Machine Learning

Mark

Coursera

Two types: supervised and unsupervised.

Supervised is most commonly applicable, where we know the data we're getting.
comes in two flavors:
classification based: yes/no values
regression based: continuim of values.

Unsupervised: model is not provided with the correct results during training; key is to find structure or clusters in data.

Gradient Descent

technique to find **local** minimum of function by taking small steps in steepest direction.
Take $f(x, y)$ with step size (learning rate) α.
Then, update x, y (simultaneously) using

$$\text{new } x = x - \alpha \frac{d}{dx} f(x, y) * f(x, y)$$

if α is too large, technique may not yield optimal value; too small, technique will be slow.

Linear Regression using gradient

can use gradient to find the least squares, minimize the squared error (called cost function).

in practice useful to rescale variables btw -1 and 1

*the squared error function supposedly only has one global min, so technique above succeeds at minimizing the error.

normal equation is an analytical method to find same result, but does not scale as nicely as gradient method.

Machine Learning

Robert Snapp

Intro

AI - rational machines (ML is often a subset)

Machine Learning improves performance as more data is fed into Algo.

Types of Machine Learning

Supervised Learning learn a mapping from input to output based on training set.

- Classification: mapping inputs to *discrete* outputs (categories) e.g., classifying a document as a web page or email.
- Regression: mapping inputs to *continuous* outputs.

Unsupervised Learning only output given with goal of finding "interesting patterns" (called "Knowledge Discovery").

- clustering
- Dimensionality Reduction. e.g., PCA
- Imputation ("matrix completion"): finding plausible values for missing data points.

*K-nearest neighbors (nearest K points to a given input) is a non-parametric classifier model.
Reinforcement Learning learning behavior based on reward/punishment signals.

Probability Review

Axioms

Formulated by Andrey Kolmogorov

a **probability space** is comprised of :

Ω: sample space, meaning the set of possible elementary events.
e.g., $\Omega = \{1, 2, ..., 6\}$

$\mathcal{A}$: all measurable sets of events
*can be something like odd numbers if we're rolling a die.
$\mathcal{P}$: probability

$\mathcal{A}$ must satisfy:
1. $\Omega \in A$
2. $E \in A \iff E^c \in A$
*complement in Ω
3. $E, F \in A \rightarrow E \cup F \in A$

$\mathcal{A}$ is a "sigma-algebra", σ-algebra
(simply properties/laws for operations on sets)

$\mathcal{P}$ assigns a numerical value for each $E \in A$ such that natural requirements:
$\mathcal{P}$(E) ≥ 0, $\mathcal{P}(\Omega) = 1$, and if $E, F \in A$ and $E \cap F = \emptyset$, then

$$\mathcal{P}(E \cup F) = \mathcal{P}(E) + \mathcal{P}(F)$$

$(\Omega, \mathcal{A}, \mathcal{P})$ defines a **probability space**.

two underlying assumptions:
pattern indicates a unique state of nature,
problem is driven by an underlying probability space.

Distributions

a **random variable** map from Ω to a "space of interest"?
a **probability mass function** of a random variable, X, is (where k is in "space of interest" produced by x)

$$P_x(k) = \mathcal{P}\{\omega \in \Omega : x(\omega) = k\}$$

e.g., X is the result of summing two dice.
(probability mass is like a probability density function,but for a discrete variable)

Then $P_x(7)$ the sum of the probability of all inputs (aka $\omega \in \Omega$) summing to 7.
The **cumulative probability distribution** (CDF) of a random variable X,

$$F_X(r) = \mathcal{P}\{\omega \in \Omega : X(\omega) \leq r\}$$

"cumulative density function"
Instantaneous probability, aka **probability density** is the derivative of the **probability distribution**.

Dice Example

Two dice:
$\Omega = \{(i,j) : i,j \in \{1,...,6\}\}$.
$|\Omega| = 36$
$\mathcal{A}$ = **power set** (all possible subsets) of Ω.
$|\mathcal{A}| = 2^{36}$ (each member of Ω has two states on/off)

Define a random variable $X(i,j) = \max(i,j)$. Then,
$P_x(1) = \mathcal{P}\{(i,j) : X(i,j) = 1\} = \frac{1}{36}$.
$P_x(2) = \mathcal{P}\{(i,j) : \max(i,j) = 2\} = \mathcal{P}\{(2,1),(1,2),(2,2)\} = \frac{3}{36}$.

For the rest, think about L shape in outcomes table.

Notation: $[x = 1] = \mathcal{P}\{(i,j) : X(i,j) = 1\}$.

Pattern Classification

Feature Space: observable trait like diameter of cell

Joint Distributions

Discrete

Two random variables:
X_1: max of rolling two dice
X_2: min.

$P_{x_1,x_2}(x_1,x_2) = \mathcal{P}\{\omega \in (i,j) : X_1(\omega) = x_1 \text{ and } X_2(\omega) = x_2\}$.

Then,

$$P_{x_1,x_2}(x_1,x_2) = \begin{cases} P(x_1,x_2) + P(x_2,x_1) & \text{if } x_2 < x_1 \\ P(x_1,x_2) & \text{if } x_1 = x_2 \end{cases}$$

Continuous

CDF: all variables $\leq$ number.

Let CDF $= F_{x_1,x_2}(x_1,x_2)$.
Then, PDF $= \dfrac{d^2}{dx_1 dy_1}$
(partial with respect to x_1 then differentiate with respect to y_1; order doesn't matter).

Combined: Discrete and Continuous

CDF as expected with discrete condition for discrete and similarly for continuous.

PDF: same as above with partial dervitives for contious and piecewise defined function for each discrete class.

Baye's Theorem

In a probability space, $(\Omega, \mathcal{A}, \mathcal{P})$.
For E, F events in A,

$$P(E|F) = \frac{P(E \cap F)}{P(F)}$$

(provided $P(F) \neq 0$).
*no independence necessary.

Why is $E \cap F$ in $\mathcal{A}$?
We know E^c and F^c in A.
So, $E^c \cup F^c$ in A.
Thus, so is $E \cap F$ by De Morgan's Law.

Baye's Rule

$$P(F|E) = \frac{P(E|F)P(F)}{P(E)}$$

e.g.,
ebola = E
H = high temp

$P(E|H) \approx 1$
$P(H|E)$ low, say 10^{-3}

can compute probability of ebola given fever.

HW

Snapp's paper before generating functions (4.4)

Independence

Events E, F are **independent events** if:

$$P(E \cap F) = P(E)P(F)$$

$E_1, E_2, ..., E_n \in \mathcal{A}$ are **independent events**:
For every combination of E_i, P(intersection) = P(first) *P(second) ...
"intersection distributes over multiplication"

Random variables are **independent** if for F CDF:

$$F_{x_1,...x_n}(x_1, ..., x_n) = F_{x_1}(x_1)F_{x_2}(x_2)...F_{x_n}(x_n)$$

Pattern Classification

In $(\Omega, \mathcal{A}, P)$, the process is:
1. seed from $\omega \in \Omega$ (elementary event
2. Feature Vector (observable trait/s)
3. L classification: maps ω to a state (like cancer no cancer)

Feature vector: $(x_1, ..., x_n)$.

Probability Distribution (combined discrete and continuous)
$F_{x,l}(x, l) = P(\omega \in \Omega : X \leq x \text{ and } L = l)$, joint probability

Using Baye's Theorem we have,

$$F(x|l) = \frac{F(x, l)}{P(l)}.$$

e.g., say we have l = 0 is sick, 1 healthy; x is diameter of cell.
$P(x|l = 0)$: normal distribution centered around c
$P(x|l = 1)$: normal distribution centered around -c.

note:

$$\begin{aligned} p(x) &= p(x, l = 0) + p(x, l = 1) \\ &= p(x|l = 0)p(l = 0) + p(x|l = 1)p(l = 1) \end{aligned}$$

What's the probability someone is sick based on x?

$$P(l = 1|x = c_0) = \frac{p(x = c_0|l = 1)p(l = 1)}{p(x = c_0)}$$

use expanded formula of normal curve to solve.
To classify a patient, compute propability of l=1 given x and l=0 given x. Higher probability leads to classification (called **Baye's Classifier**)

Denisties from the above

Joint Density
$P(x, l) = \frac{d}{dx} F_{x,L}(x, l)$
from joint denisty, we can derive all other info like conditional, or prior probability for class l.

prior class probability
(class probability density)
$P(l) = \lim_{x \to \infty} F_{x,L}(x, l)$

$P(l|x)$ **posterior probability**
Marginal Denisty

$$p(x) = \sum_{l=0}^{1} p(x, l)$$

Class Conditional Distribution

$$F(x|l) = \frac{P(X \leq x, L = l)}{P(L = l)}$$

probability of x falling within range given a value for l.
Using Baye's Theorem, we can reformulate as

$$P(l|x) = \frac{P(x, l)}{P(x)}$$

look up Baye's Classifier

Deriving other densities from join density

prior probability for class l

$$P(l) = \int_{-\infty}^{\infty} p(x, l) dx$$

$$P(x|l) = \frac{P(x, l)}{p(l)}$$

Probability of misclassifying

Suppose R_0 is a region of X the feature vector associated with $L = 0$. Similarly, R_1 is associated with $L = 1$. Then,

$$P(\text{error}) = P(x \in R_0, L = 1) + P(x \in R_1, L - 0)$$

$$= \int_{R_0} p(x, 1) dx + \int_{R_1} p(x, 0) dx$$

we can related the limits of integration via

$$P(1) = \int_{-\infty}^{\infty} p(x,1)dx = \int_{R_0} p(x,1)dx + \int_{R_1} p(x,1)dx$$

So,

$$P(\text{error} = P(1) + \int_{R_1} P(x,0) - P(x,1)dx$$

Spam Email Classifier: Naive Baye's

Trick: $p^x(1-p)^{1-x}$
if x, the feature vector is 1, brings in p^x; else just $1-p$.

From data we try to estimate $p(x,l)$, enabling us to compute

$$p(l|x) = \frac{p(x,l)}{p(x)}$$

For multidimensional $x_1, \ldots, x_n$ feature vectors:

$$p(l|x) = p(l|x_1) * \ldots p(l|x_n)$$

by **assuming each feature is independent** (that's the big assumption!).

Baye's Error: lowest possible error rate (by moving decision regions)

K-nearest Neighbor Classifier

Need to select :

1. k
2. metric (similarity function)
   e.g., euclidean distance
3. Training set

Idea is to find the k elements form the training data that are closest to an input x.
Then, classify input x based on the class label of the nearest k elements appearing most.

Discussion

Main assumption: proximity in feature space corresponds to similarity in class.

very popular algo

Trade-offs: Large k reduces variances (more samples), but makes classification less accurate.

*note method is **non-parametric** (doesn't assume an underlying distribution).

*useful rule of thumb: k scales with $\sqrt{\text{sample size}}$

Cover and Hart showed the error rate of K-nearest neighbors is no more than 2*Baye's error (missclassification based on conditional) rate for infinitely large samples.

Triangle Problem

$$p(x|0) = \begin{cases} 2x & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$p(x|1) = \begin{cases} 2-2x & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Looks like triangle on graph.

Cover and Hart, 1967 mathematical error in paper

Theorem Stone 1997:

sample size and nearest neighbor $\to \infty$,

k-neighbor error approaches baye's error

Regression

Linear

$y = \beta_0 + \beta_1 x$

we want find β_0, β_1 such that

$$\text{squared error} = \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)^2$$

is **minimized.**

Optimizing slope, intercept

Look at partial derivatives:

$$\frac{\partial}{\partial \beta_0} = 2\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i) = 0$$
$$\frac{\partial}{\partial \beta_1} = 2\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)x_i = 0$$

then solve this system of equations.

$$\begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} n & \sum x \\ \sum x & \sum x^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y \\ \sum xy \end{bmatrix}. \tag{2}$$

Multivariable Case

$B = (X^T X)^{-}1X^T y$ where all are vectors.

Linear Algebra Aside

gradient vector, denoted $\vec{\nabla} f$, is a vector,

$$< \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \cdots >$$

a **hyperplane** is an equation describing an **n-1 dimensional object** in n-dimensional **space** dividing the space into **two pieces**.

e.g., a point in 1-d, a line in 2-d, a plane in 3-d.

Fittting via any function

write as $Y = B * f(X)$, where $f(x)$ is the function we're using to fit

same results apply.
Solution: $B = (X^T X)^{-1} X^T y$ (y is point in n-space)

e.g., **cubic fit**: $y = b_0 + b_1 x + b_2 x^2 + b_3 x^3$
we optimize by

letting $x1, x_2, x_3, \ldots$ data points.
and $f_1(x) = x, f_2(x) = x^2, f_3(x) = x^3$.

Then, $X =$

$$\begin{bmatrix} 1 & f_1(x_1) = x_1 & f_2(x_1) = x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \ldots & & & \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \tag{3}$$

Finally solve $B = (X^T X)^{-1} X^T y$.

Variations

Local Regression increase weight of points near a point you're intested in predicting.
score * 1.25

Exam Solutions

1.

$$p(x, l) = \begin{cases} \alpha/\beta & \text{if } 0 \leq x \leq \beta \text{ and } l = 1 \\ (1-\alpha)/\beta & if 1 - \beta \leq x \text{ and } l = 2 \\ 0 & \text{otherwise} \end{cases}$$

with $\alpha, \beta \in [0, 1]$ and $L = 1$ or 2.
(a) got it; can also use b*h (base times height)
(b) Compute the posterior probability. We want to find

$$P(l|x).$$

By Baye's,

$$P(l|x) = \frac{p(x, l)}{p(x)}$$

First let's compute $p(x)$.

$$p(x) = p(x, 1) + p(x, 2)$$

Two cases:
1. $\beta > 1/2$:

$$p(x) = \begin{cases} \alpha/\beta & : 0 \leq x \leq 1 - \beta \\ \frac{1-\alpha}{\beta} + \alpha/\beta + & : 1 - \beta \leq x \leq \beta \\ \frac{1-\alpha}{\beta} & : \beta \leq x \leq 1 \end{cases}$$

2. $\beta < 1/2$:

$$p(x) = \begin{cases} \alpha/\beta & : 0 \leq x \leq \beta \\ \frac{1-\alpha}{\beta} & : 1-\beta \leq x \leq 1 \end{cases}$$

—

3. review gaussian, including second moment

Perceptron Algorithm

Linearly separable classification problem: (we can draw a line to seperate groups)

Vectors Review

Recall:
line in $\mathbb{R}^2$: ax + by = c
alternately written as

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \textbf{plane} in \mathbb{R}^3 : ax + by + cz = d \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4}$$

In general **n-dimensions**:
(suppose vectors are all column vectors)
$W^T = [w_1, w_2, w_3, \ldots, w_n]$ and $X =$

$$\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} \tag{5}$$

Then, the equation of a **plane in n-dimensions** is $W^T X = d$

dot product of $(WX) = ||W||||X||$ cos θ, where θ angle between W, X.
(also just sum of $x_1 w_1 + x_2 w_2 + \ldots x_n w_n$)

Therefore,

$$W^T X = \text{dot prod} = ||W||||X|| \cos\theta$$

Phrasing the problem

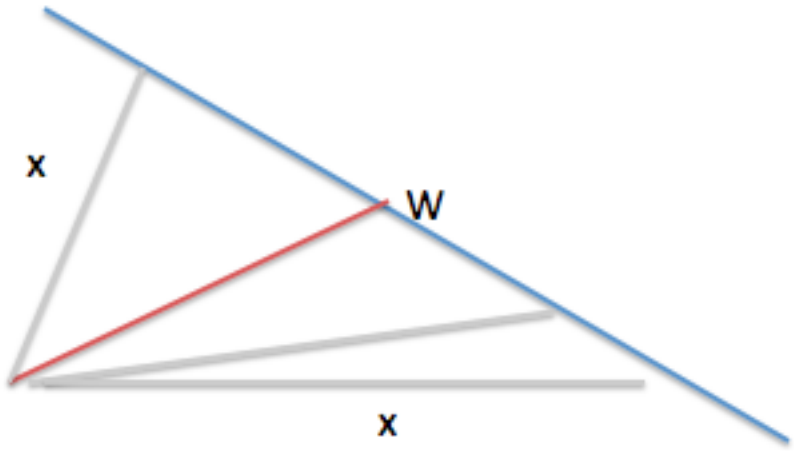
Given W and d; then the set of solutions x forms a plane in n-space.
(think about "projections" of W (by varying length and angle))

Let $g(x) = W^T x - d$, called "discriminant".

Our goal is to find values for W so that

$$g(x) > 0 \text{ if x belongs to 1st class}$$
$$g(x) < 0 \text{ otherwise.}$$

If we can find this, it defines a line (or plane) potentially seprating our dataset).



To the right side of blue line is one class to the left is another.
Suppose data is **linearly seperable** then, we can find W and d to solve.

Choose classes $l_1 = 1, l_2 = -1$.

Idea (with α learning rate):

if $w * x < 1$ and l = 1:
– update: w = w + x (α)
if $w * x > -1$ and l = -1:
– update: w = w - x * (α)

The pseudo-code for algo:

```
initialize W, d at some values

    for every element in training set:
        if not correclty classified:
```

```
            W += li Xi
            d += li
            (restart loop again)
            #called batch process since loop over entire set again

#idea: W tweaked down or up (depending on l = 1 or -1)
#turns out tweaking leads to convergence (if linearly seperable)
```

If data is separable and we find one boundary separating the data, we can always tilt the boundary a bit to find another boundary (actually infinitely many boundaries exist).

Why does tweaking always lead to convergence?

idea: as you tweak, you move away from initial weights, W, and the **cone of solutions** gets larger. Therefore, as we increment, we are 'guaranteed' to fall within cone.
to imagine cone of solutions: think of **rays** (corresponding to boundaries) going out into space. The area they cover gets bigger.

Theorem algo converges to a correct solution after finite number of weight corrections.

called " Perceptron Convergence Theorem"

click for proof

proof from class:

GOAL: Find weights w_0, w_1, w_2 such that the discriminant function

$$g(x_1, x_2) = w_0 + w_1 x_2 + w_2 x_2$$

divides the data set ($g(x) > 0$ if $l = 1, g(x) < 0$ if $l = 1 - 1$).
Define **normalized augmented** feature vector:
$\hat{x}' = l\hat{x} =$

$$\begin{bmatrix} l \\ lx \end{bmatrix} \tag{6}$$

Let $w[k]$ be the adjusted weights at the kth step.
Then,

$$w[1] = w[0] + n\hat{x}'(1)$$
$$w[2] = w[1] + n\hat{x}'(2)$$

meaning

$$w[k] - w[0] = n(x[1] + \cdots + x[k])$$

Therefore,

$$w^*(w[k] - w[0]) = nw^*(x(1) + \cdots + x(k))$$

where w^* is the hypothetical solution.
We want to show

$$AK^2 \leq ||w[k] - w[0]||^2 \leq BK$$

for some constants A, B. The thing sandwiched is the error think; area between parabola and line.
(means converges after a max of $k = B/A$ updates)

By Cauchy Schwarz we have the left inequality:

$$[w^*(w[k] - w[0])]^2 \geq ||w^*||^2||w[k] - w[0]||^2$$

Next, to bound above:
first, square the equations for $w[1], w[2], etc..$
next, sum up all the equations to get lots of cancellation:

$$||w[k] - w[0]||^2 \leq n^2(||x'(1)||^2 + ||x'(2)||^2 + \cdots + ||x'(k)||^2) \\ - 2nw(0^T(x'(2) + \cdots + x'(k))$$

Next let $M = \max\{$ all $||x'||^2\}$, then

$$\leq n^2M - n \text{ stuff })K$$

To simplify further we take $u =$ min of stuff:

$$\leq (n^2M - nu)K$$

et voila.
*note: u is always negative, since u ¿ 0 implies no data was misclassified!

Variation: augmented feature vector

make each feature vector x sit in n+1 dimension by adding 1 as a component. Then, the algebra cleans up and we get

$$||W||||X||$$

This makes updating quicker, since both values are part of W.

Another Variation: Augmented Normalized Feature Vector

agument the feature vector and normalize values.

To normalize: multiply by 1 if in class 1 and -1 if class 2 (so flips sign)

The **augmented normalized** feature vector is denoted: $\hat{x}'$
Idea: make an error for either class look the same.

LMS: Least Mean Squares

part of **linear regression**
Cost Function is the sum of squares (goal is to minimize this)
also called **objective function**

We have linear function

$$y = w_0 + w_1 x_1 + w_2 x_2$$

where output y, depends on two variables with weights w_1, w_2.

Goal is to use tweak w_i to minimize **cost function**.

Use Calculus to minimize.

Set gradient (partial derivatives with respect to each variable x_i).
Then, the optimal solution (after some algebra) is

$$optimal = (X^T X)^{-1} X^T Y$$

where Y are the set of outputs for the training data.

Often although the above is analytically convenient, it can consume quite a bit of memory. Other methods such as **steepest descent** (also called gradient descent) are used, because their implementation is **more pratical** in terms of memory and processing power required.

Steepest (Gradient) Descent

We use **steepest descent** (also called gradient descent) to do so. Gradient descent works by updating w_i using

$$\text{new } w_i = w_i - \alpha \frac{\partial}{\partial w_i} * \text{cost function}$$

where α is the learning rate.

Stop when gradient (partials) are zero.

multivariable chain rule (aside)

$$\frac{d}{dt} f(x(t), y(t)) = \frac{\partial}{\partial x}\frac{dx}{dt} + \frac{\partial}{\partial y}\frac{dy}{dt}$$

Two ways to implement LMS:
batch gradient descent entire dataset is used at every step, then updates
Sequential (Widrow-Hoff) update after each point in training data

Backpropagation Algorithm

Mathematically equivalent to LMS (similar to perceptron as well).

Goal: change weights of feature vectors to mimick a desired output.
At each step adjust the weight based on: actual - desired.
check out : tutorial
Use sign of

$$\sum_{j=1} w_{i,j}^{(l)} y_j$$

Hyperbolic tan and arctan are useful choice for Sigmoidal functions to solve optimization. (see slides for example with derivatives)

Kronecket Delta Function (get 1 if and only if the network nodes match; else 0)
Key math from snapp's slides:

- δ tells you error between
- update w based on δ
- Kronecket used inside sums to simply to one expression when network nodes match
- chain rule used to take derivatives

Generalization: when can we separate data

Linear Threshold Unit is the function $g(x) = sign(wx + w_0)$ separating the data into two groups.

The **capacity** of the linear threshold unit is the number of patterns we can separate by a line.

First a definition:
A distribution of m points in $\mathbb{R}^n$ is in **general position** if

- no subset of $n+1$ points lies on a line (plane, or hyperplane)

(if $m \leq n$, then we instead stipulate the set of m points is not contained in an $m-2$ dimensional hyperplane)

e.g., $n = 2, m = 2$, then we stipulate the two points can't coincide (0 dimensional hyperplane: same point).

e.g., $n = 2$ and any m, the points are in general positions if no two points are the same.

e.g., $m = 4, n = 2$, then no 3 points lie on same line.

Counting Linear Dichotomies

ways a set of points can be split up into groups.

For a **single point**, in any space $\mathbb{R}^n$, it's either group 1 or 2, so there are **two linearly** separable dichotomies.

In R^1, there are $2m$ linearly separable dichotomies, since points lie on one line. You can draw a line at m points, then switch classes.

To summarize:

- d(1, n) = 2
- d(m, 1) = 2m

$$\boxed{\text{In general: } d(m,n) = d(m-1,n) + d(m-1,n-1) \text{ (recursive)}}$$

Without recursion we have

$$d(m,n) = 2\sum_{i=0}^{n} \binom{m-1}{i}$$

e.g., $d(4,2) = 14$ (four points in 2 dimensions)

observation: in a sufficiently high dimension, **every linear dicotomy is possible**!

exam:

Baye's Rule, Theorem definitions: prior class, posterior, marginal density (p(x) = sum p(x, L) for each l), augmented feature vector, augmented normalized feature vector look over test 1: max/min, problem 1, integration gaussian

Pseudo-code: nearest neighbor, perception (backprop too hard)

what does naive baye's assume (each feature is independent), 1D regression derive solution,

Perceptron in Pseudo-code

```
Perceptron( {(x1, l1), ..., (xm, lm)}, step size = n )

W = random weight vector
W0 = random scalar

While error:
    error = false
    for (xi, li) in training data:
        if sign(w0 + w^T xi) != li:
            error = True
            #label doesn't match adjust
            W += n li Xi
            #learning rate * label * weight
```

```
            w0 += n*li

return w0, w
```

Counting Dichotomies continued

none

Support Vector Machines

Optimization problem for

$$L(W, w_0, \alpha) = 1/2 w^t w - \sum_{i=1}^{m} \alpha[l_i(w^t x_i + w_0) - 1]$$

where we minimize with respect to w and max with respect to α,
subject to constraint $\alpha_i \geq 0$ and

$$l_i(w^t x_i + w_0) - 1 \geq 0$$

where $w = \sum_{i=1}^{m} \alpha_i l_i x_i$.

To solve we rewrite as a **dual problem**.

Dual Problem

We instead want to maximize with respect to α

$$J(\alpha) = -1/2 \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j l_i l_j x_i^T x_j + \sum_{i=1}^{m} \alpha_i$$

subject to the constraint $\alpha_i \geq 0$ and $\sum_{i=1}^{m} \alpha_i l_i = 0$.

advantage: complexity of computation depends on the number of data points in training set, not the number of features observed.

Kernel is defined to be: $K(x, y) = (x^T y + 1)^2$.

Example

$$X_4 = \{[(-1, -1), -1], [(-1, 1), 1], [(1, -1), 1], [(1, 1), -1]\}$$

Note X_4 is not separable, so can't solve as usual (look up usual way).
Letting x, y be in $\mathbb{R}^2$, we find the kernel

$$\begin{aligned} K(x,y) &= (x^T y + 1)^2 = (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 1 \end{aligned}$$

Next we compute

$$J(\alpha) = -1/2[9\alpha_1^2 - 2\alpha_1\alpha_2 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2]$$

look at partial with respect to: $\alpha_1, \alpha_2, \alpha_3, \alpha_4$: set to zero to find minimum.
idea to look up is the **Kernel trick**
goal is to find $g(x)$ so that output is correct class label.
For this example the optimal $g(x) = -x_1 x_2$.