
- Construct $\phi(a)$, then find kernel $K(a,b)$.
- Check that Gram matrix is always PSD & symmetric.
- Construct it from simple kernels.

"Kernelize" a learning algorithm.
- replace inner product by kernel.

* For ranking, we learn utility function.

L#14 Discriminative Vs Generative learning

Course So far
- Decision Trees learning
- ~~greedy~~ greedily construct small tree with low training error
- Problem: no theory

- Perceptron
- find separating hyperplane
- mistake bound
- Problem: should be no noise

- SVM
- soft margin
- theory about prediction error
- non linear / non-vectorial data via kernels.
- learning to rank

Uptill now, we have hypothesis space & we are searching for hypothesis that have smallest training error (empirical risk minimization) or more generally discriminative method.

Discriminative Training of Linear Rules

$$\min_{w,b} \underbrace{R(w)}_{\text{Regularizer}} + \underset{\text{Regularization Parameter}}{C} \underbrace{\sum_{i=1}^{n} \Delta(w^T x_i + b, y_i)}_{\text{training loss / empirical risk / Training error}}$$

- Soft Margin SVM

  $R(w) = \frac{1}{2}\|w\|^2 = 0.5\, w^T w$

  $\Delta(\bar{y}, y_i) = \max(0, 1 - y_i \bar{y})$ [margin = 1]

- Perceptron

  $R(w) = 0$

  $\Delta(\bar{y}, y_i) = \max(0, -y_i \bar{y})$ [margin = 0]

- Linear Regression

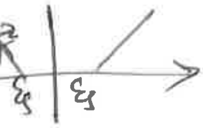
  $R(w) = 0$

  $\Delta(\bar{y}, y_i) = (y_i - \bar{y})^2$

- SVM Regression

  $R(w) = \frac{1}{2} w^T w$

  $\Delta(\bar{y}, y_i) = \cancel{(y_i - \bar{y})^2}$



- Ridge Regression

  $R(w) = \frac{1}{2} w^T w$

  $\Delta(\bar{y}, y_i) = (y_i - \bar{y})^2$

- Lasso

  $R(w) = \frac{1}{2} \sum |w_i|$

  $\Delta(\bar{y}, y_i) = (y_i - \bar{y})^2$

- Regularized Logistic Regression / Conditional Random Field

  $R(w) = \frac{1}{2} w^T w$

  $\Delta(\bar{y}, y_i) = \log(1 + e^{-y_i \bar{y}})$

$\boxed{\text{Continue from 31:00}}$

Example: X is patient in hospital

$$P(y=1 \mid X=x) = 0.7$$
$$P(y=-1 \mid X=x) = 0.3$$

→ Predict $+1$ has error of 0.3

→ If we predict -1, we will have an error of 0.7.

Bayes Rule: $h_{bayes}(\vec{x}) = \underset{y \in Y}{\operatorname{argmax}}\; P(Y=y \mid X=\vec{x})$

Bayes Error Rate

What is $Err_P(h_{Bayes})$?

- Given an instance X, h_{bayes} has prob of making an error of
$$\min_y \left[1 - P(Y=y|X=x)\right].$$
- Average over all X:
$$Err_P(h_{bayes}) = \sum_{x \in X} \underbrace{P(X=x)}_{known} \min_y \left(1 - P(Y=y|X=x)\right)$$

This is Bayes optimal error rate.

As $P(X,Y) = P(Y|X)P(X)$: we know that $P(X,Y)$ is our learning task and we ~~can't~~ dont know ~~the~~ this distribution. For instance in ~~text~~ text classification task, $P(X,Y)$ is the distribution of authors writing documents.

Generative vs Discriminative Models

$$\boxed{\begin{array}{l}\text{Process:}\\ \bullet\ \text{Generator: Generate descriptions according to distribution } P(X)\\ \bullet\ \text{Teacher: Assigns a value to each distribution based on } P(Y|X)\end{array}}$$

$$\downarrow$$

$$\boxed{\text{Training Examples } (\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n) \sim P(X,Y)}$$

Discriminative Model

- Select classification rules H to consider (Hypothesis space)
- find h from H with lowest training error
- Argument: low training error will lead to low prediction error
- Examples: SVM, decision trees, Perceptron

Generative Model

- Select set of distributions to consider for modelling $P(X,Y)$
- find distribution that matches $P(X,Y)$ on training data
- Argument: If match close enough, we can use Bayes' decision rule
- Example naive Bayes, HMM

L#4

Deriving a generative learning Algorithm

$$h(x) = \underset{y}{\arg\max} \left[P(Y=y \mid X=x) \right]$$

Binary Classification

$$P(Y=1 \mid X=x) = \frac{P(X=x \mid Y=1)\, P(Y=1)}{P(X=x)}$$

$$P(Y=-1 \mid X=x) = \frac{P(X=x \mid Y=-1)\, P(Y=-1)}{P(X=x)}$$

$$\Rightarrow \quad h(x) = \underset{y}{\arg\max} \left(P(Y=y \mid X=\vec{x}) \right)$$

$$= \underset{y}{\arg\max} \left[\frac{P(X=x \mid Y=y)\, P(y)}{P(X)} \right]$$

As we want max over y, we can remove $P(X)$

$$= \underset{Y}{\arg\max}\; P(X=x \mid Y=y)\, P(y)$$

Intuition

$P(y)$ = How likely is class "y" occuring apriori

$P(X=\vec{x} \mid y=+1)$ = How likely do I see $\vec{x}$ in class +1

$P(X=\vec{x} \mid y=-1)$ = How likely do I see $\vec{x}$ in class -1

Now we have our model as follows:

$$h(x) = \underset{y}{\arg\max}\; P(X=\vec{x} \mid Y=y)\, P(y)$$

Intuition: Lets say we have two bags of +ve & -ve class. We flip a coin & see which class is this. It will be $P(y)$.
Then we go to respective bag & sample $\vec{x}$. It is $P(X|Y)$.
In this way, we generate our dataset. We sample our class based on $P(Y)$ & then we sample X based on $P(X|Y)$. So if we know two distributions i.e $P(Y)$ & $P(X|Y)$ then we can classify in generative way.

Naive Bayes Assumption

$$P(X=\vec{x} \mid Y=1) = P(X_1=x_1, X_2=x_2, \dots, X_N=x_n \mid Y=1)$$
$$= \cancel{P(x_1 \dots \mid Y=1)}\, P(x_1=x_1 \mid Y=1)\, P(x_2=x_2 \mid Y=1) \dots P(x_n=x_n \mid Y=1)$$
$$P(X=\vec{x} \mid Y=-1) = \dots$$

i.e all features are independent

Example on slides on NB

$$\hat{P}(y=1) = 3/4 \qquad \hat{P}=(Y=1) = 1/4$$

$$\hat{P}(x_{fever} = high \mid Y=1) = 2/3, \quad \hat{P}(x_{fever} = low \mid Y=1) = 1/3, \quad \hat{P}(x_{fever} = n \mid Y=1) = 0$$

$$\hat{P}(x_{fever} = high \mid Y=-1) = 0, \quad \hat{P}(x_{fever} = low \mid Y=-1) = 1, \quad \hat{P}(x_{fever} = n \mid Y=-1) = 0$$

Classify new example

$$P(Y=1 \mid X) = P(Y=1) \prod P(X_i=x_i \mid Y=1)$$
$$= 3/4 \; (\underset{fever=h}{3/4} \times \underset{cough=no}{1/3} \times \underset{pukes=yes}{2/3})$$
$$= 1/11 \text{ or check it}$$

we now need to estimate polynomial number of parameters i.e # of params = # of features × # of classes + # of classes

Before making naive bayes assumption we had exponential # of parametres

$$P(Y=-1 \mid X) = P(Y=-1) \prod P(X_i=x_i \mid Y=-1)$$
$$= 1/4 \times 0$$
$$= 0 \text{ (use laplace smoothing)}$$

Now look at :

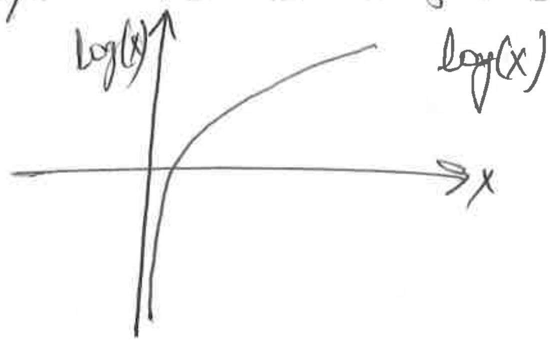
$$h_{naive}(\vec{x}) = \underset{y\in\{+1,-1\}}{\arg\max} \left\{ P(Y=y) \prod_{i=1}^{n} P(X_i=x_i \mid Y=y) \right\}$$

Apply log & it will not change max.

$$h(x) = \arg\max \Big[\underbrace{-\log P(Y=y)}_{b_y} + \underbrace{\sum_{i=1}^{n} \log P(X_i=x_i \mid Y=y)}_{W_y \cdot X} \Big]$$

i.e it will become a linear ~~rule~~ function. OR in Naive Bayes we actually learn linear rule in log scale.

we take negative log ~~the~~ (see graph). b/c log of real number b/w 0 & 1 is negative.



L#15 Generative Model for Classification

Review:

- Discriminative learning (for us Expected Risk Minimization)
  - find rule with low training error
- Generative learning
  - Estimate $P(X,Y)$ from our training data and then derive classification rule via Bayes decision rule.
- Bayes Decision rule: $h(x) = \underset{y}{\operatorname{argmax}} \{P(Y=y|X=x)\}$
  - If $P(X,Y)$ is known, then Bayes decision rule is optimal
- Multivariate Naive Bayes algorithm

$$h(x) = \underset{y}{\operatorname{argmax}} \left\{ \frac{P(X|Y)\, P(Y)}{P(X)} \right\}$$

$$= \underset{y}{\operatorname{argmax}} \{ P(X|Y) P(Y) \}$$

$P(Y)$ = prior prob of class before looking at X.

Assume:

$$P(X=x \mid Y=y) = \prod_{i=1}^{N} P(X_i = x_i \mid Y=y)$$

In Generative Model, we need to estimate following two distributions from ~~our~~ training data: (i) $P(Y=y)$ ii) $P(X_i=x_i|Y=y)$

* For different generative algorithms, we have different assumptions over class conditional model i.e $P(X|Y)$. In naive Bayes, each feature of X is independent & in HMM, x_t depends on x_{t-1}.

* Why do we call it generative ~~approach~~ approach?

Because we are modelling generative process that generate our training data.

Consider classification rule (Bayes classification rule):

$$h(x) = \arg\max_y \underbrace{P(X=\vec{x} \mid Y=y)}_{\text{Class conditional}} \underbrace{P(Y=y)}_{\text{Prior}}$$

Here we assume that our ~~data~~ class conditional model can be decomposed as follows

~~$\arg\max_y P($~~

$$P(X=\vec{x} \mid Y=y) = \prod_{i=1}^{N} P(X_i = x_i \mid Y=y)$$

$\boxed{\text{ASSUMPTION OF DATA Generation}}$

~~Now imagine we have "N" bags for +ve & N bags for -ve training examples~~

BINARY CLASSIFICATION

Imagine we have "N" bags for "N" features that belong to +ve class

" " " " " " " " " " " -ve "

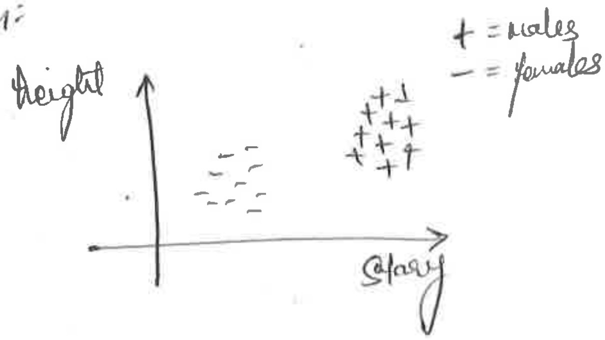
We flip a coin, and it turns +ve. We will go to 1st bag that belong to +ve example & pull out feature value. Then will will move to 2nd, ..., N^{th} bag & we will form $\vec{X_1} = (x_1, x_2, \ldots x_N)$ 1st example labelled as +ve.

In the same way, we will generate whole training set. So, we assumed that our data was generated in the above explained way & we therefore learnt two distributions $P(X=\vec{x} \mid Y=y)$ & $P(Y=y)$ from data. That's why we call this approach, generative approach. Because we just modelled generative process.

The above model is convenient b/c we had discrete ~~variable~~ variables i.e $\vec{X_i} = (x_1, x_2, \cdots x_N)$ takes discrete values.

Now consider different problem:

How would we come out for generative model for this data?



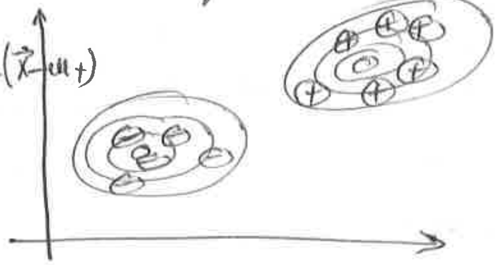
Ans: <u>Linear Discriminant Analysis</u>

$$h(x) = \underset{y}{\operatorname{argmax}} \{ P(X=x \mid Y=y)\, P(Y=y) \}$$

Now we need to model:

i) $P(Y=y)$ that's easy. Just count +ve & -ve classes

ii) $P(X=x \mid Y=y)$: what should be the assumption for this distribution? Ans: Gaussian or spherical distribution. Fit a gaussian



Our Assumption:

$$P(X=\vec{x} \mid Y=1) = N(\vec{\mu}_+, \Sigma_+) = \frac{1}{2\pi^{N/2} |\Sigma_+|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\mu_+)\Sigma^{-1}(\vec{x}-\mu_+)}$$

$$P(X=\vec{x} \mid Y=-1) = N(\vec{\mu}_{-1}, \Sigma_-)$$

Since we have round shapes

$$\Sigma_+ = \Sigma_{-1} = 1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$ (Spherical Gaussian with equal covariance)

If our data is elongated, we can normalize data with $\mu = 0$ & $\Sigma = 1$ which will give above illustration of data.

$$h(x) = \underset{y}{\operatorname{argmax}} \{ N(\mu_y, \Sigma_y)\, P(Y=y) \}$$

Dropping out constants

$$= \underset{y}{\operatorname{argmax}} \{ e^{-\frac{1}{2}(\vec{x}-\mu_y)^2}\, P(Y=y) \}$$

Apply log

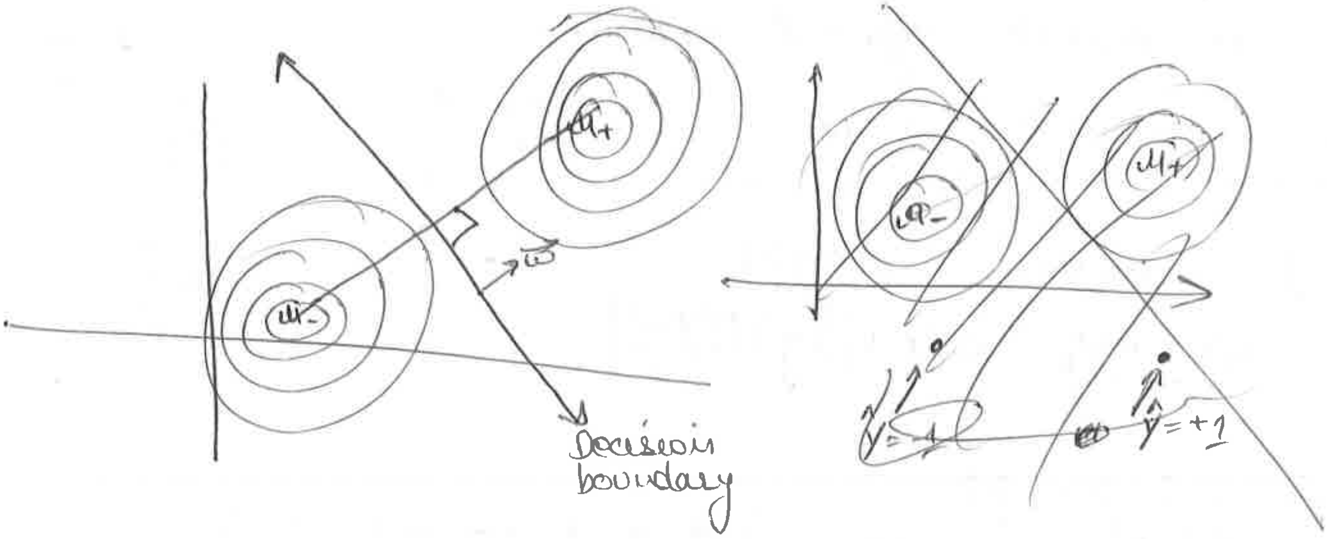
$$= \underset{y}{\operatorname{argmax}} \{ -\frac{1}{2}(\vec{x}-\mu_y)^2 \log P(Y=y) \}$$

Assume $P(y=+1) = P(y=-1) = 0.5$

$$= \underset{y}{\operatorname{argmin}} \ -\frac{1}{2}(\vec{x}-\mu_y)^2$$

$$= \underset{y}{\operatorname{argmin}} \{ \|\vec{x}-\mu_y\| \}$$

i.e we can classify by measuring the distance b/w $\vec{x}$ & mean of class.



If we assume that prior probabilities are same then we actually classify based on distances. If we classify based on distances, then we again learnt linear decision boundary as shown in above fig. That decision boundary is $\perp$ to the line passing through μ_+ & μ_-. But we have to assume that we have unit covariance for both classes.

$$w = \mu^+ - \mu^-$$
$$b = ??$$

We got linear classifier with minimization error framework.

Naive Bayes Classifier (Multinomial)

Bayes Rule:

$$h(x) = \arg\max_y \underbrace{P(X=x \mid Y=y)}_{\text{How to model it}} P(Y=y)$$

b/c X = sequence of words

Assume: Documents are generated by independently drawing "l" words according to a class conditional distribution.

$$P(X=x \mid Y=1) = P(\omega_1 = \omega_1, \omega_2 = \omega_2, \ldots, \omega_l = \omega_l \mid y=1)$$
$$= \prod_{i=1}^{l} P(\omega_i = \omega_i \mid Y=1)$$

$P(X=\vec{x} \mid y=-1) = \ldots$

Now, lets see its generative process & consider the following example.

$x_1 =$ (The, art, of, Programming) $y_1 = +1$

$l_1 = 4$

Now we have two bags of ~~...~~ words. One bag contains words for computer science documents and another bag contains words for the articles that donot belong to computer science. We flip a coin ~~...~~ & get class label. then we go to the respective bag & draw each word independently of others i.e by replacement. So for $l_1=4$, we will draw 4 words & during each draw we put the previously drawn word back to the bag.

This is different from ~~multinomial~~ multivariate Naive Bayes model where we had "N" different bags for "N" features and for each class.

The key observation there is that we are not modelling syntax. We are just drawing words whose order does not matter. We could have used multivariate ~~Naive Bayes assumption~~ method here for modeling $P(X|Y)$ but then we would have to go through huge amount of words. Therefore, we just modeled those words which are present in example.

How to ~~model~~ estimate $P(W=w \mid Y=y)$, see slides.

Its also linear decision rule. (See homework)

* We know text is not generated in the above described way. Therefore this model is gross-oversimplification. But this model is superfast.
* Statistically, order ~~@~~ order does not matter.
* LDA can be kernelized.
* Good slides for comparing different model at the end of this lecture.

L#16

Modeling Sequence Data: Markov Models

Review

- Generative Modeling: $h(x) = \underset{y \in Y}{\operatorname{argmax}} \{ \underbrace{P(X|Y)}_{\text{Class conditional distribution}} \overbrace{P(Y)}^{\text{Prior}} \}$

  → Prior: How likely is "y" before seeing data

- Multinomial Naive Bayes (applied to sequence, documents)

  Assumption: $P(X=x \mid Y=y) = \prod_{i=1}^{l} P(W_i = w_i \mid Y=y)$ (bag of words)

  Estimate: $\underbrace{P(w|Y)}_{\#\text{ of words} \times \#\text{ of classes}}, \underbrace{P(Y)}_{\#\text{ of classes}}$

- Linear Discriminant Analysis:

  Assume: $P(X=\vec{x} \mid y=y) = N(\mu_x, 1)$

  Estimate: μ_y

Less Naive Bayes

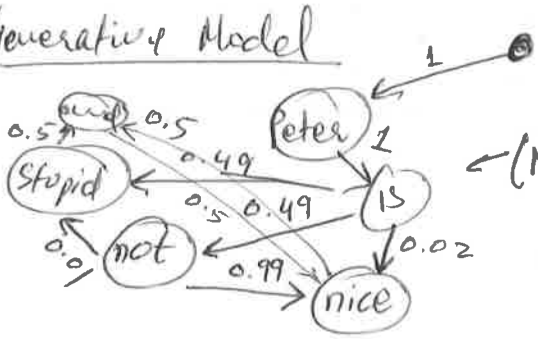
Assume:

$$P(X=x \mid Y=y) = P(w_1, w_2, \dots, w_l \mid y)$$ } Here X is a sequence

(Rule of Probability) $$= P(W_1|y)\, P(W_2=w_2 \mid W_1=w_1, y=y) \dots P(W_l=w_l \mid W_1=w_1, \dots, W_{l-1}=w_{l-1}, y)$$

$$= P(W_1=w_1 \mid Y=y)\, P(W_2=w_2 \mid W_1=w_1, Y=y)\, P(W_3=w_3 \mid W_2=w_2, Y=y) \dots P(W_l=w_l \mid W_{l-1}=w_{l-1}, Y=y)$$

Generative Model

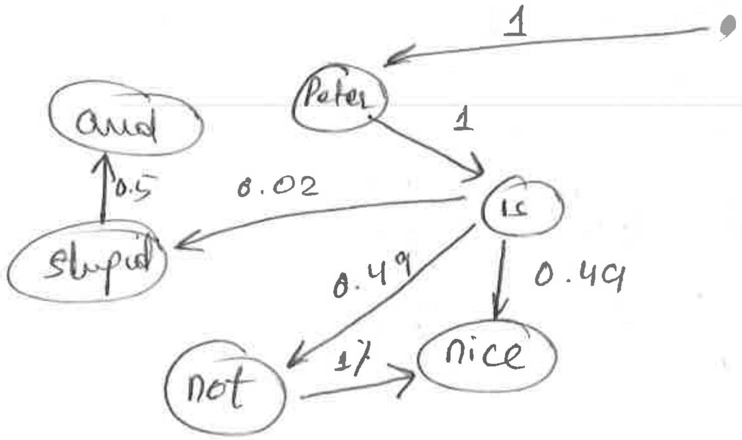


← (Model for insulting sentence)

* Edges represent class-conditional probability

*

NOT INSULT



(sum of edge from node = 1)

We flip a coin and go to respective graph. Then we start from ~~root~~ start node and perform random walk "l" times on respective graph to generate sentences of length "l".

Assume: $P(y) = 0.5$ (uniform)

$$P(y=1 \text{ (insulting)})\, P(X=x_1 \mid y=\text{insult})$$
$$= 0.5\, P(\text{Peter, is, nice, and, not, stupid} \mid \text{insult})$$
$$= 0.5 \times 1 \times 1 \times 0.02 \times 0.5 \times 0.5 \times 0.01$$
$$=$$

$$P(y=-1)\, P(X=x_2 \mid y=-1)$$
$$= 0.5 \times \ldots$$

$$\# \text{ of params} = \left[(\#\text{ of words})(\#\text{ of words}) - 1\right] \times \text{classes}$$

Generative Model for POS tagging

$$h(x) = \underset{y \in Y}{\operatorname{argmax}} \{P(X=x \mid Y=y)\ \underbrace{P(Y=y)}\}$$

(This was "2" classes when there was no sequence. With sequence its exponential)

Prior: $P(Y=(y_1, \ldots, y_l))$

- Dependencies follow Markov Model
- $$P(Y=(y_1,\ldots,y_l)) = \underbrace{P(Y_1=y_1)}_{\text{start prob}} \underbrace{\prod_{i=2}^{l} P(Y_i=y_i \mid Y_{i-1}=y_{i-1})}_{\text{transition}}$$

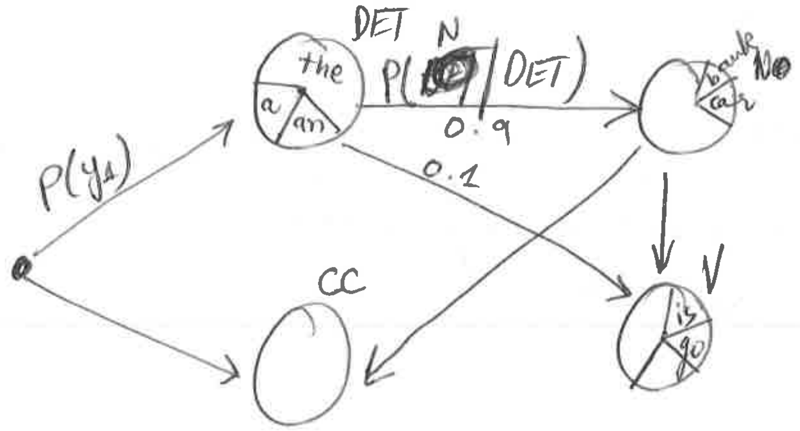
Class - Conditional Model $P(X=x|Y=y)$

- Assume that word x_i is independent of
  - all other words
  - all tags except y_i
- $P(X=(x_1,\dots,x_\ell)|Y=(y_1\dots y_\ell))$

$$= \prod_{i=1}^{\ell} P(X_i=x_i|Y_i=y_i)$$

i.e we have markov model over tags & simple model over $P(X|Y)$.

Generative Process



* Circles are bag of words with state (DET, N, CC, V)

SAMPLE Y & then X:

$y=(\text{DET},$ $\xrightarrow{\text{random walk}}$ $y=(\text{DET}, \text{Noun}$ $\dashrightarrow$ $y=(\text{DET}, \text{N}, \text{V}$

$X=(\text{The},$ $\xrightarrow[\text{bag of words}]{\text{draw from}}$ $X=(\text{The}, \text{car}$ $\dashrightarrow$ $X=(\text{The}, \text{car}, \text{is}$

→ we will have a random walk over above graph which which be guided by $P(y_i|y_{i-1})$. We will generate label sequence & then we will pick a word from respective ~~word~~ bag. We will sample words based on their frequencies/probabilities inside bag.

* We need fast algorithm to compute argmax ~~…~~ for sequence of labels.

L#17 Modeling Sequence Data: HMMs and Viterbi

Review:

Prob. Dist over sequences

- Markov Model
  - Prob distribution over sequences

$$P(s=(s_1,\cdots,s_\ell)) = \underbrace{P(S_1=s_1)}_{\text{Start Prob}} \prod_{i=2}^{\ell} \underbrace{P(S_i=s_i \mid S_{i-1}=s_i)}_{\text{transition prob}}$$

  - Sampling from above distribution is random walk over graph whose edges are transition Prob
  - # of sequences of length "ℓ" with "K" symbols is K^ℓ. We have defined probability distribution over K^ℓ sequences by Markov assumption. Its parameters are $K^2 + K$

$$\underset{\#\text{ of trans}}{K^2} + \underset{\#\text{ of start prob}}{K}$$

- Less Naive Bayes Classifier
  - Assumption: Class Conditional model $P(X|Y)$ are Markov Models
- Till here, we were predicting binary variables

- Sequence Prediction:
  - Application: POS tagging
  - Task was: Given sequence X, predict sequence Y
  - Generative Model:

$$P(X=x|Y=y) \times P(Y=y)$$

— In $\underbrace{P(X=x|Y=y)}$ $P(Y=y)$

Since it is huge as $\vec{x}$ is a sequence of words, it will have huge # of conditional distributions. So, we made an assumption as follows:

$$P(X=x|Y=y) = \prod_{i=1}^{l} P(X_i=x_i|Y_i=y_i) \quad \text{i.e we draw } x_1$$

$$\boxed{\vec{x} = (\underset{x_1}{I}, \underset{x_2}{am}, \underset{x_3}{living}, \underset{x_4}{in}, \underset{x_5}{PAK})}$$

by sampling y_1 & then we goto bag of words ~~label~~ belonging to state y_1. Then we sampled x_1.

$$P(Y=y) = P(Y_1=y_1) \prod_{i=2}^{l} P(Y_i=y_i|Y_{i-1}=y_{i-1}) \rightarrow \text{Modeling POS sequence by random walk.}$$

— Put everything together, we will see HMM.

i.e $P(X,Y) = P(X|Y) \times P(Y)$

$= P(x_1,\ldots,x_l, y_1,\ldots,y_l)$

— See Viterbi Example in slide #3.

Example:

$P(x=(I, bank, at, CFCU), y=(N,V,N,V)) = ??$ what is the probability of this sequence

$= P(Y_1=N)\,P(X_1=I|Y_1=N) \cdot P(Y_2=V|Y_1=N)\,P(X_2=bank|Y_2=V)$

$P(Y_3=N|Y_2=V)\,P(X_3=at|Y_3=N) \cdot P(Y_4=V|Y_3=N)\,P(X_4=CFCU|Y_4=V)$

= Get value from slides

$= (0.1\times 0.01) \cdot (0.19\times 0.4) \times (0.3\times 0.01) \times (0.19\times 0.01)$

$= 4.332\times 10^{-10}$ (very unlikely)

— For HMM, ~~we need~~ if we have "K" symbols then we need to estimate following params:

trans = K^2

emission = l (# of distinct words in English language or vocabulary)

* If we don't simplify our model, then number of parameters would be exponentially huge.

* Viterbi is general case of Belief Propagation.

* Viterbi Algorithm

* Prob of most likely sequence of states/tags ending in state "y" at position "i".

$$\delta_y(i) = \max_{(y_1 \cdots y_{i-1}) \in S^{i-1}} P(X_1 = x_1 \cdots X_i = x_i, Y_1 = y_1, \dots, Y_{i-1} = y_{i-1}, Y_i = y)$$

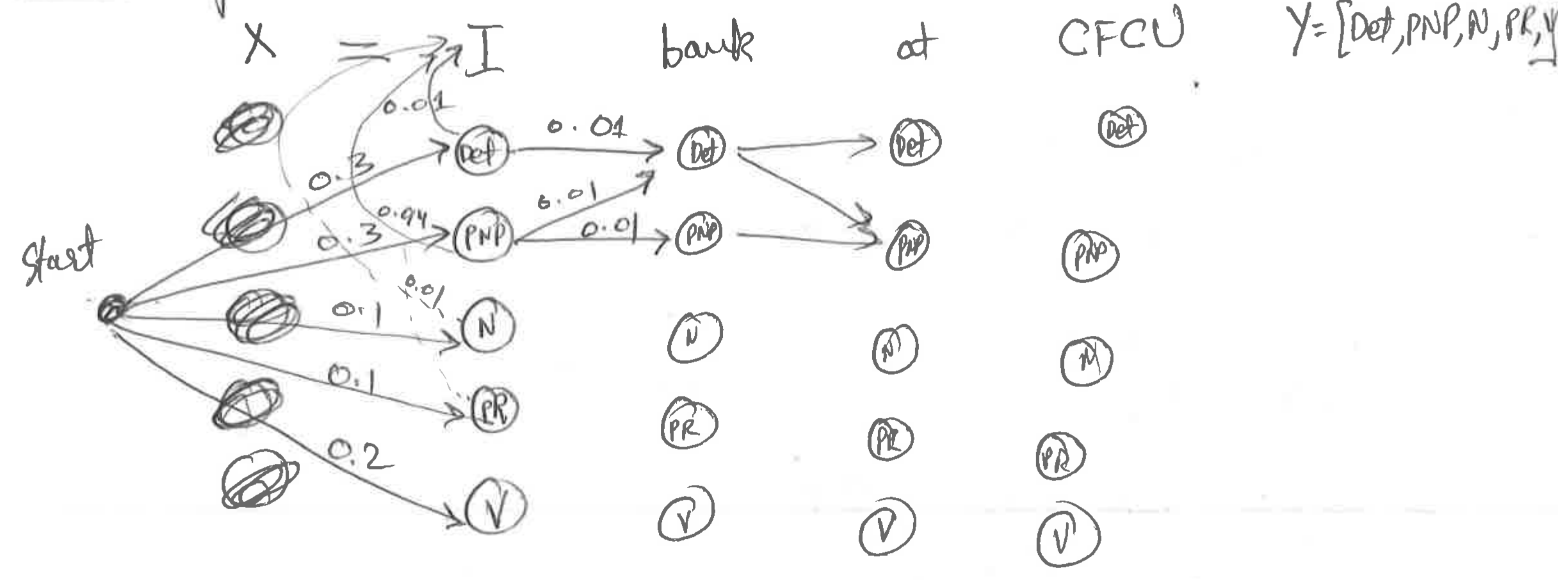
$$= \max_{(y_1 \cdots y_{i-1}) \in S^{i-1}} \Big\{ P(X_1 = x_1 | Y_1 = y_1)\, P(y_1) \prod_{j=2}^{i-1} P(X_j = x_j | Y_j = y_j)\, P(Y_j = y_j | Y_{j-1} = y_{j-1}) \Big\}$$

$$P(X_i = x_i | Y_i = y)\, P(Y_i = y | Y_{i-1} = y_{i-1}) \Big)$$

$$\delta_y(i+1) = \max_{V \in S} \big\{ \delta_V(i) \cdot P(Y_{i+1} = y | Y_i = V) \cdot P(X_{i+1} = x_{i+1} | Y_{i+1} = y) \big\}$$

$$\delta_y(1) = P(Y_1 = y)\, P(X_1 = x_1 | Y_1 = y)$$

Trellis Graph



δ	1	2	3	4
DET	0.003 (1 oper)	2.82×10^{-5} (5 oper)		
PRP	0.282 (1 DP loops)	2.82×10^{-5}		
N	0.001 (1 op)	1.128×10^{-3}	~~1.5×10^{-2}~~	1.5×10^{-3}
PREP	0.001 (1) op	5.64×10^{-4}	1.25×10^{-4}	
V	0.002 (1) op	8.69×10^{-2}		
	PRP	V	Prep	N

$\delta_{DET}(1) = 0.3 \times 0.01 = 0.003$

$\delta_{PRP}(1) = 0.3 \times 0.94 = 0.282$

$\delta_N(1) = 0.1 \times 0.01 = 0.001$

$\delta_{prep}(1) = 0.1 \times 0.01 = 0.001$

$\delta_V(1) = 0.2 \times 0.01$

$$\delta_{DET}(2) = \max \begin{cases} V=DET & 0.003 \times 0.01 \times 0.01 \\ V=PRP & 0.282 \times 0.01 \times 0.01 \\ V=N & 0.001 \times 0.01 \times 0.01 \\ V=PREP & 0.001 \times 0.3 \times 0.01 \\ V=\text{Verb} & 0.002 \times 0.2 \times 0.01 \end{cases} = 2.82\times10^{-5}$$

$$\delta_{PRP}(2) = \max \begin{cases} DET & 0.003 \cdot 0.01 \cdot 0.01 \\ PRP & 0.282 \cdot 0.01 \cdot 0.01 \\ N & 0.001 \cdot 0.2 \cdot 0.01 \\ PREP & 0.001 \cdot 0.2 \cdot 0.01 \\ V & 0.002 \cdot 0.19 \cdot 0.01 \end{cases} = 2.82\times10^{-5}$$
PRP

$Y^* = (PRP, V, PREP, N)$

L#18 STATISTICAL Learning Theory: PAC Learning

Review:

- Structured Prediction
  - → sequence prediction
    - $h: X \to Y$ (X,Y) are sequences

  Y = structured multivariate object

- HMM Generative Model
  $P(X,Y) = P(y_1)\, P(x_1|y_1) \prod_{i=2}^{l} P(y_i|y_{i-1})\, P(x_i|y_i)$

- Compute $h(x) = \underset{y}{\operatorname{argmax}}\; P(x,y)$
  - Viterbi Alg $O(l\, |\Sigma_y|^2)$

  l = length of sentence
  Σ_y = # of tags

- Estimating $P(X,Y)$ is extremely difficult b/c Y is sequence. Therefore we have Markov assumption $P(Y_i|Y_{i-1})$ & $P(X_i|Y_i)$. After this assumption, we can easily impute $\arg\max_Y P(X,Y)$ using Viterbi algorithm.

<u>Setup:</u>

- $|H|$ students $H = \{h_1, \dots, h_{|H|}\}$
- n bit sequence
- $p = 0.5$ probability of erring on a single bit given you are not psychic

<u>Question:</u> How likely is it that a particular player "i" guesses the code without being psychic?

$$P(h_i \text{ correct} \mid h_i \text{ nonpschic}) = (1-p)^n$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$P(h_{|H|} \text{ correct} \mid h_{|H|} \text{ nonpschic}) = (1-p)^n$$

<u>Real Question:</u> How likely is it that atleast one student ~~guesses~~ guesses code without anybody being psychic?

$$P(h_1 \text{ correct} \vee h_2 \text{ correct} \vee \dots \vee h_{|H|} \text{ correct} \mid \text{all non psychic})$$

$$= 1 - P(h_1 \text{ not correct} \wedge \dots \wedge h_{|H|} \text{ noncorrect} \mid \text{all non psychic})$$

with independent assumption

$$= 1 - [P(h_1 \text{ not correct} \mid h_1 \text{ not psychic}) \dots P(h_{|H|} \text{ not corr} \mid h_{|H|} \text{ not psychic})]$$

$$= 1 - [\underbrace{(1-(1-p)^n) \dots (1-(1-p)^n)}_{|H| \text{ times}}]$$

$$= 1 - \prod_{i=1}^{|H|} [1-(1-p)^n]$$

$$= 1 - (1-(1-p)^n)^{|H|}$$

Let $|H| = 173$, $n=4$, $p = 1/2$: $\quad 1 - [1-(\tfrac{1}{2})^4]^{173} = 0.9999986$

i.e I am pretty sure that somebody will guess code. There is no evidence that all are non psychic.

How long do I have to make the sequence to be confident of psychic abilities?

$$1-(1-(1-p)^n)^{|H|} < \delta$$ small i.e we want to find "n" so that prob of getting tracked goes down

$$n > \log_{(1-p)}\left[1-(1-\delta)^{\frac{1}{|H|}}\right]$$

for $\delta = 0.05$, (prob of detecting n bits given every one is non-psychic)

$|H| = 173$

$p = 0.5$ (null hypothesis)

$$n > \log_{1/2}\left(1 - 0.95^{1/173}\right) = 11.7$$

i.e we have to make our sequence 12 bits long to make the chance of detecting 12 bit sequence 5%.

ANALOGY In learning

- H hypothesis space $\hat{=}$ students in class
- n size of training sample $\hat{=}$ length of sequences
- $Err_{train}(h)$ number of train error $\hat{=}$ number of bit guessed correctly
- $Err_p(h)$ prediction/true error $\hat{=}$ p (prop erring on bit)

Question: Can we bound (upper bound) the probability that our learning alg "L" returns the hypothesis with large prediction error $(Err_p(h) \geq \epsilon_p)$?

Assumption:
i) Learning algorithm "L" has finite hypothesis space "H" [for instance decision trees]
ii) At least one hypothesis $h \in H$ has zero prediction error $Err_p(h) = 0$ $(\rightarrow Err_{train}(h) = 0)$
iii) Learning Alg "L" returns zero training error hypothesis $\hat{h}$.

To answer the above question, lets decompose that question

(1) Probability that single hypothesis "h" ~~#~~ with $Err_p(h) \geq \varepsilon$ has train error 0?

$$P\left(Err_S(h) = 0 \mid Err_p(h) \geq \varepsilon\right) \leq (1-\varepsilon)^n$$

(our assumption lies in i.i.d data)

(2) Probability that there exists at least one $h \in H$ with $Err_p(h) \geq \varepsilon$ and no training error? (It makes it indistinguishable from $h \in H$ that had $Err_S(h) = Err_p(h) = 0$).

Let $h_1 \dots h_K$ be the hypothesis in H with $Err_p(h) \geq \varepsilon$

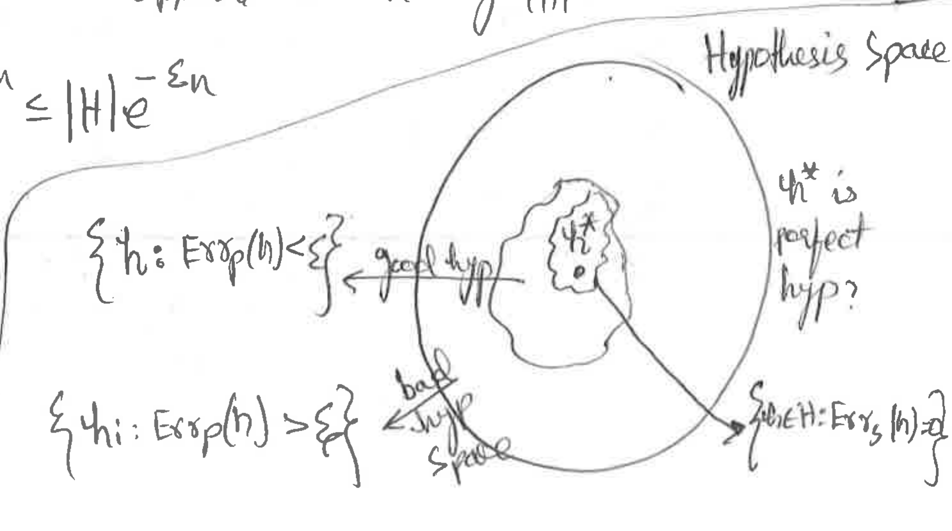
$$P\left(Err_S(h_1) = 0 \vee \dots \vee Err_S(h_K) = 0 \mid Err_p(h_1) \geq \varepsilon, \dots, Err_p(h_K) \geq \varepsilon\right)$$

$$\leq \sum_{i=1}^{K} P\left(Err_S(h_i) = 0 \mid Err_p(h_i) \geq \varepsilon\right)$$: Here we have used union bound that does not have any independence assumption i.e $P(X_1 = x_1 \vee X_2 = x_2 \vee \dots \vee X_n = x_n) \leq \sum_{i=1}^{n} P(X_i = x_i)$. Here we don't have independence assumption. Imagine two decision trees that differ by leaf, they will have dependence among their predictions b/c of their similarity in structure.

$$\leq K(1-\varepsilon)^n$$ here K is the no of hypothesis in hypothesis space

$$K(1-\varepsilon)^n \leq |H|(1-\varepsilon)^n$$: How to upper bound K? we can upper bound it by $|H|$.

$$K(1-\varepsilon)^n \leq |H|(1-\varepsilon)^n \leq |H| e^{-\varepsilon n}$$



BASICALLY, what we have proved is the probability that learning error returns hypothesis is actually from the set where $h: Err_p(h) < \varepsilon$ i.e set of those hypothesis whose prediction error $Err_p(h)$ is less than ε.

→ The probability that L returns h with prediction error greater than $\mathcal{E}$ is atmost $|H| \bar{e}^{\mathcal{E}n}$

Here $n \rightarrow$ # of training examples

$|H| \rightarrow$ cardinality of hypothesis

Now we have an upperbound on the worst performance of algorithm "L".

~~Question~~ <u>Sample Complexity</u>

How many examples does a learning algorithm "L" needs so that with prob $1-\delta$ it learn h with pred error $[Err_p(h)]$ less than $\mathcal{E}$?

$$|H| \bar{e}^{\mathcal{E}_p n} \leq \delta$$

Let $\delta = 0.05$ (prob of screwing up)

Now Solve for "n".

$$\boxed{n \geq \frac{1}{\mathcal{E}}\left(\log |H| - \log \delta\right)}$$

(under three assumptions we stated before)

All of this is PAC learning = Probably approximately correctly.

We want ~~probably~~ approximately correct & we can be correct upto some measure of $\mathcal{E}$. And we can never be fully correct i.e we can always be probably correct with probability of messing up δ. This is PAC learning.

i.e $$P(Err_p(h_{L(S)}) \leq \epsilon) \geq 1 - \delta$$

i.e probability of learning hypothesis that have prediction error less than ϵ is 1 − prob of messing up.

By increasing "n", we can increase the chances of getting such hypothesis whose $P(Err_p(h_L(S))) \geq 1 - \delta$.

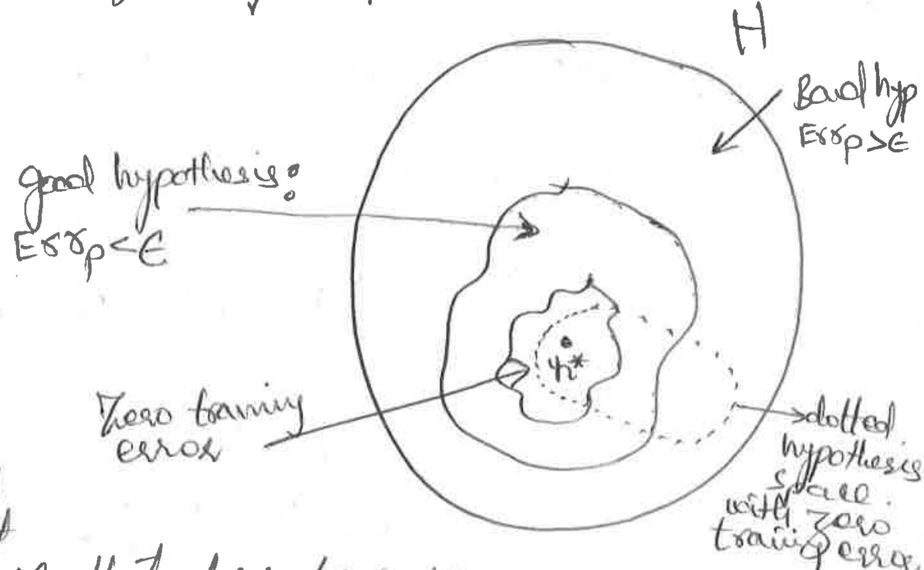
This is PAC-learning.

L#19: Statistical Learning Theory: Error bounds & VC-Dimension

Review:

- PAC ~~style~~ learning
  - $P(Err_p(h_{L(S)}) \le \epsilon) \ge 1 - \delta$
  - ϵ is the amount of error (prediction error) we are willing to tolerate
  - δ is the prob that "L" fail. It is the prob we allow learner to fail.
  - We can never be sure about zero-generalization error.
- Generalization error bound for finite H, ~~the~~ concept is in H:
  - $P(Err_p(h_{L(S)}) \ge \epsilon) \le |H| \cdot e^{-\epsilon n}$

    n = # of training examples



* Problem starts when our dotted hypothesis space actually goes out from the region of good hypothesis space where $Err_p < \epsilon$. Therefore, our learner "L" can actually return hypothesis "h" that lies in bad hypothesis space that has training error = 0 but $Err_p > \epsilon$.

The proof we did last time said:
"Can we ~~pro~~ say with certain probability that zero training error hypothesis are strictly contained in a set of good hypothesis set." $P(Err_p(h_{L(S)}) \ge \epsilon) \le |H| \, e^{-\epsilon n}$

"probability that no bad hypothesis ended up in the set of zero training error hypothesis".

- Sample Complexity:

"How many training examples do I need to be reasonably sure that I get good hyp"

OR

"It asks how many training examples do I need so that with probability $1-\delta$ I get a hypothesis returned by my learning algorithm that has prediction error no greater than ϵ." Basically it solves for "n" the following

$$P\left[Err_P(h_{L(S)}) \geq \epsilon\right] \leq |H| e^{-\epsilon n}$$

In todays lecture we will relax some assumptions:

i) We can have hypothesis with non-zero training error

ii) Our hypothesis space is infinite i.e linear function.

Generalization Error bound: finite H, non-zero error

Learner: L returns $h \in H$ with minimum training error

Idea: Bound the deviation b/w training and generalization error

$$P\left(\left|Err_S(h_{L(S)}) - Err_P(h_{L(S)})\right| \leq \epsilon\right) \geq 1-\delta$$

* we will say something about all hypothesis in hypothesis space

$$\Leftarrow \quad P\Big(\underbrace{\max_{h \in H} \left|Err_S(h_i) - Err_P(h_i)\right|}_{\text{what is the maximum deviation.}} \leq \epsilon\Big) \geq 1-\delta$$

$$\Leftrightarrow \quad P\left(\max_{h \in H} \left|Err_S(h_i) - Err_P(h_i)\right| > \epsilon\right) < \delta \quad \text{(takeing complement of above)}$$

$$P\left(\left|Err_S(h_1) - Err_P(h_1)\right| > \epsilon \vee \ldots \vee \left|Err_S(h_{|H|}) - Err_P(h_{|H|})\right| > \epsilon\right)$$

Using Union bound

$$\leq \sum_{i=1}^{|H|} P\left(\left|Err_S(h_i) - Err_P(h_i)\right| > \epsilon\right)$$

: $Err_S(h_1) = \frac{1}{n} \sum_j \Delta(h(x_j), y_j)$

Using Hoeffding / Chernoff Bound (for binomial distribution)

$$P\left(\left| \frac{1}{n} \sum_{i=1}^{n} x_i \right) - p \right| \geq \epsilon \right) \leq 2 e^{-2n\epsilon^2}$$

$$\leq \sum_{i=1}^{|H|} 2 e^{-2n\varepsilon^2} = |H| \, 2 e^{-2n\varepsilon^2}$$

We can turn this ~~...~~ around interms of sample complexity.

Sample Complexity

$$|H| 2 e^{-2n\varepsilon^2} < \delta$$

i.e $n \geq \frac{1}{2\varepsilon^2} (\log|H| - \log \delta - \log 2)$

Example: Boolean conjunctions over "l" literals ($N=10$, $l=3$)

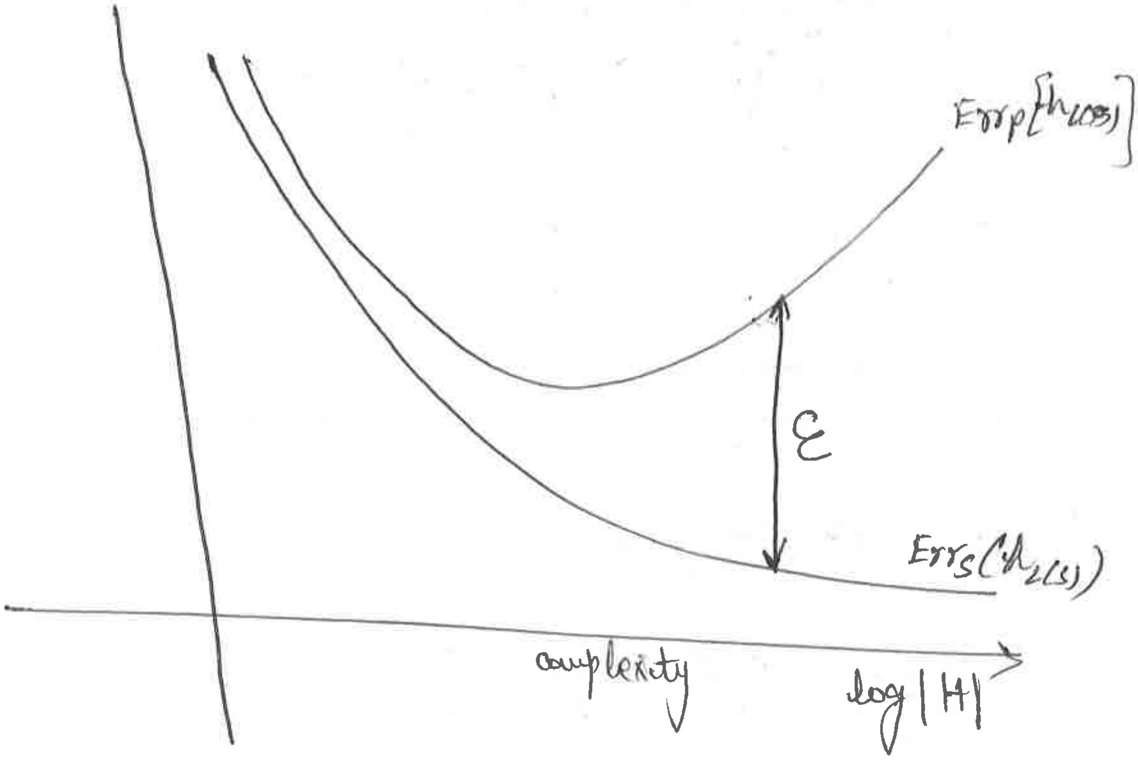
$$\rightarrow |H| = \binom{N}{l} \cdot 2^l = 960 \text{ Hypothesis}$$

How many examples do I need so that $\varepsilon = 0.1$ i.e 10% deviation b/w training & prediction error?

$\varepsilon = 0.1$

$\delta = 0.05$ (screw-up probability)

$\Rightarrow$ $n \geq 528$ (I need 529 examples so that I get no more than 10% deviation b/w training & prediction error with 95% confidence interval). In theory, it tells us how many examples do we need. In practise, we need much more example. These bound tells you behaviour.



These bound tells us why overfitting is happening.

<u>Overfitting</u>

for a fixed δ, solve for ε

$$2|H|2e^{-2n\varepsilon^2} = \delta$$

$$\iff \varepsilon = \sqrt{\frac{1}{2n}\log(2|H|) - \log(\delta)}$$

$$\Rightarrow P\left(|Err_S(h_{L(S)}) - Err_P(h_{L(S)})| \geq \sqrt{\frac{1}{2n}(\log(2|H|) - \log(\delta))}\right) \leq \delta$$

Take complement

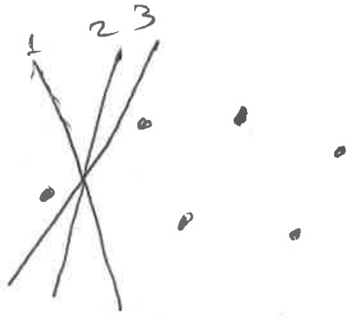
" " $\leq$ " $) \geq (1-\delta)$

$\iff$ with prob $1-\delta$: $Err_P(h_{L(S)}) \leq Err_S(h_{L(S)}) + \sqrt{\frac{1}{2n}(\log(2|H|) - \log\delta)}$

Infinite Hypothesis Spaces

Idea: Consider only the "effective" number of Hypothesis (i.e hypothesis that donot give the same classification).

Example:



* we will lump 1, 2, 3 together b/c all of them classify exactly similarly

Question: How many different hypothesis exist in H for sample S?

$\Pi_H(s)$ contains one $h \in H$ for each different classification of sample "S"

→ It is effectively finite number of hypothesis

(If we have "n" points, we will have 2^n different hypothesis)

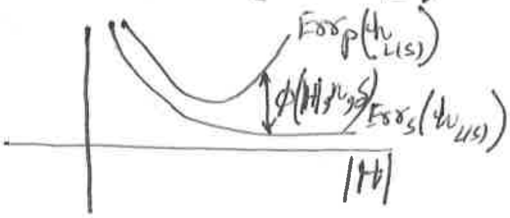
→ fortunately, for many infinite H

$|\Pi_H(s)| << 2^n$ for all Samples S.

L#20 STATISTICAL Learning theory: Experts And Bandit

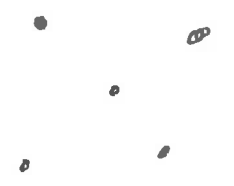
Review:

- Generalization Error Bound
  → bound $P(|Err_S(h_{L(S)}) - Err_P(h_{L(S)})| \geq \varepsilon) \leq \delta$
  → with Prob $1-\delta$, $Err_P(h_{L(S)}) \leq Err_S(h_{L(S)}) + \phi(|H|, n, \delta)$
- Infinite Hypothesis Spaces



$\Pi_H(S)$ contains one $h \in H$ for each different classification of S. It is always a finite number and usually $< 2^n$

Consider this datapoint:



we have 5 points. We need linear hyperplane to separate them. Thus $\Pi_H(S)$ is much less than 2^4.

<u>Saur's lemma</u>

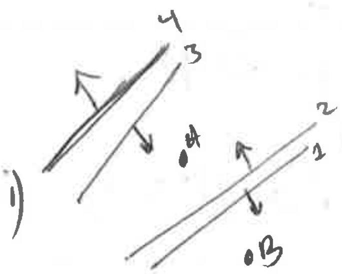
$$\forall S, |\Pi_H(S)| \leq \left(\frac{e\,n}{VCdim(H)}\right)^{VC\,Dim\,(H)}$$

$|\Pi_H(S)|$ is effective measure of hypo.

<u>Example</u>: VC dimension of Hyperplanes in $\mathbb{R}^2$

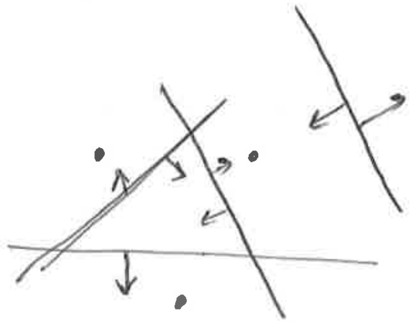
$$H_{L2} = \{h : h(x) = \text{sign}(w.x+b)\}$$

i)



$\text{VC-dimension}(H_{L2}) \geq 2$

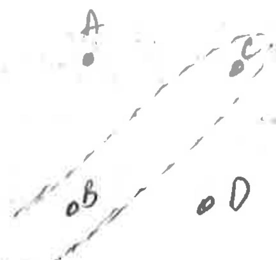
ii)



shattered three points with 2^3 hyperplanes

$\text{VC dim}(H_{L2}) \geq 3$

iii)



we can not ~~class dotted~~ separate B, C from A & D using linear hyperplane

i.e you can not shatter 4 points in 2D planes.

$$\boxed{VC\dim(H_{L2}) < 4}$$

General:

- For linear classifiers in $\mathbb{R}^N$

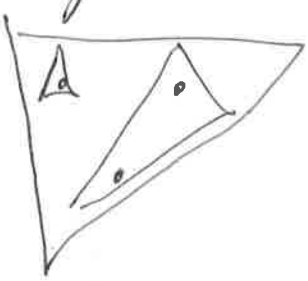
$$VC\dim(H_{LN}) = N+1$$

- For linear classifiers with margin δ

$$VC\dim(H_{\delta}) \leq \frac{R^2}{\delta^2}$$

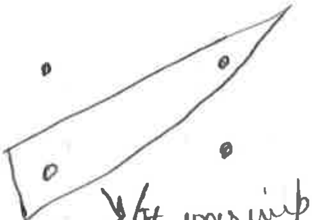
VC dimension of Triangles in $\mathbb{R}^2$

i)



Can a triangle shatters three points?
A: Yes

ii)

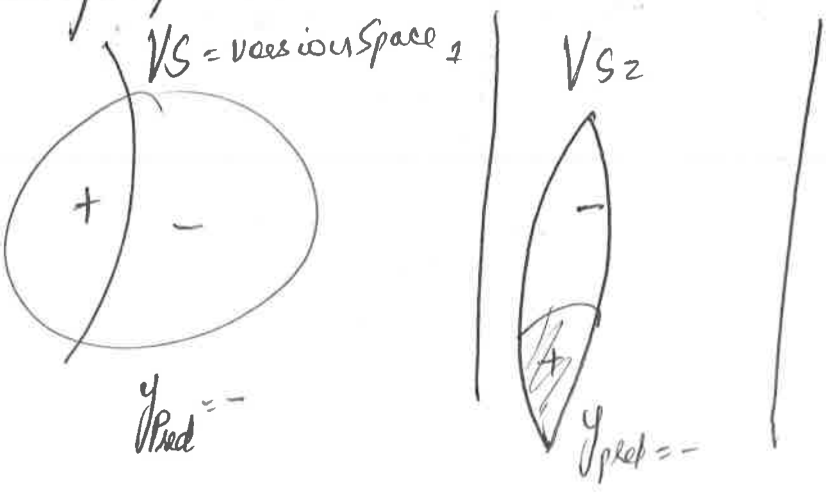


Can a triangle shatters 4 points
A: Yes

(It was impossible in linear classifiers)

And move on to find 5, 6 .... points to find the lower bound of VC dimension of triangles in $\mathbb{R}^2$.

Halving Algorithm



Theorem: After $\log_2(N)$ mistakes, the version space contains only the perfect expert.

Proof: Consider each iteration t:

- Case 1: Algorithm predicts incorrectly
  → more than half of the hypothesis in VS_t were wrong & kicked out.
  → $|VS_{t+1}| \leq \frac{1}{2} |VS_t|$
- Case 2: Algorithm predicts correctly
  → $|VS_{t+1}| \leq |VS_t|$

→ Can only ~~half~~ halve VS for $\log_2(N)$ times before we run out of hypothesis.

→ It is with no noise. Can we make it robust to noise? See Weighted Majority Algorithm.

Example: $H = \{h_1, h_2, h_3\}$, $\beta = \frac{1}{2}$ (experts = hypothesis)

- $t=1$: given x_1: $h_1(x_1) = 1$, $h_2(x_1) = 1$, $h_3(x_1) = -1$

  $w_1 = (1, 1, 1)$: Weight of $y=+1$ is $(w_{11} + w_{12}) = 2$, weight of $y=-1$ is $w_{13} = 1$ } Prediction $y = +1$

  Receive true label y_1: $y_1 = -1$ (we made an error)

  $w_2 = (1/2, 1/2, 1)$

- $t=2$: given x_2: $h_1(x_2) = -1$, $h_2(x_2) = 1$, $h_3(x_2) = -1$

  $w_2 = (\frac{1}{2}, \frac{1}{2}, 1)$: weight of $y=+1$ is $1/2$, weight of $y=-1$ is $3/2$ } $y_{pred} = -1$

  Receive $y_{true} = -1$

  $w = (\frac{1}{2}, \frac{1}{4}, 1)$

  i.e. we can see hypothesis that predicts correctly have larger weight.

Theorem: Let Δ^* be the number of mistakes of the best expert in hindsight (experience), the weighted majority algorithm makes atmost $\Delta^{wm} \leq 2.4\left(\Delta^* + \log_2(N)\right)$ for $\beta = \frac{1}{2}$

Proof: Let h^* be the best expert in hindsight $\longrightarrow w^* = \left(\frac{1}{2}\right)^{\Delta^*}$

Let $W_t = \sum_{i=1}^{N} W_{t+1}$. $W_1 = n$. for each mistake, $W_{t+1} \leq \frac{3}{4} W_t$, since more than half of the weight gets slashed by $\beta = \frac{1}{2}$.

Since $h^* \in H$: $\underbrace{\left(\frac{1}{2}\right)^{\Delta^*}}_{\text{one expert}} \leq \underbrace{n\left(\frac{3}{4}\right)^{\Delta^{wm}}}_{\text{sum of all other experts}} \longrightarrow$ solve for $\Delta^{wM} \leq \frac{\Delta^* + \log_2(N)}{-\log_2\left(\frac{3}{4}\right)}$

L#21 Clustering: Similarity based Clustering

Review

- VC dim of H
  - largest set of examples that can be shattered by function from hypothesis space H.
  - Sour's lemma
  - Bound: $Err_p(h_{L(S)}) \leq Err_S(h_{L(S)}) + \phi\left(\sqrt{\frac{VC.dim(H) + \log S}{n}}\right)$

  for IID data

- Expert Setting
  - Halving Algorithm (perfect expert exist, 0/1 loss) $\rightarrow \log_2(|H|)$ mistakes
  - Weighted majority algorithm (0/1 loss) $\rightarrow \Delta^{WM} \leq 2.4\left(\Delta^* + \log_2(|H|)\right)$

    Δ^* = loss of experts in hindsight
  - Exponentiated Gradient Algorithm (bounded loss)

    $\hookrightarrow \Delta^{EG} \leq \Delta^* + O\left(\sqrt{n \log(|H|)}\right)$

    Divide by "n" to get upper bound on VC-dimension

    $$\frac{\Delta^{EG}}{n} \leq \frac{\Delta^*}{n} + O\left(\sqrt{\frac{\log(|H|)}{n}}\right)$$

- Bandit Setting
  - EXP3 algorithm

Supervised learning
- $P(X,Y)$ learning task
- i.i.d training sample $S = \{(x_1,y_1),\dots,(x_n,y_n)\}$
- learn $h: X \to Y$

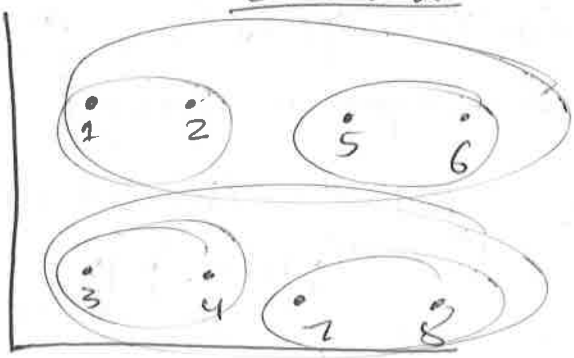
Clustering (trying to find structure in $P(X)$)
- $P(X)$
- i.i.d sample $S = \{(x_1)\dots(x_n)\}$
- To learn: structure of $P(X)$

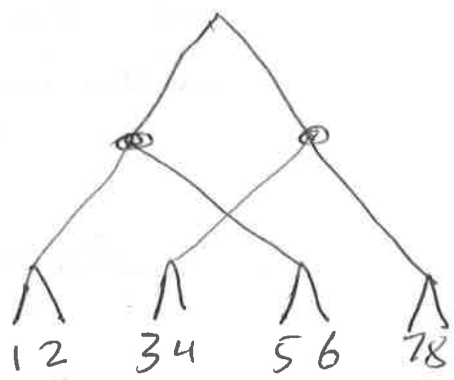
Outlier Detection:
- $P(X)$: we don't know
- i.i.d sample $S = (x_1,\dots x_n, x'_1,\dots x'_m)$
- find x' that did not come from $P(X)$

Novelty Detection:
- $P(X)$ and $P(X')$
- $S = ((x_1),\dots(x_K), (x'_1),\dots,(x'_m))$

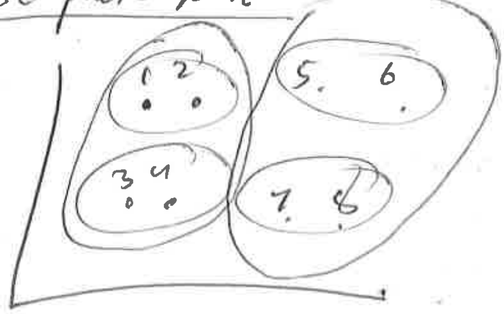
  we want to find where the switch happens
- find locations of change

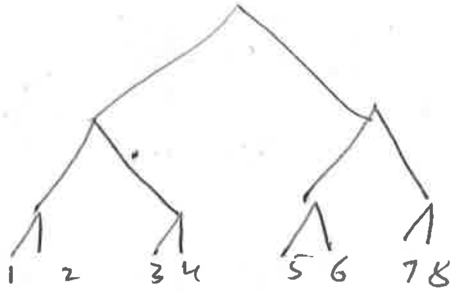
Close-Link

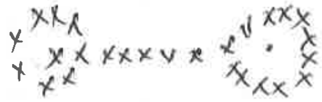




Complete link







L#22: Clustering: K-Means & Mixture of Gaussians

Review:

- Unsupervised learning → Clustering (finding structure in distribution)
- Hierarchical Aggl Clustering
  - start with instances, then merge
  - → Merging gives dendogram (tree)
- Similarity among Clusters
  - → single link (similarity b/w two most similar members of clus
  - → complete-link (" " " least " " "
  - → group average
- $O(n^2 \log n)$ for single/complete link

Non-Hierarchical (flat) Clustering → KMean
→ GMM with EM

Objective function of KMeans

$$\min_{C=\{C_1 \cdots C_K\}} \left\{ \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \right\}$$

It is redundant → $\mu_1, \dots, \mu_K$

Step 1: Update centroids

Given $C=\{C_1, \dots C_K\}$, $\left\{ \min_{\mu_j} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \right\}$ → solution of this is centroid. You can take derivative & set to 0

$$\min_{x_i \in \mu_j} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

→ Here K-Means improves objectives

Step2: Assign points to closest mean

→ K-Means improves objective

In both steps, K-Means improves objective function

Theorem: K-Means always terminates

Proof: finite number of clusters and no cycles since objective always improves.

Clustering as Prediction

→ $P(X)$ unknow

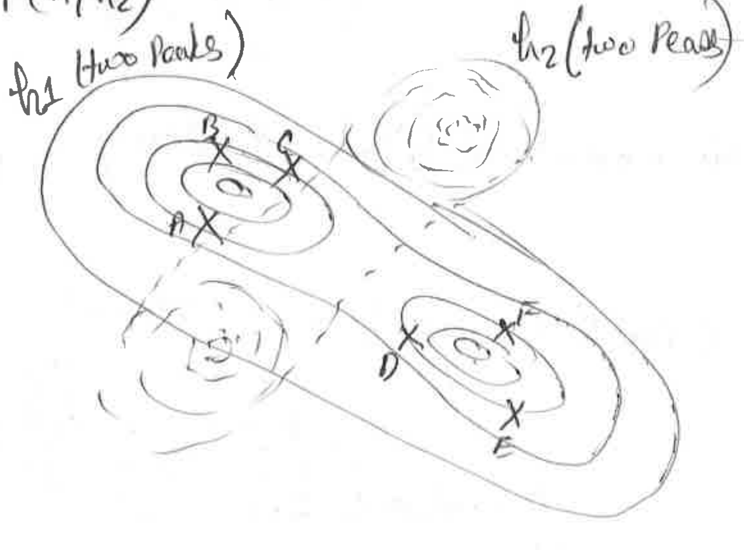
→ $S = (x_1, \dots x_m)$

→ $H = \{h_1, \dots, h_{|H|}\}$ where h_i is different clustering of S.

We need to ~~as~~ find which h_i in H generated our data S?

Example: Pick b/w two hypothesis:

$P(X|h_1)$ and $P(X|h_2)$



i) whats the prob that h_1 generated A, B, C, D, E, F?
Ans: Very high

ii) Whats the probability that h_2 generated A … F
Ans: Very low

why is h_1 better than h_2 ?

$$P(S|h_1) = \prod_{i=1}^{n} P(x_i|h_1) \quad \text{pretty high}$$

$$P(S|h_2) = \prod_{i=1}^{n} P(x_i|h_2) \quad \text{very low}$$

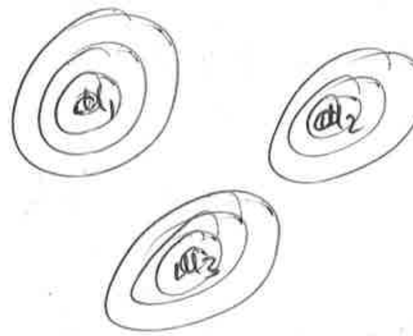
$$P(S|h) = \text{likelihood function}$$

Strategy

we will pick h_i with maximum likelihood. Maximum likelihood inference is very ~~g~~ widespread in statistics.

Generative Process of Mixture of Gaussians

lets assume mixture of 3 Gaussian distributions.



~~Gauss~~ GMM will be parametrized by $(\mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3)$

We will toss a dice (three sides), ~~then we~~ with probability $P(y=j)$. Then we will select that particular mixture and sample point x_i from that Gaussian $N(x_i; \mu_j, \Sigma_j)$.

GMM

$$h = (\mu_1, \mu_2 \ldots \mu_K, \Sigma_1, \ldots, \Sigma_K)$$

has to be estimated by EM

* EM is used for latent variable models.

GOAL: Find $\mu_1, \dots, \mu_K$ for which sample "S" has maximum likelihood.

$$\max_{\mu_1,\dots,\mu_K} P(S|\mu_1,\dots,\mu_K) = \max_{\mu_1 \dots \mu_K} \prod_{i=1}^{n} \sum_{j=1}^{K} P(X=x_i | y=j, \mu_j) P(y=j)$$

$$\frac{d}{d\mu_j}(obj) = 0$$

$$\forall j \ \frac{d}{d\mu_j}\left[\prod_{i=1}^{n}\sum_{l=1}^{K} P(X=x_i|y=l,\mu_l) P(y=l)\right] = 0$$

$$\Leftrightarrow \forall j \ \left[\sum_{i=1}^{n} \frac{d}{d\mu_j}\left[\ln\left(\sum_{l=1}^{K} P(X=x_i|y=l,\mu_l) P(y=l)\right)\right]\right] = 0$$

$$\Leftrightarrow \forall j \ \sum_{i=1}^{n} \frac{1}{\sum_{l=1}^{K} P(X=x_i|y=l,\mu_l) P(y=l)} \frac{d}{d\mu_j}\left[\sum_{l=1}^{K} P(X=x_i|y=l,\mu_l) P(y=l)\right]$$

$$\Leftrightarrow \forall j \ \sum_{i=1}^{n} \frac{1}{\dots} \frac{d}{d\mu_j}\left[P(X=x_i|y=l,\mu_j) P(y=l)\right]$$

$$\Leftrightarrow \forall j \ \sum_{i=1}^{n} \frac{P(X=x_i|y=l,\mu_j)P(y=l)}{\dots} \times \frac{d}{d\mu_j}\left[\frac{1}{2}(x_i-\mu_j)^2\right] = 0$$

$$P(X=x|\mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2}$$

with $\sigma = 1$

$$\Leftrightarrow \forall \sum P(y=l|X=x_i,\mu_j)(x_i-\mu_j) = 0$$

→ Solve for μ_j

$\mu_j = \dots$ in slide

μ_j = weighted average: we are weighting the points by probabilities. In K-Means, we were measuring distance

- if we assign probability $P(Y=j|X=x_i,\mu) = 1$ to the cluster it ~~assign~~ belongs, we can turn EM to K-Means. In EM, we have soft assignment of clusters rather than hard assignment of K-Means.

L#23: Structured Output Prediction

* Can we learn HMM in discriminative fashion?

$$P(x,y) = P(y_1)\cdot P(x_1|y_1)\cdot \prod_{i=2}^{l} P(y_i|y_{i-1})\,P(x_i|y_i)$$

Apply log

$$\log P(x,y) = \log P(y_1) + \log P(x_1|y_1) + \sum_{i=2}^{l}\left[\log P(y_i|y_{i-1}) + \log P(x_i|y_i)\right]$$

$$= w^T\phi(x,y)$$

"w" will be the log probabilities

If "m" symbols

$$\phi(x,y) = \begin{array}{c|l} \begin{bmatrix} \text{Start Freq} \\ \text{Trans Freq} \\ \text{Freq} \end{bmatrix} & \begin{array}{l} \text{start } S_1 \\ \text{start } S_2 \\ \vdots \\ \text{start } S_m \\ S_1 \to S_1 \\ S_1 \to S_2 \\ \vdots \\ S_m \to S_m \\ \left.\begin{array}{l} S_1 - \text{the} \\ \vdots \end{array}\right\} \text{emission} \end{array} \end{array}$$

In this way ~~$w^T\phi(x,y)$~~ $\arg\max_y w^T\phi(x,y)$

$= \arg\max_y P(y|x)$

* Example

The/Det bank/N opens/V

$$\phi(x,y) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{array}{l} \text{DET} \\ \text{DET} - \text{N} \\ \text{N} \to \text{V} \\ \text{DET} - \text{the} \\ \text{N} - \text{bank} \\ \text{V} - \text{opens} \end{array} \cdot W$$

- the/Det bank/V opens/V

$$\phi(x,y) = \begin{pmatrix} 1 - \text{Det} \\ 1 - \text{Det} \to \text{N} \\ 1 - \text{V} \to \text{V} \\ 1 - \text{Det} - \text{the} \\ 1 - \text{Bank} \to \text{bank} \\ 1 - \text{V} \to \text{opens} \end{pmatrix}$$

Different than above $\phi(x,y)$

Thus we get linear discriminative function and we will minimize error on training set.