

ML concepts and pitfalls

Eitan Farchi

IBM Haifa Research Laboratory

2017

Machine learning concepts

Learning - a simple setting

Let X be a domain. An adversary chooses a labeling function $f : X \rightarrow \{0, 1\}$ and a distribution D over X . Simultaneously, the learner chooses m and is given m examples $(x_1, f(x_1)), \dots, (x_m, f(x_m))$. $x_i \in X$ are m samples drawn independently using D . The learner deduce a labeling function $h : X \rightarrow \{0, 1\}$ and wants to minimize $Pr_D(h(x) \neq f(x))$.

Generalization is a must

The adversary picks a continuous distribution D over an infinite domain X , i.e., the probability of choosing a set of finite number of points from X is zero. The learner specifies m and sets $h(x) = f(x)$ if $x = x_i, i = 1 \dots m$ and $h(x) = 0$ otherwise. The adversary then chooses $f(x) = 1$ for $x \neq x_i, i = 1, \dots m$. We get that $Pr_D(h(x) = f(x)) = 0$. Thus, probability of making a mistake is 1^1 .

¹The adversary could have chosen any labeling function, f , and any distribution, D , such that $Pr_D(f(x) = 1) = \frac{1}{2}$. As a result, we would get that $Pr_D(h(x) = f(x)) = \frac{1}{2}$.

Learning a threshold on $[a, b] \subset \mathcal{R}$

- ▶ An ERM strategy for learning a threshold classifier
- ▶ The intermediate value theorem
- ▶ We need points close to the threshold
- ▶ The sample size should be big enough to ensure that
- ▶ Recapping with an updated definition of learnability

An ERM classifier for a threshold function

$X = [a, b] \subset \mathbb{R}$. Prior knowledge indicates that $f \in \text{threshold}(X)$ where $\text{threshold}(X) = \{h_t : X \rightarrow \{0, 1\} : h_t(x) = 0 \text{ if } x \leq t \text{ and } 1 \text{ otherwise}\}$.

We focus on the performance of an ERM classifier that after obtaining m points $x_i \in [a, b], i = 1, \dots, m$, chooses an $h_t \in \text{threshold}(X)$ such that $\sum_{i=1}^m \mathbf{1}_{h(x_i) \neq f(x_i)}$ is zero.

Intermediate value theorem

Given a continuous function $f : [a, b] \rightarrow \mathbb{R}$, and given that $c \in f([a, b])$, there exist a $d \in [a, b]$ such that $f(d) = c$.

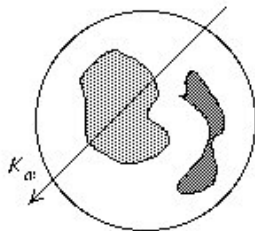


Figure 1: Intermediate Value

We need to ensure that sample points hit close to f

Let $\frac{1}{2} \geq \varepsilon > 0$. As $f \in \text{threshold}([a, b])$, $f = h_t$ for some $t \in [a, b]$. It is not possible for both $Pr([a, t])$ and $Pr([t, b])$ to be less than ε as that will mean that $Pr([a, b]) < 1$. B.l.g., $Pr([a, t]) \geq \varepsilon$. Assume the distribution is continuous², then by the intermediate value theorem we have that there exist an $r \in [a, t]$ such that $Pr([r, t]) = \frac{\varepsilon}{2}$. We want at least one sample point in $[r, t]$. Similarly we have a point $s \in [t, b]$ such that $Pr([t, s]) \leq \frac{\varepsilon}{2}$. Again we want a point in the sample to hit $[t, s]$. If that happens the ERM will choose h_q with $q \in [r, s]$. As a result $Pr(h_q(x) \neq f(x)) \leq \varepsilon$ will hold.

²i.e., that it has a continuous density function

Enough samples will ensure that we hit close to f

The probability of not hitting $[r, t]$ in the sampling process m times is $(1 - \frac{\epsilon}{2})^m \leq e^{-\frac{\epsilon}{2}m}$ which we want to be less than $0 < \delta < 1$. We get that $e^{-\frac{\epsilon}{2}m} < \delta$ or that $m \geq \frac{-2\ln(\delta)}{\epsilon}$

Enough samples will ensure that we hit close to f

The probability of not hitting both $[r, t]$ and $[t, s]$ is bounded by the union of these probabilities, i.e., $2(1 - \frac{\epsilon}{2})^m$ that should be less than δ which gives us our final requirement on m .

Learning - a simple setting - updated

Let X be a domain. An adversary chooses a labeling function $f : X \rightarrow \{0, 1\}$ and a distribution D over X . Fix a hypothesis set of functions H , such that $h \in H$ is a function from X to $\{0, 1\}$. Assume $f \in H$. We say that f is learnable if for any $\varepsilon, \delta \in (0, 1)$, the learner may choose a sample of size of at least $m(\varepsilon, \delta)$ such that with probability of at least $1 - \delta$ on the sample, the probability of error of the h the learner learned is $Pr_D(h(x) \neq f(x)) \leq \varepsilon$. In other words the "payment" to the adversary will not exceed ε almost certainly regardless of the choice of labeling function from H and distribution function. We have proved that the class of threshold functions on $[a, b] \subset R$ is learnable.

On the impossibility of learning - no free lunch theorem

- ▶ Learning the set of all functions for a finite domain is not possible - the setting
- ▶ Learning the set of all functions for a finite domain is not possible - the proof
- ▶ An infinite domain example

Learning the set of all functions - the setting

We are given a finite domain X and a model H of all functions from X to $\{0, 1\}$, i.e., $H = \{h : X \rightarrow \{0, 1\}\}$.

The learner requires a sample size $m \leq \frac{|X|}{2}$. The adversary picks a uniform distribution D over X , chooses f randomly on H and provides m samples $S = ((x_1, f(x_1)), \dots, (x_m, f(x_m)))$ i.i.d sampled. Note that at least $\frac{|X|}{2}$ points are not sampled.

Learning the set of all functions is not possible - the proof

Given a sample S , and a choice made by the learning algorithm $h \in H$. The probability of making a mistake is at least the probability of making a mistake on a point not in the sample, i.e., at least on $\frac{|X|}{2}$ points from X . A point not in the sample is chosen in probability at least $\frac{1}{2}$.

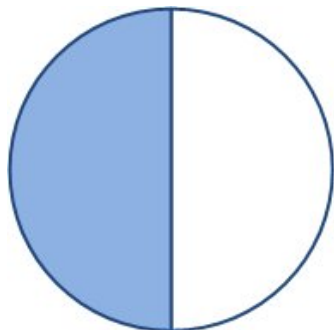


Figure 2: $\frac{|X|}{2}$ of the points in X are not sampled

Learning the set of all functions is not possible - the proof

Next the probability of making a mistake once the learner fixed her learned function $h \in H$ and the adversary chooses a point not in the sample is $\frac{1}{2}$ as the adversary chooses a function in H at random. Thus, the probability of an error is at least $\frac{1}{4}$. As we averaged over H there exists a function $h \in H$ for which the probability of error is at least $\frac{1}{4}$ ³

³Otherwise, all choices of functions from H will give an error less than $\frac{1}{4}$ and therefore the average will not be more than $\frac{1}{4}$.

Not learning an infinite set

Let $X = N$. Let $H = \{h : N \rightarrow \{0, 1\} \mid h(x) \neq 0 \text{ on a finite number of points}\}$. Then for any finite subset $A \subset N$, all functions on A can be obtained by projecting H to A . Applying the previous argument yields that the adversary can make the learner error with probability at least $\frac{1}{4}$ by restricting the probability to A . As this is done for any finite A , the adversary can cause the learner to error with probability at least $\frac{1}{4}$ no matter how big the required sample by the learner is.

The VC dimension

- ▶ Shattering sets
- ▶ The VC dimension
- ▶ Revisiting the no learning examples "through the eyes" of the VC dimension

Shattered Sets

Given a domain X and a model H , a set $A \subset X$ is shattered by H if for any labeling of A , $l : A \rightarrow \{0, 1\}$, there exists a function $f \in H$ such that $\forall x \in A, f(x) = l(x)$.

Below $X = R^2$ and H are all of the half spaces in R^2 .

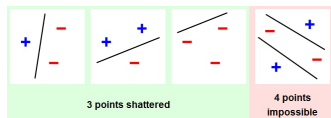


Figure 3: Shattered and non shattered sets

VC dimension

Given a domain X and a model H , the VC dimension of H is the maximal number d such that $\exists A \subset X$ such that $|A| = d$ and A can be shattered by H if such number exists and ∞ otherwise.

In our running example below we learned that the VC dimension of half spaces in R^2 is 3.

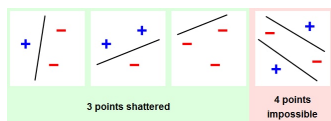


Figure 4: VC dimension of half spaces in

Revisiting the no learning examples "through the eyes" of the VC dimension

- ▶ For a finite set X and a model H of all labeling functions over X , the VC dimension is $|X|$
- ▶ For N and a model H of all finite 1 labeling functions, the VC dimension is ∞

In the first case above it was hard and the second not possible to learn. Thus, the VC dimension captures how hard learning is.

Learning, search problems and convexity

- ▶ Learning algorithms implements a search for $h \in H$ that minimizes some lose function resulting in a search problem
- ▶ An example of a search problem
- ▶ The challenge of local minima
- ▶ Convex sets
- ▶ Convex functions
- ▶ For a convex function every local minima is a global one

Learning algorithms minimize some loss function

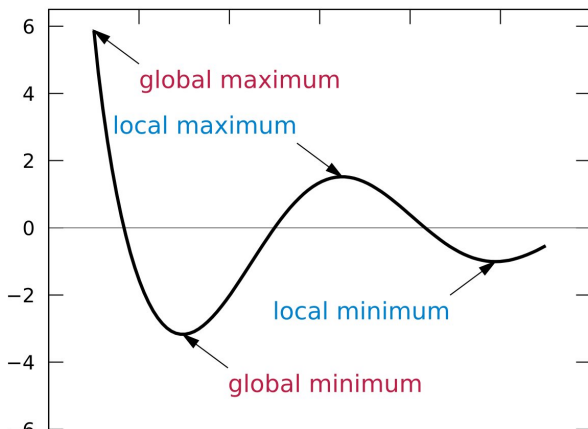
- ▶ Typically we want to have a "simple" explanation that minimizes the error on the sample space
- ▶ For example, if the real explanation is half a space we want to avoid explaining the data using a polygon (called overfitting, more on this latter)

An example of a search problem

- ▶ Given a function $f : R \rightarrow R$, find its root, i.e., $x \in R$ such that $f(x) = 0$.
- ▶ Iterate on the following search process $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$ (called Newton's method)

The problem of Local minima

- ▶ Given a function $f : R^n \rightarrow R$, $u \in R^n$ is a local minima if there exists $r \in R$, $r > 0$ such that $\forall x \in R^n$ with $\|x - u\| < r$, $f(x) \geq f(u)$.
- ▶ A search process may converge to a local minima but our objective is the global minimum of the lose function



Convex sets and convex functions

- ▶ $A \subset R^n$ is convex if $\forall x, y \in A$ and any $0 \leq a \leq 1$, $ax + (1 - a)y \in A$.
- ▶ For any set $B \subset R^n$, there is a minimal set $\text{cov}(B)$ that contains B and is convex. It is the intersection of all convex sets that contains B which is non empty as R^n is convex and contains B .
- ▶ A convex function $f : R^n \rightarrow R$ is convex if $f(ax + (1 - a)y) \leq af(x) + (1 - a)f(y)$ for $x, y \in R^n$ and $0 \leq a \leq 1$.
- ▶ For $f : R^n \rightarrow R$, if f is convex the set $y \in R$ such that $y \geq f(x)$ for some $x \in R^n$ is convex

Local minima are global for convex functions

Assume u is a local minima. Then we have $r > 0$ such that $f(u) \leq f(x)$ for $\|x - u\| \leq r$. Take a point $v \in R^n$, and chose an a so that $u + a(v - u)$ is of distance less then r from u ⁴. We have that $f(u) \leq f(u + a(v - u)) \leq (1 - a)f(u) + af(v)$. Thus $f(v)$ is greater or equal $f(u)$

⁴Formally prove using the intermediate theorem tat such a a exists.

Search algorithms

Next we'll cover some well known search problems over the loss function utilized in many learning techniques (e.g., regression, deep learning, etc.)

- ▶ Gradient Descent
- ▶ Newton's method revisited

Gradient Descent

- ▶ For clarity we'll present GD for two dimensions. Given a loss function $f : R^2 \rightarrow R$, the gradient of f , for $\theta \in R^2$ is given by $\nabla f(\theta) = (\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2})$
- ▶ Gradient decent is then defined by $\theta_k = \theta_{k-1} - r\nabla f(\theta)$ which updates θ at the direction of $\nabla f(\theta)$ at the rate of r
- ▶ r is a hyper parameter. Linear search is a technique for searching for the best r
- ▶ Newton's method "chooses the rate" but at the cost of inverting a potentially big matrix (in the number of features, the Hessian)⁵

⁵More about that shortly.

Gradient descent motivation

The gradient is chosen because it is perpendicular to a contour of equal height on f . In details, given a contour $\theta : [0, 1] \rightarrow \mathbb{R}^2$, $\theta(t) = (\theta_1(t), \theta_2(t))$ such that $f(\theta_1(t), \theta_2(t)) = c$ then $\frac{d_x f(\theta(t))}{dt} = \frac{\partial f}{\partial \theta_1} \theta_1'(t) + \frac{\partial f}{\partial \theta_2} \theta_2'(t) = 0$ and therefore ∇f is perpendicular to $\theta(t)$

Validation/testing that we learned correctly

- ▶ ML based system testing
- ▶ Single ML component testing
 - ▶ Probability concentration
 - ▶ Boosting TBC

ML based system testing

- ▶ A software system is built by recursive decomposition of the system to components
 - ▶ Each component provide interfaces and has well defined inputs and outputs with a known expected behavior
- ▶ In contrast a ML based system is compost of two types of components - ML components and traditional software components
 - ▶ A ML component input is typically of the same type of its training set its output is a random variable
 - ▶ It is implicitly assumed that the input is applied to the system using the same distribution that was used in the training phase and actually this is PART OF WHAT WE NEED TO TEST
- ▶ When testing a system interface at any level of abstraction one needs to determine if the input or part of it is translated to a ML training set format and if it does how that is done
 - ▶ In the next slides we will focus on the simplest scenario in which the interface being tested is a ML interface and the inputs are of the same type as the ML algorithm training set

What if we knew the distribution? and by the way we can estimate it!

- ▶ Loss function - Decision Theory, Hypothesis test, Game Theory, learning and a word on their respective optimization problems
- ▶ Had I known some probabilities what would be the best classifier?
- ▶ OK, so why isn't ML reduced to classical estimation problems of probability distributions?

Loss function

- ▶ Recalling that we are given $x_1 \dots x_n$ chosen at random by an adversary from X . In addition, the adversary chooses $f : X \rightarrow Y$. Our purpose is to come up with $f_l : X \rightarrow Y$ that "well represents" f
- ▶ Given an $x \in X$ our loss function is a $L(f(x), f_l(x))$ (sometimes referred to as a loss function)
- ▶ Think of the loss function as the payment the learner pays to the adversary

Loss function - an example

- ▶ Learning problem - identify if a test failure indicates a fault in the test/environment, T , or a fault in the program under test, P . Thus, $Y = \{T, P\}$
- ▶ Developers are more expensive than testers for concreteness say they are twice as expensive. Thus, we could have the following loss function, $L(P, T) = 1$, $L(T, P) = 2$ and otherwise 0

Loss function - Decision Theory, Hypothesis test, Game Theory, learning and a word on their respective optimization problems I

- ▶ Learning - the adversary may choose any distribution and any function. The learner determines the minimal size of the training set n , so that the average loss $E(L(f(x), f_l(x)))$, given the distribution chosen by the adversary is bounded
- ▶ Game Theory. Game in extensive form (not a matrix game). Adversary chooses a distribution and a function $f : X \rightarrow Y$. Learner, without knowing the choices of the adversary, chooses the size of the training set, n . Nature chooses n i.i.d samples $(x_1, y_1), \dots, (x_n, y_n)$. The learner chooses f_l . Trainer pays the adversary $E(L(f(x), f_l(x)))$. We would like to find the Nash equilibrium of this game⁶

⁶Haven't seen this type of work yet

Loss function - Decision Theory, Hypothesis test, Game Theory, learning and a word on their respective optimization problems II

- ▶ Decision theory and hypotheses testing. Assume some knowledge on the probability distribution capture by a family of distributions and chose a function to minimize the loss under that assumption
 - ▶ In hypothesis testing there are just two alternative distributions you need to chose from

Had I known some probabilities what would be the best classifier?

- ▶ Slightly changing the framework. We are given a pair of domain and its possible classification (X, Y) . The adversary chooses some distribution P over (X, Y)
 - ▶ For a discrete probability P , $P(y|x)$ is the probability that x is labeled y . We thus have $\forall x \in X, \sum_y p(y|x) = 1$
- ▶ We'll assume that Y is finite
- ▶ The learner chooses $f_l : X \rightarrow Y$. Consider the indicator loss function $L(y_1, y_2) = 1$ iff $y_1 \neq y_2$ and 0 otherwise
- ▶ We get that the average loss which we want to minimize is $E_P(L(y, f_l(x)))$

Had I known some probabilities what would be the best classifier? II

- ▶ We'll calculate for a discrete variable⁷
- ▶ $E_P(L(y, f_I(x))) = \sum_{(x,y)} L(y, f_I(x))P(x, y) = \sum_{x \in X} \sum_{y \neq f_I(x)} P(x, y)$
- ▶ For any $x \in X$ denote by $y_{max}(x)$ a $y \in Y$ such that $P(x, y_{max}) \geq P(x, y)$ for any $y \in Y$
- ▶ Had we known P we could set $f_I(x) = y_{max}(x)$ and we will get that $\sum_{x \in X} \sum_{y \neq f_I(x)} P(x, y) \geq \sum_{x \in X} \sum_{y \neq y_{max}(x)} P(x, y)$

⁷For a continuous variable change the sum to an appropriate integral everywhere

OK, so why isn't ML reduced to classical estimation problems of probability distributions?

- ▶ Typical $X = Z^d$. We will need to estimate P for each $x \in X$ which is exponential in d
- ▶ Back to combinatorial design of statistical experiments...

How to estimate a parameter of a family of distributions and why?

- ▶ We have just learned that estimating a probability may be an important tool in designing learning techniques
- ▶ Maximum likelihood estimation - the technique
- ▶ Maximum likelihood estimation - some examples
- ▶ Using maximum likelihood estimation to derive naive Bayes classifiers

Maximum likelihood estimation - the technique

We assume (capture our knowledge about the learning problem) that our training set $D = (x_1, \dots, x_n)$, $x_i \in R^d$ are sampled i.i.d from some probability distribution $P(\theta)$, $\theta \in \Theta$. θ is unknown and we are trying to estimate it. A natural approach is to solve $\theta_{mle} = \operatorname{argmax}_{\theta \in \Theta} P(D|\theta)$ choosing a parameter that is most likely to give the training set D .

As the training set was sampled from an i.i.d we have that $\theta_{mle} = \operatorname{argmax}_{\theta \in \Theta} P(D|\theta) = \operatorname{argmax}_{\theta \in \Theta} \prod P(x_i|\theta)$

MLE example

We assume that the training set, D , is obtained from a normal probability with density function $\frac{1}{\sqrt{(2\pi\sigma^2)}} e^{-\frac{(x-\theta)^2}{2\sigma^2}}$ where the average θ is the unknown parameter. We thus get that

$$p(D|\theta) = p(x_1, \dots, x_n|\theta) = \prod_1^n p(x_i|\theta) = \prod_1^n \frac{1}{\sqrt{(2\pi\sigma^2)}} e^{-\frac{(x_i-\theta)^2}{2\sigma^2}}.$$

Taking the derivate wrt to θ to find the maximum. The log function is monotonic increasing so maximizing the log of the above function will yield the same maximum.

$\log(D|\theta) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_1^n (x_i - \theta)^2$. Taking the derivate wrt to θ and equating to zero yields $\theta = \frac{1}{n} \sum_1^n x_i$.

MLE example - TBC

Bayesian networks, plausible inference and learning

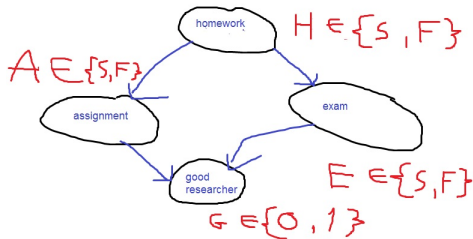
- ▶ An approach to learning and how it ties to classical statistic inference
- ▶ Bayesian network definition and example
- ▶ Conditional independence in Bayesian networks
- ▶ d-separation and hidden Markov models
- ▶ Using the model - query a Bayesian network

Bayesian network definition and example I

Consider the Bayesian network below. The network captures the assumption that a node is only conditional dependent on its parents. Concretely this is expressed in the following calculation.

By the law of probabilities we have

$P(H, A, E, G) = P(H)P(A|H)P(E|A, H)P(G|HEA)$. Applying the dependence assumptions of the network this reduces to $= P(H)P(A|H)P(E|H)P(G|AE)$. In words doing well in the exam is only dependent on doing the homework well, same for the final assignment. Being a good researcher is dependent on doing the final assignment well and doing well in the exam.



Bayesian network definition and example II

$P(H)$

S	F
P_S^H	P_F^H

$P(E|H)$

H \ E	S	F
S	$P_{S S}^E$	$P_{S F}^E$
F	$P_{F S}^E$	$P_{F F}^E$

$P(A|H)$

H \ A	S	F
S	$P_{S S}^A$	$P_{S F}^A$
F	$P_{F S}^A$	$P_{F F}^A$

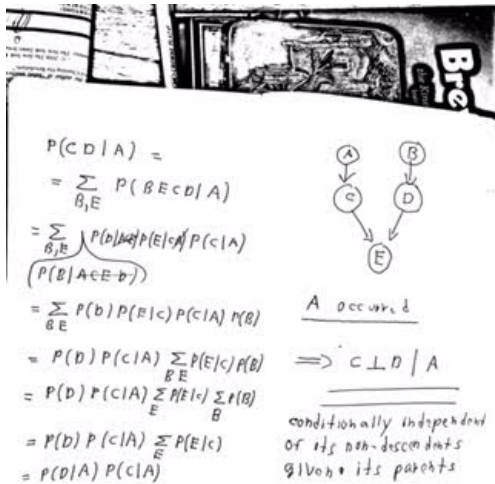
$P(G|A, E)$

A \ E	0	1
S	$P_{S 0}^G$	$P_{S 1}^G$
S	$P_{S 0}^G$	$P_{S 1}^G$
F	$P_{F 0}^G$	$P_{F 1}^G$
F	$P_{F 0}^G$	$P_{F 1}^G$

Conditional independence I

- ▶ It is desirable to determine the conditional independence of variables given some observed events. This simplifies the estimation of other events (that did not occur yet)
- ▶ One easy observation is that a variable is conditionally independent of its non-descendants given its parents. A detailed example of this is given in the next slide.

Conditional dependency II



$$P(CD|A) =$$

$$= \sum_{B,E} P(BE|CD|A)$$

$$= \sum_{B,E} \underbrace{P(D|A)P(E|C)P(C|A)}_{P(B|A \in E \neq b)}$$

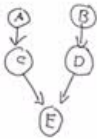
$$= \sum_{B,E} P(B)P(E|C)P(C|A)P(B)$$

$$= P(D)P(C|A) \sum_{B,E} P(E|C)P(B)$$

$$= P(D)P(C|A) \sum_E P(E|C) \sum_B P(B)$$

$$= P(D)P(C|A) \sum_E P(E|C)$$

$$= \underline{P(D|A)P(C|A)}$$



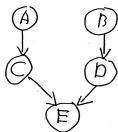
```

    graph TD
      A((A)) --> C((C))
      B((B)) --> D((D))
      C --> E((E))
      D --> E
    
```

A occurred
 $\Rightarrow \underline{\underline{C \perp D | A}}$
 conditionally independent
 of its non-descendants
 given its parents

Conditional dependency III

$$\begin{aligned}P(C \perp D | A B) &= \\&= \sum_E P(E \perp C D | A B) \\&= \sum_E P(E | A B C D) P(C | A B D) P(D | A B E) \\&= \sum_E P(E | C D) P(C | A) P(D | B) \\&= P(C | A) P(D | B) \sum_E P(E | C D) \\&= P(C | A) P(D | B) = \\&= P(C | A B) P(D | A B)\end{aligned}$$



A B occurred

$$\Rightarrow C \perp D | A B$$

Conditional independence V

Once we have deduced that some variables are independent we want to test for arrows directions wrt variables assumed to be dependent. Below given that B is independent of C we determine the expected conditional independence on A given different possible directions of the arrows.

$$\begin{aligned}P(B \perp C | A) &= \\P(B | AC) P(C | A) &= \\= P(B | A) P(C | A)\end{aligned}$$

$$\begin{aligned}P(B \perp C | A) &= \\P(B | AC) P(C | A) &= \\= P(C | AB) P(B | A)\end{aligned}$$

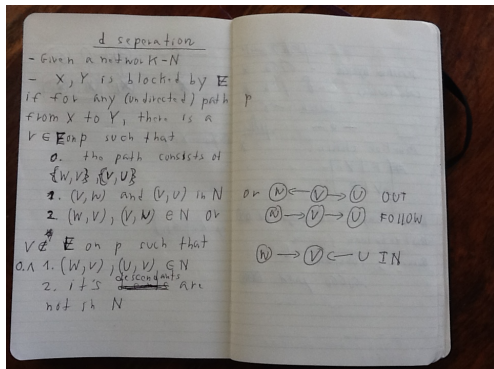
$$\begin{aligned}P(B \perp C | A) &= \\= P(B | AC) P(C | A) &= \\= P(C | A) P(B | A) &= \\= P(C | A) P(B) &= \\= P(C | A) P(B)\end{aligned}$$



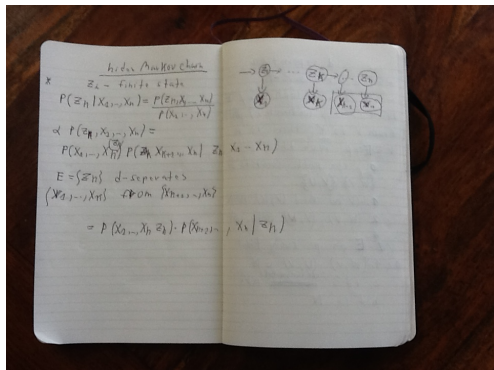
$B \perp C$



d-separation



Hidden Markov model



Naive Bayes

Naive Bayes

$(x_2, y_1), \dots, (x_n, y_n) \in \mathbb{R}^n$

$y_i \in \{0, 1\}, x_i \in \mathbb{R}^m, i = 1, \dots, n$

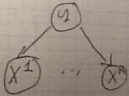
$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

$$P(x|y) = P(x^1, \dots, x^m | y)$$

$$P(x^1 | y) P(x^2 | y x^1) P(x^3 | y x^1 x^2) \dots$$

$$\dots P(x^m | y x^1 \dots x^{m-1}) =$$

Not!



\Rightarrow

$$= P(x^1 | y) P(x^2 | y) \dots P(x^m | y)$$

$$\Rightarrow P(y|x) = \frac{\prod_{i=1}^m P(x^i | y) P(y)}{P(x)}$$

200b density
2030 - A/on
2100 - 35503

~~Part 201~~

$P(x) =$
 $\rightarrow P(x|$
 $= \prod P(x^i | y)$

Quality of ML and data intensive solutions

Outline

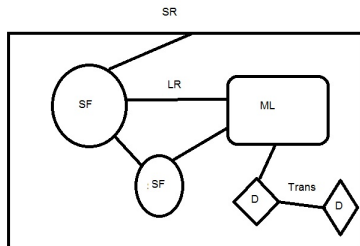
- ▶ Quality of ML and data intensive solutions
 - ▶ White box
 - ▶ Black box

Quality of ML and data intensive solutions

- ▶ Is ML quality determined by software quality?
 - ▶ Quality of software construction is a necessary but not sufficient condition to the quality of the ML/data solutions
- ▶ Instead its determined by
 - ▶ Marriage of clear quantitative business goals with data that contains the right information to realize them
 - ▶ Correct abstraction of the data that does not lose information
 - ▶ Training data is representative of the real life data and has the right size to guarantee successful learning
 - ▶ Correct choice of ML technique and hyper parameters maximizes the chance of high quality learning
- ▶ The workshop focuses on the above items and teaches
 - ▶ Pitfalls to learning
 - ▶ How to avoid them
 - ▶ Best practices and techniques to insure high quality of learning

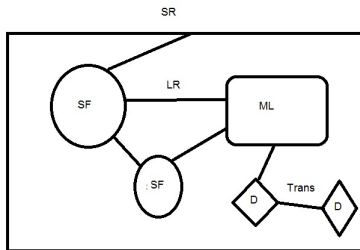
Black box testing challenges I

- ▶ Below is the architecture of a *ML* based system. The system is decomposed into:
 - ▶ A Machine learning component (*ML*). The *ML* component used the data (*D*) as its training set
 - ▶ The data *D* is pre-processed and sampled using a transformation function, *Tran*
 - ▶ The system has two software components (*SF*) that interact with the *ML* component and each other to obtain the system result
 - ▶ The *ML* training objective is defined at the *ML* component level (*LR*)
 - ▶ The system requirements, *SR*, are defined as a black box



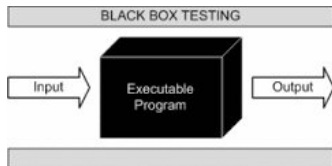
Black box testing challenges II

- ▶ The way the system is constructed and the learning requirements, LR , are defined may not realize the SR . In addition, the SR may not be quantifiable
- ▶ The data transformation, $tran$, may miss or abstract factors. The information needed to learn LR or SR is not present in the data.
- ▶ The system is decomposed and interacts in a wrong way resulting in not meeting the system requirement, SR . Interface to the ML component is not well defined
- ▶ Overtime data changes rendering the ML component useless



ML/data black box quality

- ▶ The marriage of data and business requirements is crucial to quality
- ▶ It should be monitored over time to identify drifts (e.g., in data distribution or probability of accurate classification)
- ▶ Drift may occur in the data, the business requirements or both. The following best practices apply:
 - ▶ Develop an independent of the learning model statistic test to determine that the data is drifting
 - ▶ Develop a traceability capability that determines the source of the data and identify that the number or type or distribution of the data sources changed
 - ▶ Determine when loss function of learning algorithm changes
 - ▶ Identify when the performance of some independent business value centered metric is degrading



ML/data white box quality

- ▶ ML loss function and feature vector composing the training set should correctly reflect the business objectives
- ▶ Data is abstracted and manipulated to obtain the feature vector composing the training set
 - ▶ Abstraction must not lose key features that determine the result of the learning function, or abstract their values in the wrong way
- ▶ Data is representative of the real life data. The generative process that created the data reasonably represents the way the data will be obtained in production time and does not introduce biases
- ▶ The data is big enough to successfully apply the ML technique we decided to apply
- ▶ Choice of ML technique and hyper parameters is the right one and maximizes the chance of high quality learning

White Box pitfalls and antidote

List statistical assumptions and find ways to check them

- ▶ ML approaches make statistical assumptions.
 - ▶ As an example naive Bayes assumes conditional independence, i.e., that $p(x|y) = p((x_1, \dots, x_n)|y) = \prod p(x_i|y)$ where x_i are the features and y is a class to be classified. In addition, the probability $p(x_i|y)$ may be assumed to be normal or Bernoulli
- ▶ Apply statistical reasoning to check the assumption
- ▶ Caution - you may still proceed with the approach even if the statistical assumptions are not completely met
 - ▶ Naive Bayes may be used although it is observed that the features are not independent as it simplifies the task of estimating n features space distribution to an estimation of one dimension feature space

Did you generalize?

- ▶ Separate the test data
- ▶ Consider the hypothesis set compared to the data
 - ▶ Is it too strong and are you going to overfit?
- ▶ Prefer simpler models that perform well
 - ▶ Use VC dimension to determine complexity of the model
 - ▶ Other heuristics to determine complexity? Number of dimensions? Number of model parameters? TBC
- ▶ For a given hypothesis, prefer a simpler classifier with the same Experimental Error, ERM, on the training set

Learning objective, business objective

- ▶ learning objective or loss function drives the optimization
- ▶ it should represent the business goals as much as possible
 - ▶ sometimes this is not possible and only an approximation is possible leading to the need for black box testing representing the business goals in addition to the ML performance against the test set that can be labeled as white box testing
- ▶ simple example of choosing the objective or loss function carefully. With spam classification the mail could be spam but classified as non spam. That is not too bad, you look at the mail and determine it's spam. On the other hand the mail can be non spam and a mail from your manager and you ignore it because its labeled as spam - that is bad. The loss function should account for that
 - ▶ objective function should be carefully design to represent the business objective (but also to simplify the optimization, e.g., by preferring concave loss functions when possible

Embedding knowledge is a must

- ▶ You must embed knowledge to achieve learning. - no free lunch
- ▶ This is done in two possible ways
 - ▶ Define a class of possible hypothesis to learn from
 - ▶ Define a family of distribution the data is obtained from
- ▶ A trade off is to be achieved
 - ▶ Given a trend observed for a segment of the entire space
 - ▶ The trend may be random or part of the function to be learned behavior
 - ▶ Two mistakes are possible (sometimes referred to as overfitting and underfitting)
 - ▶ Representing a random "trend" as part of the learned function. If the hypothesis class is too complex we have a higher chance of that error occurring resulting in over fitting
 - ▶ Not representing or ignoring a real trend of the function. If the hypothesis class is not complex enough we will not be able to represent some real trends e.g., if the trend is parabolic and the hypothesis class is linear
- ▶ Bottom line - we need to capture our knowledge in a "just right" hypothesis - not too complex and not too simple

Why overfitting is always possible

Consider our training set $(x_1, y_1), \dots, (x_n, y_n)$. It is always possible to define a polynomial that fits the data **perfectly**. For example, note that $p(i, x) = \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} x_i - x_j}$ is 1 if $x = x_i$ and 0 for $x_j, j \neq i$. Then $f(x) = \prod_{i=1 \dots n} p(i, x) y_i$ is a polynomial with $p(x_i) = y_i$.⁸

⁸See discussion on VC dimension latter at slide number 17. Try to answer the following questions in that context.

- ▶ What is the degree of $f(x)$?
- ▶ What is the VC dimensions of all polynomial up to the degree of $f(x)$?
- ▶ OK - what does that tell me about learning?

Over fitting and under fitting resulting from too much or not enough knowledge. Can be combated by TBC

- ▶ Cross validation e.g., in order to choose the size of a tree
- ▶ Regularization in order to choose a simpler classifier
- ▶ Significance test to check if two classifiers are different, e.g., chi- test

Cross validation I

- ▶ We are given n models M_1, \dots, M_k ⁹. Initially one may fit M_i to the training set $(x_1, y_1), \dots, (x_k, y_k)$ and pick up the best model, M_i , which will result in overfit if n is big enough¹⁰.
- ▶ In cross validation we choose between $\frac{1}{3}$ and $\frac{1}{4}$ of the training set and keep it as the test set. We next train on the rest of the training set and check our performance on the test set.

⁹For concreteness, think about all polynomials of degree $1, \dots, n$.

¹⁰Consider the following two questions:

1. How big should n be to result in an overfit?
2. Can this approach be modified in a way other than cross validation to work well?

Cross validation II

Some general observations on cross validation. See wikipedia for variation on the theme.

- ▶ The performance on the training set is a statistic that correspond to the performance on the real data
- ▶ If you choose the training set again and follow the same learning process you will get slightly different performance
- ▶ The performance statistic is slightly biased (its average is slightly different than the real average) due to the test set being a fraction of the training set
- ▶ Confidence interval for the performance on the test set is considered a hard problem

Cross validation - feature selection I

- ▶ If the number of features, n , is much bigger than the number of training set, m , even if you use a linear model to classify, its VC dimension will be $O(n)$, which is much bigger than m , resulting in potential overfit
- ▶ This may be augmented by the intuition that not all features are relevant...
- ▶ We thus need to choose a subset of the features

Cross validation - feature selection I

Denote the set of features as $F = \{F_1, \dots, F_n\}$. If n is not too big you could preform chose a subset of F and preform cross validation on it. When n is to big the following can be done.

1. Set $F' = \emptyset$
2. For $i = 1, \dots, n$, if $i \notin F'$, perform cross validation on $F' \cup \{i\}$.
3. Update F' to the best model obtained in the previous stage

Another option to order the features based on some information gain criterion.

Chose a representation that let you easily capture your knowledge and update it as needed - tbc with examples -

- ▶ Feature similarity - instance model ?
- ▶ Conditional probabilities - graph representation
- ▶ Precondition - if then models

Over fitting and under fitting resulting from too much or not enough knowledge. Canberra combated by TBC

- ▶ cross validation e.g., in order to choose the size of a tree
- ▶ regularization in order to choose a simpler classifier
- ▶ significance test to check if two classifiers are different, e.g., chi- test

High dimension pitfalls

- ▶ Assume we have 100 binary features
 - ▶ Classification is determined by the first two features
 - ▶ rest of the features are random
 - ▶ using say the hamming distance and k nearest to classify we will be fooled to determine close vectors that are not close — on average vectors will be close TBC make statement precise
- ▶ use of information gain to exclude features that are noise wrt the classification
- ▶ - sometimes the class you wish to learn focuses on a lower dimension sub space – utilize this implicitly or through explicit dimension reduction algorithm