Ng 2

Linear regression
Gradient descent – skim

Viewer numbers ?

$\theta = (X^T X)^{-1} X^T y$

Has car driving itself – neural networks;
↓ & gradient descent
Someone at Carnegie Mellon, 1990

After 2 mins, it learns to imitate the reactions of the human driver; repeat for other road types;
12/sec; digitize image, & feed to NN;

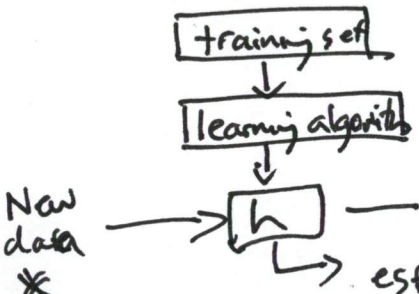
↓

regression problem: steering as continuous value

Notation

m = number of training examples # of training ex
x = "input" variables / "features"
y = "output" variable / "target" variable
(x, y) := 1 training example.
i^{th} training example: $(x^{(i)}, y^{(i)})$

Supervised learning



e.g. price...

How to represent h?

$h(x) = \theta_0 + \theta_1 x$

θ_i are parameters, real numbers

e.g. x_1 = size
x_2 = number of bedrooms

$h_\theta(x) = h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

For convenience, define $x_0 = 1$

$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$

n = # features

$\min_\theta \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$: to find parameters

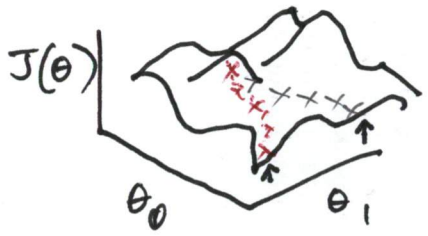
$\min_\theta \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 = J(\theta)$: cost function

$\min_\theta J(\theta)$

Ng 2

- start ε some θ (say $\theta = \vec{0}$: vector of 0's
- keep changing θ, to minimize J(θ)

Gradient descent



: stand at x;
- look around, small step to go down fastest

- where you end up can depend on where you start;

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

":=" setting a variable to be equal, in an algorithm

a = b : means identity assertion

For 1 training example

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2}(h_\theta(x) - y)^2$$

$$= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_i}(h_\theta(x) - y)$$: apply "chain rule"

$$= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_i}(\theta_0 x_0 + \dots \theta_n x_n - y)$$

$\theta_i x_i$ - the only one that matters

$$= (h_\theta(x) - y) \cdot x_i$$

Update:

$$\theta_i = \theta_i - \alpha (h_\theta(x) - y) \cdot x_i$$

α - alpha - the learning rate; size of the step;

For m training examples

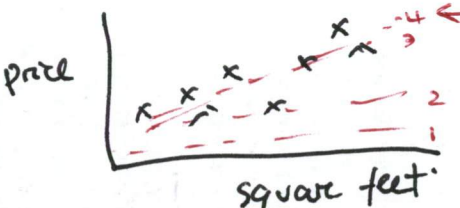
Repeat 'til convergence

$$\theta_i := \theta_i - \alpha \underbrace{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_i^{(i)}}_{\frac{\partial}{\partial \theta_i}(J(\theta_i))}$$

Batch gradient descent

For OLS, J(θ) is quadratic, & a nice bowl shape ε 1 global minimum

As you approach minima, when the gradient ↓ 0, so smaller steps



How to test for convergence?

(1) θ doesn't change?
(2) J(θ) " " ?

How is grad descent looking around?

- the derivative gives it to you ... that is the direction of steepest descent;

Batch: looks at whole training set;
- just to make 1 step!

Ng 2.

for large training sets: Stochastic Gradient Descent

Repeat until converge {
  For $j = 1$ to m {
    $\theta_i := \theta_i - \alpha (h_\theta(x^{(j)}) - y^{(j)}) \cdot x_i^{(j)}$
    (for all i
  }
}

- Look at 1st training example; adapting θ more quickly
- Much faster, but won't converge to global minimum; keep wandering a bit

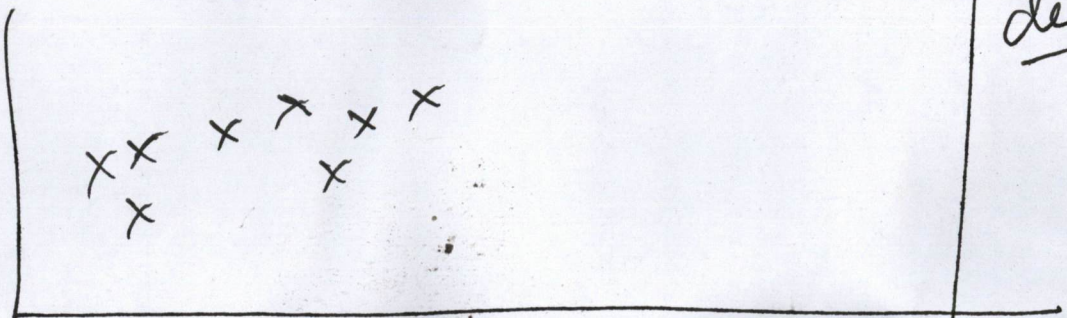
47:40

No

Lecture 2

Linear regression; gradient descent

Supervised learning



def: (1)

$$\nabla_\theta J = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \in \mathbb{R}^{n+1}$$

gradient

Notation

def (2)

$f : \mathbb{R}^{m \times n} \to \mathbb{R}$.

$f(A) : A = \mathbb{R}^{m \times n}$.

$f(A)$ ↑ matrix

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & & \\ \frac{\partial f}{\partial A_{m1}} & & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

$\nabla_A f(A)$ ↑ matrix

def (3)

If $A \in \mathbb{R}^{n \times n}$

$$\text{tr}\, A = \sum_{i=1} A_{ii}$$ - sum of diagonals

gradient descent

$$\theta := \theta - \alpha \nabla_\theta J.$$

θ : $\mathbb{R}^{n+1}$ dim vector

$\nabla_\theta J$: $\mathbb{R}^{n+1}$.

facts:

$\operatorname{tr} AB = \operatorname{tr} BA$

$\operatorname{tr} ABC = \operatorname{tr} CAB = \operatorname{tr}(BCA)$ 1.

$f(A) = \operatorname{tr}(AB)$

$\nabla_A \operatorname{tr} AB = B^T$ — transpose. 2

$\operatorname{tr} A = \operatorname{tr} A^T$ 3

If $a \in \mathbb{R}$, $\operatorname{tr} a = a$ 4

$\nabla_A \operatorname{tr} ABA^TC = CAB + C^TAB^T$ 5.

$$X\theta = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \theta$$

$$X\theta = \begin{bmatrix} x^{(1)T}\theta \\ \vdots \\ x^{(m)T}\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$1 \dots m$ = number of training examples

$0 \dots n$ = feature vector

$$X\theta - y = \begin{bmatrix} h(x^{(1)}) - y^{(1)} \\ \vdots \\ h(x^{(m)}) - y^{(m)} \end{bmatrix}$$

recall $z^Tz = \sum z_i^2$

$$(X\theta - y)^T(X\theta - y) = \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$$

$$= J(\theta)$$

Minimize J_θ

$\nabla_\theta J(\theta) \overset{\text{set}}{=} 0$

$\nabla_\theta \frac{1}{2}(X\theta - y)^T(X\theta - y) =$

$\frac{1}{2}\nabla_\theta (\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y)$

$J(\theta) \in \mathbb{R}$.

$= \frac{1}{2}[\nabla_\theta \operatorname{tr} \theta\theta^T X^T X - \nabla_\theta \operatorname{tr} y^T X\theta - \nabla_\theta y^T X\theta]$

(use facts) $= \frac{1}{2}[X^TX\theta(I) + X^TX\theta(I) + X^Ty + X^Ty]$

(A, B, $C = A$, B)

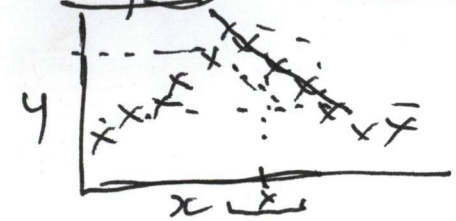
$= \frac{1}{2}[X^TX\theta + X^TX\theta - X^Ty - X^Ty]$

$= X^TX\theta - X^Ty = 0$

$\boxed{X^T X\theta = X^T y}$ → normal eqn

$\theta = (X^T X)^{-1} X^T y$ ⇐ closed form soln.

Ng 3 — Locally weighted linear regression (Loess)



parametric — fixed number of θ
non parametric — number of θ increases with m; need the whole training set

overfitting
underfitting

To evaluate h at x

LR: fit θ to min $\sum_i \left(y^{(i)} - \theta^T x^{(i)}\right)^2$

LWR: fit to just a subset of data

Fit θ to min $\sum_i w^{(i)} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$

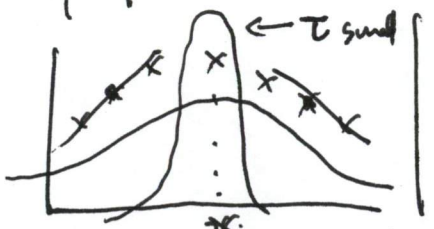
$w^{(i)}$: weights

where $w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$ — exponential decay

If $|x^{(i)} - x|$ is small, then $w^{(i)} \approx 1$.

If $|$ " " large, then $w^{(i)} \approx 0$.

τ: the bandwidth parameter.
↓
how fast weights fall off w distance



- for points far from x, $w^{(i)}$ will be close to 0;
- points close to x_i, $w^{(i)}$ will be close to 1.

- they ignore contrib^{n} of far away points
- diff communities choose diff weighting fcns

- everytime you ask, you fit entire dataset again, for the point you are querying;
- used to model dynamics of autonomous helicopter
- Andrew Moore: kd-trees to do loess for large datasets

Probabilistic interpretn of LR

(1)

Why $\sum$ error^2? for OLS?

One set of assumptions

price of house

Assume (A) $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$

$\varepsilon^{(i)}$ = error : captures other features or noise

$\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ — gaussian error
- convenient
- CLT: sum of random vari

$\varepsilon^{(i)}$ is distributed; $\mathcal{N}$: normal / Gaussian

$$p(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-\varepsilon^{(i)2}}{-2\sigma^2}\right)$$

$$p(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$\therefore$ $y^{(i)} | x^{(i)}, \theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$

$y^{(i)}$: price of house. $\theta^T x^{(i)}$: mean

If θ is not a random value (as in freq);

$p(y^{(i)} | x^{(i)}; \theta)$ ← "parameterized" by θ

Assume (B) : $\varepsilon^{(i)}$'s are IID ← independent. identically distributed: - they all come from same $\mathcal{N}$

$$L(\theta) = p(\vec{y} | X; \theta) \quad \leftarrow \text{Likelihood}$$
$$= \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}; \theta)$$
$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

"Likelihood of θ"; "probability of data"

(Ng 3)

Maximum likelihood:

Choose θ to maximise $L(\theta) = P(\vec{y} \mid X; \theta)$; as probable as possible to see data

$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= \log \prod_{i} \frac{1}{\sqrt{2\pi}\sigma} \exp(\dots) \\ &= \sum_{i=1}^{m} \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp(\dots) \right] \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} \left[-\frac{(y^{(i)} - \theta^T (x)^2}{2\sigma^2} \right] \end{aligned}$$

So max $l(\theta)$ is the same as minimising

$$\sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x)^2}{2} = J(\theta)$$

$\therefore$ OLS is ML, assuming iid errors on data

n.b σ^2 doesn't matter :- you get same values of θ;

Endows LR with probabilistic properties

Classification : $y \in \{0, 1\}$

e.g. medical problem : do they have the disease
e.g. will the house sell;
e.g. is the email spam?
e.g. will the computer crash

apply LR to classif=? O.K., but not a good idea

Add points, & the line changes

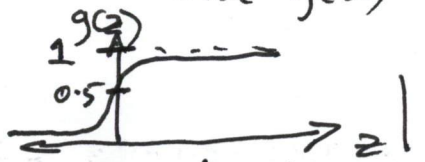
$y \in \{0, 1\}$

$h_\theta(x) \in [0, 1]$

Choose:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

where $g(z) = \frac{1}{1 + e^{-z}}$ – sigmoid or logistic function



$P(y = 1 \mid X; \theta) = h_\theta(x)$: estimate probability that $y = 1$.

$P(y = 0 \mid X; \theta) = 1 - h_\theta(x)$

$P(y \mid X; \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$: compact way of writing

(if $y = 0$, one half disappears;)

$$\begin{aligned} L(\theta) = P(\vec{y} \mid X; \theta) &= \prod_i P(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_i h(x^{(i)})^{y^{(i)}} (1 - h(x^{(i)}))^{1 - y^{(i)}} \end{aligned}$$

Find θ that max $L(\theta)$;

max the $l(\theta)$ =

$$\sum_{i=1}^{m} y^{(i)} \log (h_\theta(x^{(i)}))^{y^{(i)}} + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))$$

Apply gradient descent argument → gradient ascent

$\theta := \theta + \alpha \nabla_\theta l(\theta)$: max the quadratic

Compute partial deriv for each w.r.t θ_j: ↙ lots of algebra

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}) x_j^{(i)}$$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}) - x_j^{(i)})$$

∴ (batch gradient descent)

Ng 3

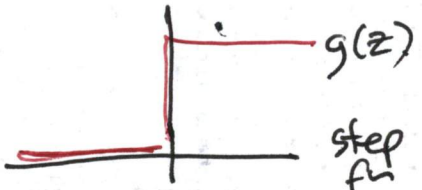
Is it the same algorithm? No. In Logistic regression $h_\theta(x^i)$ is not the linear, but logistic function;

Same learning rule/algorithm

To learn: read it; cover-up the deriv^{n}; do it yourself.

Perceptron algorithm

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



$$h_\theta(x) = g(\theta^T x)$$

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) \cdot x_j^{(i)}$$ — learning rule

- difficult to endow $\bar{c}$ probabilistic semantics
- simpler than logistic regression

Ng 4

1. Logreg → newtons method
2. expfam family
3. generalized linear models

Recall:

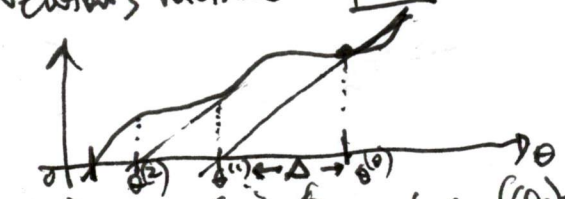
$P(y=1|x;\theta) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$\ell(\theta) = \sum_i y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))$

$\theta_j := \theta_j + \alpha (y^{(i)} - h(x^{(i)})) x_j^{(i)}$ ← stoch grad descent

To fit logreg, batch/stoch gradient descent is ok.

Newton's method is faster



$f(\theta)$: find θ so that $f(\theta)=0$

$f'(\theta^{(0)}) = \frac{f(\theta^{(0)})}{\Delta} \Big| \Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$

$\theta^{(1)} = \theta^{(0)} - \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$

$\theta^{(t+1)} = \theta^{(t)} - \frac{f(\theta^{(t)})}{f'(\theta^{(t)})}$

$\ell(\theta)$: want θ such that $\ell'(\theta)=0$

$\theta^{(t+1)} = \theta^{(t)} - \frac{\ell'(\theta^{(t)})}{\ell''(\theta^{(t)})}$ | to find max likelihood of parameters for log. regression

All of these algorithms converge, but the speeds are different;

Newton M enjoys fast convergence; (quadratic convergence)

N.M for θ as a vector

(4)

$\theta^{(t+1)} = \theta^{(t)} - H^{-1} \nabla_\theta \ell$

where H = Hessian matrix : — of 2nd derivative

where $H_{ij} = \frac{\partial^2 \ell}{\partial \theta_i \partial \theta_j}$

On every iter, you have to invert H

Generalized linear models

$P(y|x;\theta)$

① $y \in \mathbb{R}$: Gaussian → OLS

② $y \in \{0,1\}$: Bernoulli → log regression

For logreg, where did $\frac{1}{1+e^{-\theta^T x}}$

Both are special cases of more general case

Bernoulli (ϕ) $P(y=1;\phi) = \phi$

$\mathcal{N}(\mu, \sigma^2)$

→ exponential family

$P(y;\eta) = b(y) \exp(\eta^T T(y) - a(\eta))$ η: 'eta'

η = "natural parameter"

T = sufficient statistic; usually $T(y)=y$

For a, b, T, as we vary η get different dists.

e.g. for Bernoulli

$P(y;\phi) = \phi^y (1-\phi)^{1-y}$

$= \exp(\log \phi^y (1-\phi)^{1-y})$

$= \exp(y \log \phi + (1-y) \log(1-\phi))$

$= \exp(\log \frac{\phi}{1-\phi} \cdot y + \log(1-\phi))$

η : $\log \frac{\phi}{1-\phi}$; $T(y)$: y ; $-a(\eta)$: $\log(1-\phi)$

Ng 4 (cont)

$\therefore \eta = \log \frac{\phi}{1-\phi} \Rightarrow \phi = \frac{1}{1+e^{-\eta}} \approx$ logistic function!

$a(\eta) = -\log(1-\phi) = \log(1+e^{\eta})$
$T(y) = y$
$b(y) = 1$
} → pdf for Bernoulli

for Gaussian

$N(\mu, \sigma^2)$: set $\sigma^2 = 1$; (σ^2 is a scaling factor)

$\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(y-\mu)^2) = \dots$

$= \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}y^2) \exp(\mu y - \frac{1}{2}\mu^2)$

$b(y)$ $\quad \mu = \eta$ $\quad T(y) = y$

$a(\eta) = \frac{1}{2}\mu^2 = \frac{1}{2}\eta^2$

Most textbook dists can be written in terms of exp. family
e.g. multivariate N; multinomial; Poisson ... counts
Gamma (≥ 0); exponential dist (≥ 0); beta; dirichilet = dist over fractions

---

Having chosen an exp fam dist, how do you derive a GLM?

Assume: / design
(1) $y|x;\theta \sim$ ExpFamily(η)
(2) Given x, our goal is to output $E[T(y)|x]$
want $h(x) = E[T(y)|x]$

(3) $\eta = \theta^T x$: allows us to use linear regression;
$[\eta_i] = \theta_i^T x$

Example

(i) Bernoulli

$y|x;\theta \sim$ ExpFamily(η)

for fixed x, θ, learning algorithm outputs

$h_\theta(x) = E[y|x;\theta] = p(y=1|x;\theta)$
$= \phi$
$= \frac{1}{1+e^{-\eta}} =$ | log regression algorithm
$= \frac{1}{1+e^{-\theta^T x}}$

core of GLM

Only decision I need to make is choose Bernoulli dist^n; knowing y is 1 or 0)

n.b. $g(\eta) = E[y;\eta] = \frac{1}{1+e^{-\eta}}$: "canonical response function" "canonical link function"
g^{-1} ←

(ii) {Skips Gaussian example: so simple it is confusing}

(iii) example : Multinomial

$y \in \{1, k\}$

Parameters: $\phi_1, \phi_2 \dots \phi_k$
$P(y=i) = \phi_i$
$\phi_k = 1 - (\phi_1 + \dots + \phi_{k-1})$
$\therefore \phi : \phi_1, \phi_2 \dots \phi_{k-1}$

Ng 4 (cont) Multinomial cont

$y = \{1, 2 \dots k\}$

$T(1) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ $T(2) = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$: $\in \mathbb{R}^{k-1}$

$T(k-1) = \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}$, $T(k) = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ $k-1$ dimensional vectors

$1\{\text{True}\} = 1$; $1\{\text{False}\} = 0$: "indicates notation" indicates T/F statement

$T(y)_i = 1\{y = i\}$: ith element of T is y

$$P(y) = \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1\{y=k\}}$$
$$= \phi_1^{T(y)_1} \phi_2^{T(y)_2} \dots \phi_{k-1}^{T(y)_{k-1}}$$
$$= b(y) \exp(\eta^T T(y) - a(\eta))$$

where $\eta = \begin{bmatrix} \log(\phi_1/\phi_k) \\ \vdots \\ \log(\phi_{k-1}/\phi_k) \end{bmatrix} \in \mathbb{R}^{k-1}$

$a(\eta) = -\log(\phi_k)$; $b(y) = 1$.

} exp fam

$$\phi_i = \frac{e^{\eta_i}}{1 + \sum_{j=1}^{k-1} e^{\eta_j}} \quad (j = 1 \dots k-1)$$

$$= \frac{e^{\theta_i^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}} \quad [\eta_i = \theta_i^T x]$$

So l. algorithm:

$$h_\theta(x) = E[T(y) | x; \theta]$$
$$= E\begin{bmatrix} 1\{y=1\} \\ \vdots \\ 1\{y=k-1\} \end{bmatrix} \Big| x; \theta \Big] = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix}$$
$$= \begin{bmatrix} e^{\theta_1^T x} / (1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}) \\ \vdots \\ e^{\theta_{k-1}^T x} / (1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}) \end{bmatrix}$$

} "softmax regression"

TO do ML

So $y \in \{1 \dots k\}$

choose multinomial;

train training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$

then $L(\theta) = \prod_{i=1}^{m} P(y^{(i)} | x^{(i)}; \theta)$

$$= \prod_{i=1}^{m} \phi_1^{1\{y^{(i)}=1\}} \dots \phi_k^{1\{y^{(i)}=k\}}$$

where $\phi_1 = \frac{e^{\theta_1^T x}}{1 + \sum_j e^{\theta_j^T x}}$ ...

Ng 5 — Classifiers — 51k views

GDA
Naive Bayes

Discriminative

$P(y|x)$ = learns

or learn

$h_\theta(x) \in \{0, 1\}$

e.g. logreg: output 0, 1 directly;

Generative

$P(x|y)$

↗

builds a probabilistic model of features conditioned on class labels y.

$$P(y=1|x) = \frac{P(x|y=1) \cdot P(x)}{P(x)}$$

where $P(x) = P(y=0|x)P(x) + P(y=1|x)P(x)$

Example

Assume $x \in \mathbb{R}^n$, continuous

Gaussian discriminant analysis algorithm

Assume $P(x|y)$ is Gaussian

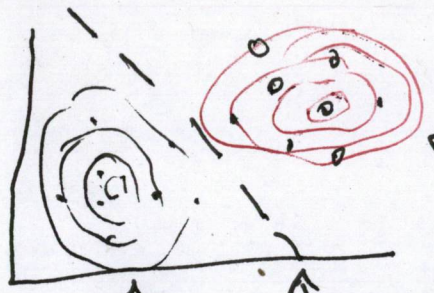
$z \sim N(\vec{\mu}, \Sigma)$ ... Σ = covariance matrix, μ = mean

$\Sigma = E[(x-\mu)(x-\mu)^T]$

$$p(z) = \frac{1}{\sqrt{2\pi}^n |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$ — Multivariate normal

$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & -.5 \\ -.5 & 1 \end{bmatrix}$

Useful look at Gaussians on the move.



GDA 2. benign vs malignant cancers

$$P(y) = \phi^y (1-\phi)^{1-y}$$ — Bernoulli ← true value of y given data

$$P(x|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T \Sigma (x-\mu_0)\right)$$

$$P(x|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T \Sigma (x-\mu_1)\right)$$

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$$ ← joint likelihood

$$= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}) \cdot p(y^{(i)})$$

cf logreg

$$\ell(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$$ ← conditional likelihood

To fit, do **MLE**

$\max \ell(\phi, \mu_0, \mu_1, \Sigma)$

$$\phi_{MLE} = \frac{\sum_i y^{(i)}}{m} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}$$: fraction with label '1'

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$: examples w/ label '0'

Ng 5

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$: examples ∈ label '1'

"stare at it until it makes sense"

$\Sigma = \cdots$

Predict:

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

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

If $P(y)$ is uniform:

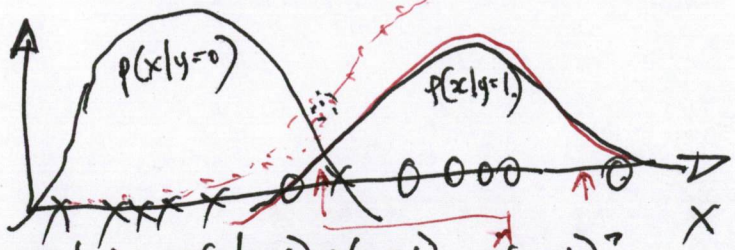
$$\arg\max_y P(x|y)$$ ← simpler case, not common.

e.g. $\arg\min_x (x-5)^2 = 5$; ie. value of x that minimizes the function

GDA has rel^n to logreg

26.04

e.g.



$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)} \quad P(y=1)?$$

ie. under GDA; when you do $P(x|y)$, you get logreg; but with a different steepness of the sigmoid

(2)

Ad/disadv of GDA

$x|y \sim$ Gaussian

⇓

logistic posterior for $P(y=1|x)$] implic^n in the opposite direct^n doesn't hold;

e.g. $x|y=1 \sim$ Poisson(λ_1), $x|y=0 \sim$ Poisson(λ_0)] e.g. ExpFam(η_1), ExpFam(η_0). Gaussian assumption is stronger;

then ⇒ $P(y=1|x)$ is logistic.

- shows robustness of logistic regression to modelling assumptions

Tradeoffs b/w GDA & logreg

- when $x|y$ is Gaussian, then GDA will do better; if makes better use of the data
- But if you don't know shape of $x|y$, then logreg will do better.
- real advantage of generative algorithm: require less data;
  - GDA needs less training data
  - Logreg makes less assumptions, but needs more training data
- philosophical discussion of ϕ vs ϕ_{MLE} (41.39)

∴ Above discussion shows another derivation of logregression

Ng 5 — Naive Bayes

$y \in \{0, 1\}$ e.g. email spam

0 = non-spam, 1 = spam

$x = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$ a aardvark … buy … cs229 … zymurgy — fermentation → feature vector

Lets build a generative model for this

$P(x|y)$ $P(y)$

$x \in \{0,1\}^n$: binary value

$n = 50{,}000$

$\therefore 2^{50{,}000}$ bit vectors $\triangleq 2^{50{,}000}$ parameters for multinomial distribution

Assume:

x_i's are conditionally independent given y

$P(x_1, x_2 \dots x_{50k}|y) = P(x_1|y) \cdot P(x_2|y) \dots$ use chain rule

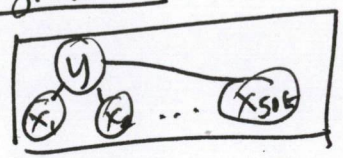
assume $= P(x_1|y) \cdot P(x_2|y) \dots P(x_{50k}|y)$

$= \prod_{i=1}^{n} P(x_i|y)$ } this is the model of $P(x|y)$

This is a false assumption, because words are related.

But Naive Bayes works well for text docs.

Digression



→ Bayesian network version

Parameters

$\phi_{i|y=1} = P(x_i = 1 | y = 1)$

$\phi_{i|y=0} = P(x_i = 1 | y = 0)$

$\phi_y = P(y = 1)$

Joint likelihood:

$\mathcal{L}(\phi_y, \phi_{i|y=1}, \phi_{i|y=0}) = \prod_{i=1}^{n} P(x^{(i)}, y^{(i)})$

Given a training set, write joint likelihood of the parameters & do MLE

$\phi_{j|y=1} = \frac{\sum_{i=1}^{n} 1\{x_j^{(i)} = 1, y^{(i)} = 1\}}{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}}$

Numerator: loop: run thru training set, count all times you saw word j, for $y = 1$

Denominator: number of spam emails

→ gives ratio of spam emails that have the word j in them

MLE: $\phi_y = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}}{n}$

Having estimated all of these parameters, when given a new piece of email, you can classify $P(y|x)$ for it using Bayes rule $\propto P(x|y)P(y)$

Ng 5

So it turns out that this idea almost works

e.g. 'NIPS' contains

$P(x_{30000}=1 \mid y=1) = 0$: } you've never seen it before;
$P(x_{30000}=1 \mid y=0) = 0$ }

So span classifier

$$P(y=1 \mid x) = \frac{P(x \mid y=1)(P(y=1)}{P(x \mid y=1) P(y=1) + P(x \mid y=0)(P(y=0))}$$

$$\text{so } \prod_{i=1}^{50,000} P(x_i \mid y=1)$$

but $P(x_{30k}=1 \mid y=1) = 0$

So $= \frac{0}{0+0}$ ⇐ undefined

Statistically bad idea to say:

$P(x_{30k} \mid y=1) = 0$

Just ∵ you haven't seen it before, ...

Fix:

Basketball game stats

		won
2/8	W.S.	0
2/11	Wash	0
2/22	USC	0
⋮	UCLA	?
3/15	Louisville	

MLE for Bernoulli random variable

$P(y=1) = \frac{\#1\text{'s}}{\#0\text{'s} + \#1\text{'s}}$

Add 1 to all counts

$P(\text{win}) = \frac{0+1}{5+1+0+1}$] Laplace ~~smoothing~~ smoothing

"will the sun rise tomorrow?"

More generally:

if $y \in \{1 \dots k\}$, $P(y=1) = \sum_{j=1}^{m} \frac{1\{y_i^{(i)}=j\}+1}{m+1}$

(multinomial MLE)

So for Naive Bayes, that gives us:

$$\phi_{i|y=1} = \frac{\sum_{i=1}^{m} 1\{x^{(i)}=1, y^{(i)}=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}+2}$$

Ng 6

- Naive Bayes
  - Event models
- Neural networks
- Support vector machines

Naive Bayes algorithm

$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$ ← Bernoulli:

$x_i \in \{0,1\}$ $\begin{bmatrix} 0 \\ i \\ a \end{bmatrix}$ ← the i^{th} word

n = # words in dict

$P(y)$

$\underset{y}{\text{argmax}}\,(P(y|x)) = \underset{y}{\text{argmax}}\, P(x|y)\,P(y)$ ← Bayes Rule

2 variations on Naive Bayes

(1) $x_i \in \{1,\dots,k\}$

$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$

↑ multinomial

e.g.: if you discretize a feature into k-values

for houses: living area

Living area	< 500	500–1000	1000–1500	1500–2000	...
	1	2	3		

Commonly use 10 buckets

(2) For text documents or sequences

Event model takes into a/c number of occurrences

↑ "multinomial event model"

Given an email

$(x_1^{(i)}, x_2^{(i)}, \dots x_{n_i}^{(i)})$ where n_i = # of words in email

$x_j \in \{1, 2, \dots$ ↖ index to dictionary

* So n_i varies for every training example & x_j is no longer binary

Generative model ← joint likelihood:

$P(x,y) = \left(\prod_{i=1}^{n} P(x_i|y)\right) P(y)$

way to think about this formula, is to imagine that there is some random distrib^n that generates emails, & that

(i) 1st y label is chosen — spam / not spam

(ii) having decided, someone iterates over all 3rd positions of the email/words, & generates words depending on the label a/c to some distrib^n.

Parameters of this model

$\phi_{k|y=1} = P(x_j = k | y=1)$: if it's spam, what's the probability that the next word is 'k'

$\phi_{k|y=0} = P(x_j = k | y=0)$

$\phi_y = P(y=1)$

Ng 6 — 52K views

Given a training set, we can work out the MLE of the parameters

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\} \sum_{j=1}^{n_i} 1\{x_j^{(i)}=k\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\} \cdot n_i}$$

(+1 — Laplace smoothing; +|V| — number of words in dictionary)

Numerator says:
- sum over all your emails, if take spam, & count of all the times you observed word k

Denominator
- sum over training set of spam, the length
- total length of all spam emails in training set

Ratio: what is the fraction of all words in spam that were word 'k'

Q: "What is the def^n of MLE?"

- write down the likelihood of parameters

$$\ell(\phi_{k|y=1}, \phi_{k|y=0}, \phi_y) = \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi_{k|y=1}, \phi_{k|y=0}, \phi_y)$$

$$= \log \prod_{i=1}^{m} \prod_{j=1}^{n_i} p(x_j^{(i)} | y^{(i)}; \phi_{k|y=1}, \phi_{k|y=0}, \phi_y) \cdot p(y^{(i)}; \phi_y)$$

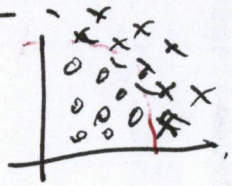
- you maximise this in terms of parameters

for text class^n, the multinomial event model does better. The event model takes into a/c the number of times it appears, but not sure why... The bigram & trigram don't do better on class^n

(2)

25.13 Non-linear classifiers

logreg $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$



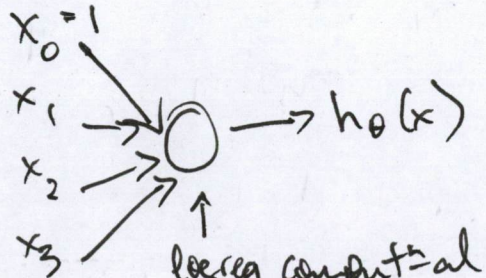
Is there a non-linear boundary?

$\longleftrightarrow$ 'eta'

$x|y \sim \text{ExpFam}(\eta)$

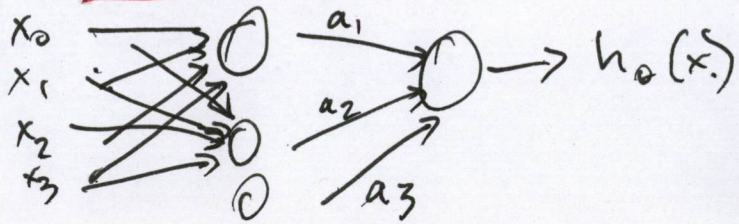
$x|y=0 \sim \text{ExpFam}(\eta_0) \Leftrightarrow$ logreg.

Under NB, actually using a linear classifier.



To get non-linear decision boundaries, :

(A) Neural network



Ng 6

Each unit has its own Param

$a_1 = g(x^T\theta^{(1)})$ where $g(z) = \frac{1}{1+e^{-z}}$

$a_2 = g(x^T\theta^{(2)})$

. . . .

$h_\theta(x) = g(\vec{a}^T\theta^{(4)})$ where $\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))^2$

↑ cost fn. ↑ quadratic cost function

- use grad descent to make predn's fit observn's
- a.k.a "back propagation"

You can look at what hidden layer is doing

Shows video for ~~digital~~ digit recognition from prof at NY. "LeNet 5"

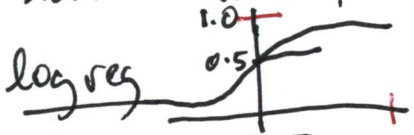
34:33 For neural network, lots of local optima;
Vast researchers support SVM as better
intuitive off-the-shelf;
Neural networks were the best for many years

video Robustness to noise

Video: "The machine that changed the world" BBC / PBS
"Nettalk" neural network to read text.
Terry ~~Zyrowski~~ Sejnowski: generated excitement; uses child voice "getting candy at grandmothers house"

(B) Support vector Machines (SVM)

Another algo to get non-linear classifiers
- start ~ linear; linearly separable

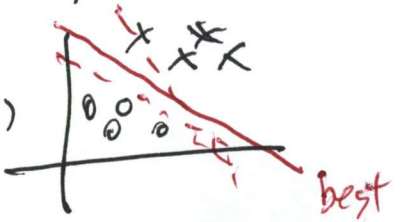


- computes $\theta^T x$
- predicts "1" iff $\theta^T x \geq 0$
  - "0" iff $\theta^T x \leq 0$

If $\theta^T x >> 0$, then very confident that $y=1$
If $\theta^T x << 0$, " " " " that $y=0$

1st intuition
Nice if $\forall i$, such that $y^{(i)}=1$, $\theta^T x >> 0$ | "functional margin"
$\forall i$ " " $y^{(i)}=0$, $\theta^T x << 0$

2nd intuition
Assuming linear separable, best is most distant
"geometric margin"



Ng 6

Change notation

$y \in \{-1, +1\}$

have h output values in $\{-1, +1\}$

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if otherwise} \end{cases}$$

replace $h_\theta(x) = g(\theta^T x)$ $\quad x \in \mathbb{R}^{n+1}$, $x_0 = 1$

with $h_{w,b}(x) = g(w^T x + b)$ i.e. separate out b as intercept

$x \in \mathbb{R}^n$

$w \in \mathbb{R}^{n+1}$

Defn: functional margin of hyperplane (w, b) w.r.t. $(x^{(i)}, y^{(i)})$ is

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

($\hat{\gamma}$: gamma)

If $y^{(i)} = 1$, want $w^T x^{(i)} + b >> 0$

If $y^{(i)} = -1$, want $w^T x^{(i)} + b << 0$

If $y^{(i)}(w^T x^{(i)} + b) > 0$, then correctly classified $(x^{(i)}, y^{(i)})$

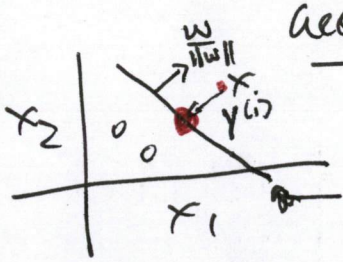
$\hat{\gamma} = \min \hat{\gamma}^{(i)}$: worst case of all margins on training set;

- we want it to be large;

To make functional margin large, could just say $w \to 2w$; $b \to 2b$.

∴ Need a normalizn condition? $||w|| = 1$?

geometric margin



"normal vector" $= \frac{w}{||w||}$; unit length

$x^{(i)} - \gamma^{(i)} \cdot \frac{w}{||w||}$ = the heavy red point on the hyperplane.

(i) $w^T\left(x^{(i)} - \gamma^{(i)} \frac{w}{||w||}\right) + b = 0$; i.e. satisfies the equation for the hyperplane.

Solve (i) for $\gamma^{(i)}$:

$$w^T x^{(i)} + b = \gamma^{(i)} \frac{w^T w}{||w||} = \gamma^{(i)} ||w||$$

$$\therefore \gamma^{(i)} = \left(\frac{w^T}{||w||}\right) x^{(i)} + \frac{b}{||w||}$$

If you have training example $x^{(i)}$, then can compute the distance;

More generally, define geometric margin

$$\gamma^{(i)} = y^{(i)}\left[\frac{w^T}{||w||} x^{(i)} + \frac{b}{||w||}\right]$$

↑ note diff notation

Ng 6

facts

1. If $\|w\| = 1$,

$$\hat{\gamma}^{(i)} = \gamma^{(i)}$$

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|w\|}$$

2. Geometric margin

$$\gamma = \min_i \gamma^{(i)}$$

Maximum margin classifier: (precursor to SVM)

algorithm that maxs geometric margin:

$$\max_{\gamma, w, b} \gamma$$

such that $y^{(i)} (w^T x^{(i)} + b) \geq \gamma$

where $\|w\| = 1$

You can scale w,b by any factor, & it doesn't change geometric margin

Runs as well as logreg;

will develop into infinite dimensional space to get SVM;

Ng 7 (41k views) SVM (cont)

Today
- optimal margin classifier
- primal/dual optimiz^n ; problem (KKT) ⇓ "convex optimiz^n"
- SVM dual
- kernels

Recap

$h_{w,b}(x) = g(w^T x + b)$

where $g(z) = \begin{cases} +1, & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$

- func^n margin $\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$: very large is good
- geometric margin $\gamma^{(i)} = y^{(i)}\left(\frac{w^T}{\|w\|} x^{(i)} + \frac{b}{\|w\|}\right)$: distance interpretation

$\gamma = \min_i \gamma^{(i)}$
$\hat{\gamma} = \min_i \hat{\gamma}^{(i)}$
} take worst case

- scaling w or b doesn't change hyperplane;

e.g. $\|w\| = 1$ doesn't change the problem
or $|w_1| = 1$
$w_1^2 + |w_1| = 17$

Optimiz^n problem :

choose w, b to max geometric margin γ

#1 find $\max_{\gamma, w, b} \gamma$ s.t $y^{(i)}(w^T x^{(i)} + b) \geq \gamma$, $\|w\| = 1$, $i = 1 \dots m$

But this is nasty : $\|w\| = 1$ means it lies on a sphere; this is a non-convex constraint ⇒ local minima, etc

#2 : $\max_{\gamma, w, b} \frac{\hat{\gamma}}{\|w\|}$ s.t $y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}$

- different way of posing the same problem based on "fact" from last [6]
- func^n margin of every example has to $\geq \hat{\gamma}$
- still maxing geom margin

Suppose I have software for convex optimiz^n, it will choose γ to be big as possible, limited by constraints (ie. the smallest functional margin).

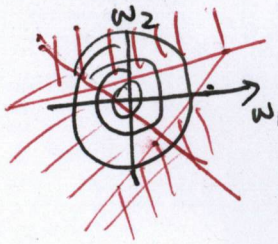
$\hat{\gamma} = 1$
i.e $\min_i y^{(i)}(w^T x^{(i)} + b) = 1$
} new scaling constraints;

Ng 7

Add this to #2

#3 $\max_{w,b} \|w\|^2$ ← $\max \frac{1}{\|w\|}$

s.t: $y^{(i)}(w^T x^{(i)} + b) \geq 1$ } optimal & margin classifier



linear constraints that limit the space;

i.e. a convex problem that has no local optima; readily solvable

Take this, & throw at QP solver

Primal & dual optimizn / Convex optimization

Lagrange multipliers

$\min_w f(w)$ s.t $h_i(w) = 0, i = 1 \dots l$ $\quad h(w) = \begin{bmatrix} h_1(w) \\ \vdots \\ h_l(w) \end{bmatrix} = \vec{0}$

Langrangian:

$\mathcal{L}(w, \beta) = f(w) + \sum_i \beta_i h_i(w)$ $\quad \beta =$ Langrange multipliers

$\frac{\partial \mathcal{L}}{\partial w} \overset{\text{set}}{=} 0$; $\frac{\partial \mathcal{L}}{\partial \beta} = 0$

for w^* to be a sol^n, necessary that

$\exists \beta^*$ s.t. $\frac{\partial \mathcal{L}(w^*, \beta^*)}{\partial w} = 0$; $\frac{\partial \mathcal{L}(w^*, \beta^*)}{\partial \beta} = 0$ (2)

Write down a general^n to solve a more difficult

Constraint:

$\min f(w)$

sub to $g_i(w) \leq 0$; $i = 1 \dots k$ ("$g(w) = \vec{0}$")

$h_i(w) = 0$ $i = 1 \dots l$ ("$h(w) = \vec{0}$")

Langrangian:

$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i}^{l} \beta_i h_i(w)$

Define: $\theta_p(w) = \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$ ← primal problem

Consider:

$p^* = \min_w \max_{\alpha, \beta, \alpha \geq 0} \mathcal{L}(w, \alpha, \beta) = \min_w \theta_p(w)$

$\theta_p(w)$: If $g_i(w) > 0$ then $\theta_p(w) = \infty$ ($\because \max_\alpha$ will go to ∞)

If $h_i(w) \neq 0$, then $\theta_p(w) = \infty$ (" " ")

Otherwise, $\theta_p(w) = f(w)$: i.e. if satisfy constraint, Lagrange multipliers drop out

Ng 7

Thus $\theta_p(w) = \begin{cases} f(w) & \text{if constraints} \\ \infty & \text{otherwise} \end{cases}$

So $\min \theta_p(w)$ = original primal problem

So here's a cool thought: Dual problem

$$\theta_D(\alpha,\beta) = \min_w \mathcal{L}(w,\alpha,\beta)$$

$$d^* = \max_{\alpha\geq 0,\beta} \min_w \mathcal{L}(w,\alpha,\beta) = \max_{\alpha\geq 0,\beta} \theta_D(\alpha,\beta)$$

switched order from p^* — dual problem

Fact

$d^* \leq p^*$: i.e. $\max \min(\ldots) \leq \min \max(\ldots)$

$$\max_{y\in\{0,1\}} \left(\min_{x\in\{0,1\}} 1\{x=y\}\right) \leq \min_{x\in\{0,1\}} \left(\max_{y\in\{0,1\}} 1\{x=y\}\right)$$

inequality holds for any f, i

0 ≤ 1.

Sometimes $d^* = p^*$

& dual problem is often easier to solve;

Write down primal problem, write down dual opt prob

then derive SVM algorithm

Conditions for $d^* = p^*$ (3)

Let f be convex (Hessian $H \geq 0$)

Suppose h_i is affine $[h_i(w) = a_i^T w + b]$ i.e. linear

and suppose g_i are strictly feasible

$[\exists w$, s.t. $\forall i$ $g_i(w) < 0]$

Then $\exists w^* \alpha^* \beta^*$ s.t. w^* solves the primal problem & α^*, β^* solve dual problem, and $p^* = d^* = \mathcal{L}(w^*,\alpha^*,\beta^*)$

And $\frac{\partial}{\partial w}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0$

$\frac{\partial}{\partial \beta}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0$

$\alpha_i^* g_i(w) = 0$ ← "KKT complementarity condition"

$g_i(w^*) \leq 0$, $\alpha_i^* \geq 0$

↑ either α_i or $g_i = 0$!

Karush-Kuhn-Tucker

If $\alpha_i > 0$, $\Rightarrow g_i(w^*) = 0$

Usually, $\alpha_i^* \neq 0 \iff$ (iff) $g_i(w^*) = 0$

$g_i(w)$ is an "active constraint"

Ng 7

Apply to optimal margin classifier to get SVM

SVM

Lagrange multiplier: α_i β_i → α_i

parameters : w → w, b

n.b Notation change

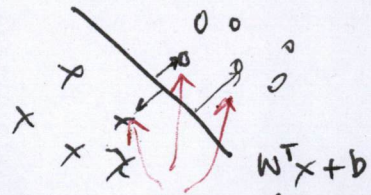
$$\min \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \quad i = 1 \dots m$$

$$g_i(w,b) = -y^{(i)}\left(w^T x^{(i)} + b\right) + 1 \leq 0$$

if $\alpha_i > 0$, $\Rightarrow g_i(w,b) = 0$

$\Leftrightarrow (x^{(i)}, y^{(i)})$ has functional margin $= 1$



examples = functional margin = 1 ; & usually $\alpha_i \neq 0$

There examples are the support vectors

There are few of them ;

$\alpha_i = 0$ for non-support vectors

#1 $$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y^{(i)}(w^T x^{(i)} + b) - 1\right)$$

Dual problem:

$$\theta_D(\alpha) = \min_{w,b} \mathcal{L}(w,b,\alpha)$$

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \overset{\text{set}}{=} 0$$

$$\Rightarrow \boxed{w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}} \Leftarrow \text{linear combination}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{m} y^{(i)} \alpha_i \overset{\text{set}}{=} 0$$

Take these 2 constraints, figure out w & plug back into #1

$$\mathcal{L} = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left(y^{(i)}\left(w^T x^{(i)} + b\right) - 1\right)$$

$$\frac{1}{2} w^T w = \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right)^T \left(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)}\right)$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

(inner product)

$$= -\sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

Dual problem

$$\max W(\alpha)$$

$$\text{s.t } \alpha_i \geq 0$$

$$\sum_i y_i \alpha_i = 0$$

Ng 7

If $\sum_i y_i \alpha_i \neq 0$

$\theta_D(\alpha) = -\infty$

when $\sum y_i \alpha_i = 0$, then $\theta_D(\alpha) = W(\alpha)$

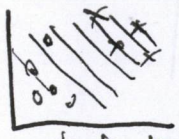
$\therefore \max_{\alpha \geq 0} \theta_D(\alpha)$

So, we've derived a dual optimization problem

For SVM, we'll solve dual opt. problem for α^*

$$W = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

Once solved for α, work out W; then easy to solve for b:



b_1, b_2, b_3, b_4

$$b = \frac{\max_{i:y^{(i)}=-1} w^T x^{(i)} + \min_{i:y^{(i)}=1} w^T x^{(i)}}{2}$$

ie. find worst + best on either side, & is halfway

(2) $W(\alpha)$: you can do everything you want without ever needing to represent $x^{(i)}$ directly

(3) α_i are non-zero only for support vectors. W will be represented only using support vectors

For next lecture

$$W = \sum_i \alpha_i y^{(i)} x^{(i)}$$

Express entire algorithm as inner products

$$h_{w,b}(x) = g(w^T x + b)$$

$$= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b$$

↑ values you are predicting

Kernels:

Training e.g. may be very high dimensional features may be ∞ dim ($x^{(i)} \in \mathbb{R}^\infty$)

1'12"

(1) But despite this, it will turn out that there will be an implicit representation $\langle x^{(i)}, x^{(j)} \rangle$ that will allow you to compute inner products like these sufficiently. $(x^{(i)})^T (x)$

This holds only for certain feature spaces.

- kernels allow efficient comp^n of inner products.
- you can make predictions by making use of these inner products

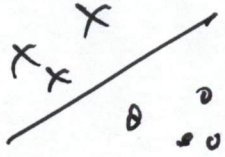
[Ng 8] 41 K views SVM - continues

SVM - kernels
- soft margins (non-linear)
- SMO algorithm

Recap

$$\min \frac{1}{2}\|w\|^2$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1$$

Assuming linear separable, & maxes the margin.



Dual problem

$$\max \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

(inner prod of $x^{(i)} x^{(j)}$: $x^{(i)T} x^{(j)}$)

$$\text{s.t. } \alpha_i \geq 0$$
$$\sum_i y_i \alpha_i = 0$$

$$w = \sum_i \alpha_i y^{(i)} x^{(i)}$$

$$h_{w,b}(x) = g(w^T x + b) = g\left(\sum \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b\right)$$

Kernels (1)

The only dependence of algorithm on x is thru the inner products (of the input feature)

Have $x \in \mathbb{R}$ e.g. living area

Take x & map it to

$$x \to \begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \phi(x)$$

Replace $\langle x^{(i)}, x^{(j)} \rangle$ with $\langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$

Sometimes $\phi(x)$ is very high dimensional, sometimes ∞ dimension. Then you can't compute them!

"If turns out that in many important special cases, we can write down a kernel f^n"

$$K(x^{(i)}, x^{(j)}) = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$$

& compute it very inexpensively

Idea of SVM is that everywhere you see these inner products, replace it a kernel function, & that lets you work in feature space $\phi(x)$, even if $\phi(x)$ is high dim.

Ng·8

Kernels

e.g. 2 inputs

$x, z \in \mathbb{R}^n$

$$K(x,z) = (x^T z)^2 = \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i x_j)(z_i z_j) = (\phi(x)^T)(\phi(z))$$

$K(x,z)$ corresponds to a feature mapping

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

Need $O(n^2)$ to compute $\phi(x)$

But only need $O(n)$ time to compute $K(x,z)$

Define $K(x,z) = (x^T z + c)^2$

other e.g. of kernels

(2)

$K(x,z) = (x^T z + c)^d$

$\rightarrow \binom{n+d}{d}$ features of all monomials up to degree d

↑ high dim feature vector; just compute inner products, & then implicitly working in high dim space

How do you come up with a kernel?

1 intuition: direction

$x \mapsto \phi(x)$; $z \mapsto \phi(z)$

$\langle \phi(x), \phi(z) \rangle$

if x, z are similar, then $\phi(x), \phi(z)$ will be pointing in the same direction; ∴ the inner product will be large;

If I gave you some random thing to classify, one way is to try to come up with a function

$K(x,z)$ – large if x, z similar
– small " " " dissimilar

Ng 8.

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

Is this really a valid kernel? Does there really exist some ϕ s.t. $K(x,z) = \langle \phi(x), \phi(z) \rangle$?

Nec & suff conditions are given by:

- Suppose K is a valid kernel, Let $\{x^{(1)} \dots x^{(m)}\}$ be given: Let $K \in \mathbb{R}^{m \times m}$ matrix.

$$K_{ij} = K(x^{(i)}, x^{(j)})$$

Then, for any vector $z \in \mathbb{R}^m$

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\ &= \sum_i \sum_j \sum_k z_i \phi(x^{(i)})_k \phi(x^{(j)})_k z_j \\ &= \sum_k \sum_i \sum_j z_i \phi(x^{(i)})_k \phi(x^{(j)})_k z_j \\ &= \sum_k \sum_i z_i \phi(x^{(i)})_k^2 \geq 0 \end{aligned}$$

i.e. positive semi-definite matrix

Converse also holds

Theorem (Mercer). Let $K(x,z)$ ∃. Then K is a valid (Mercer) kernel. ie ∃ ϕ s.t. $K(x,z) = \phi(x)^T \phi(z)$ iff for all $\{x^{(1)} \dots, x^{(m)}\}$, $m < \infty$ then kernel matrix $K \in \mathbb{R}^{m \times m}$ is symmetric positive semi-definite;

Useful test for whether any function you choose is a kernel.

e.g. $K(x,x) = -1$: not a valid kernel ∵ $\neq \phi(x)^T \phi(x)$

Ng · 8

To tie back to SVM

e.g. choose $K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$

("Gaussian kernel")

or $(x^T z + c)^d$, etc

choice depends on the problem;

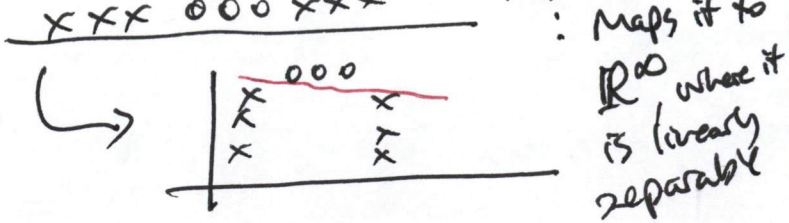
Go back to dual form^n;

Replace $\langle x^{(i)}, x^{(j)} \rangle$ with $K(x^{(i)}, x^{(j)})$

Replace feature vector x with high dim feature vector.

Turns out Gaussian kernel is an ∞ dim feature vector. But you can run SVM in finite time. You never need to represent ∞ dim feature vectors

Why is this a good idea? We started out saying we wanted non-linear learning algo.



(4)

SVM in ∞ space, find the optimal margin classifier; i.e. the classifier that separates data w/ the largest possible geometric margin

This is how they output non-linear decision boundaries. In the entire process, all you need to do is solve convex optimis^n problems

Q&A

You don't know apriori they are linearly separable. This algorithm won't work if not separable.

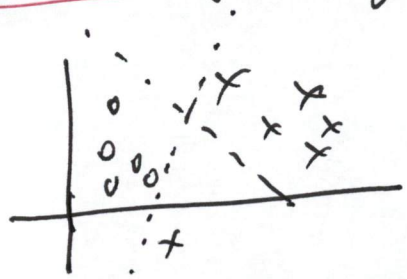
Idea of kernels is more general than SVM. The dual allowed us to write the entire algorithm using $\langle\ \rangle$ products. You can take most algorithms & rewrite them in terms of there inner products. Then you can replace with $K(x^{(i)} x^{(j)})$, & have the algorithm work in high dim

(kernelize them)

Ng8.

L1 Non-linear soft margin SVM

L1 norm soft margin SVM



deals ~ that bottom x; or with data that is not linearly sep.

SVM Primal problem was:

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } y^{(i)}(w^T x + b) \geq 1 - \xi_i; \quad \xi \geq 0 \quad i = 1 \dots m$$

Modify by adding in red; the penalty terms ξ_i; & demand that each train example be separate by $1-\xi_i$.

Remember: if $y^{(i)}(w^T(x^{(i)}) + b) > 0 \Rightarrow$ classified correctly;

∴ Allowing SVM to misclassify some of the time, but $C\Sigma\dots$ penalizes for large ξ_i

It turns out that

Dual of this problem can be done

1. construct ~~generator~~ generalized Lagrangian

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}\|w\|^2 + C\sum\xi_i$$

↑ 2 multipliers

$$\sum_{i=1}^{m}\alpha_i\left(y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i\right) - \sum_{i=1}^{m} r_i\xi_i$$

$$\max W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_i\sum_j y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle$$

$$\text{s.t. } \sum_{i=1}^{m} y^{(i)}\alpha = 0$$

$$0 \leq \alpha_i \leq C \quad i = 1 \dots m$$

Aside

Use KKT conditions; how do we know α_i have converged to optimum?

$\alpha_i = 0 \Rightarrow y^{(i)}(w^T(x)^{(i)} + b) \geq 1$

$\alpha_i = C \Rightarrow y^{(i)}(w^T(x)^{(i)} + b) \leq 1$

$0 < \alpha_i \leq C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1$

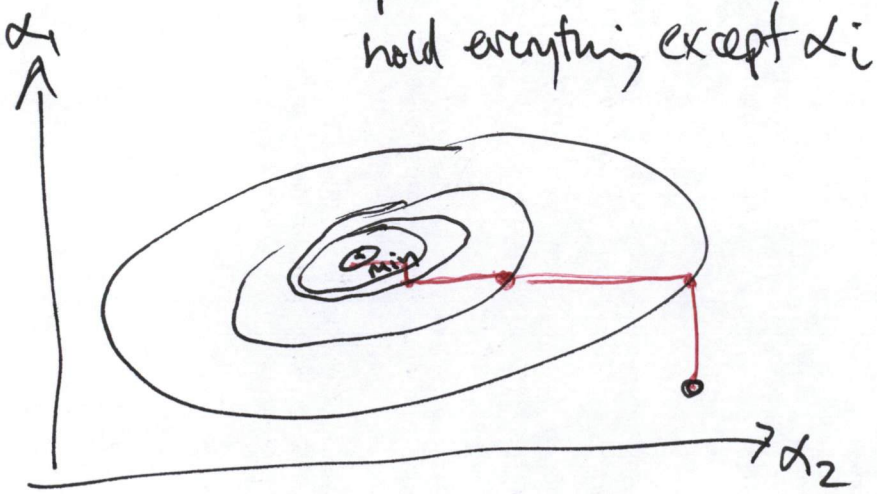
These are convergence conditions

Ng 8

Algorithm for solving optimization problem : coordinate ascent

The problem
Max $W(\alpha_1, \ldots, \alpha_m)$, no constraints

Repeat {(to convergence)
  For $i = 1$ to m
    $\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1 \ldots \hat{\alpha}_i, \alpha_{i+1} \ldots \alpha_m)$
    ↑ hold everything except α_i fixed;



May have heuristic to determine what order to visit α_i

Compared to Newton's method, CA might take more steps. But sometimes inexpensive to

Optimize $W(\alpha_1, \hat{\alpha}_i, \ldots \alpha_m)$; for one α.

Apply to SVM dual optimization : SMO

- in basic form CA doesn't work:

$\sum_{i=1}^{m} y^{(i)} \alpha_i = 0$; if you fix all the α's but one, you can't

Sequential minimal optimization : (SMO)

↖ Microsoft John Platt

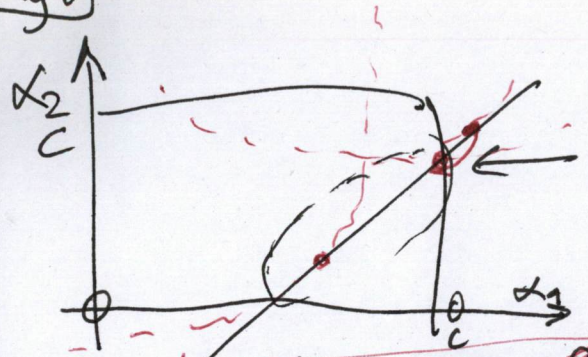
- Choose 2 α_i at a time

Outline
  select α_i, α_j (heuristic)
  hold all α_k fixed except α_i, α_j
  { optimize $W(\alpha)$ w.r.t α_i, α_j. s.t all constraints

Run until you satisfy convergence constraints.

Inner loop can be done efficiently

Update α_1, α_2;
Know $\sum \alpha_i y^{(i)} = 0$
$\alpha_1 y_1 + \alpha_2 y^2 = -\sum_{i=3}^{m} \alpha_i y^{(i)} = \zeta$



$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta$

$\Rightarrow \alpha_i = \frac{\zeta - \alpha_2 y^{(2)}}{y^{(1)}}$

$W(\alpha_1, \ldots \alpha_m) \ldots$

$= W\left(\frac{\zeta - \alpha_2 y^2}{y^{(1)}}, \alpha_2, \alpha_3 \ldots\right)$

$= a\alpha_2^2 + b\alpha_2 + c$; easy to optimize

1 D quadratic

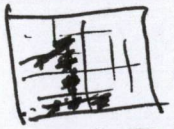
On every iteration, choose 2 α to update; $W(\alpha)$ has to increase or stay the same;

Heuristic for choosing α_1, α_2 is complicated

Easy to read;

To wrap up : applications

e.g. handwritten digit recognition



$x \in \mathbb{R}^{100}$: 100 features

Neural network was champion — ref?

But SVM gave best results

$K(x,y) = (x^T y)^d$ — polynomial

or $\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$ -

But SVM doesn't know about adjacency;

The pixels could be shuffled.

e.g. protein seq classification

20 amino acids, A ... Z

Protein : BAJTSTAIBAJTAU

Assign protein to class

$\phi(x) = ?$

Write down all combs of 4 amino acids

AAAA $\begin{bmatrix} 0 \\ 1 \\ \vdots \end{bmatrix}$

AAAB

Count occurrences of sub-seq :

Ng 8

$\phi(x) \in \mathbb{R}^{(2^4)}$

$\in \mathbb{R}^{160,000}$ features

Dyn programming: $\phi(x)^T \phi(z)$

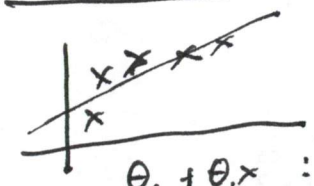
algorithm can compute

SVM one of the most effective off-the shelf learning algorithm;

Ng Lecture 9 "Learning theory" 29,275

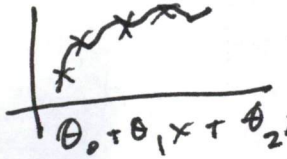
What separates people who really get ML... vs the person who read a textbook

Bias-Variance tradeoff

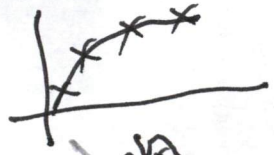


OLS – high bias

$\theta_0 + \theta_1 x$: "underfit" = bias : tons of data, but still doesn't fit.



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$: overfitting = high variance
- spurious patterns in the data



: happy medium

$h_\theta(x) = g(\theta_0 + \theta_1 x \dots \theta_{10} x^{10})$: Also for classif^n problems

Formal model of ML

eg Linear classif^n : motivating example

$h_\theta(x) = g(\theta^T x)$

$g(z) = I\{z \geq 0\}$ (note $y \in \{0,1\}$: notation change)

Given $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$ $(x^{(i)}, y^{(i)}) \sim$ IDD from some probability distⁿ

(1)

Training error of h_θ:

$$\hat{\varepsilon}(h_\theta) = \hat{\varepsilon}_S(h_\theta) = \frac{1}{m} \sum_{i=1}^m I\{h_\theta(x^{(i)}) \neq y^{(i)}\}$$

(a.k.a. risk)

ERM algorithm

empirical risk minimization

$$\hat{\theta} = \arg\min_\theta \hat{\varepsilon}_S(h_\theta)$$: minimize training error

Log reg; & SVM can be viewed this way; can be viewed as approximⁿ to this non-convex optimisⁿ problem; they find a convex approx.

Re-write ERM

see ML algo as "chooses a fⁿ from a class"

Hypothesis class $\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}^{n+1}\}$

$h_\theta : X \to \{0,1\}$: each h_θ is a mapping from X to $\{0,1\}$

As you vary θ, you get different fⁿ

ERM : $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}_S(h)$

↑ can be diff classes of fⁿ

Ng 9

Generaliz^{n} : not training error, but pred^{n} :

$$\varepsilon(h) = P_{(x,y)\sim \mathcal{D}}(h(x) \neq y)$$

↑

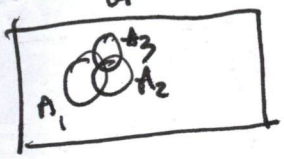
Generaliz^{n} error is what we care about:

Lemma 1

Union Bound: Let $A_1, A_2 \dots A_k$ be k events, (not necessarily independent) then

$$P(A_1 \cup A_2 \cup \dots \cup A_k) \leq P(A_1) + P(A_2) \dots + P(A_k)$$

↑ or

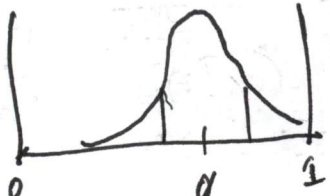


Lemma 2

Hoeffding inequality: Let $z_1, z_2 \dots z_m$ be $\sim$ IDD Bernoulli (ϕ) random var ($P(z_i = 1) = \phi$)

Let $\hat{\phi} = \frac{1}{m}\sum_{i=1}^{m} z_i$ & let any $\gamma > 0$ be fixed

Then $P(|\hat{\phi} - \phi| > \gamma) \leq 2\exp(-2\gamma^2 m)$



As you increase m, the width of tails will decrease exponentially

The case of finite $\mathcal{H}$

$\mathcal{H} = \{h_1, h_2 \dots h_k\}$: k hypotheses

$\hat{h} = \underset{h_i \in \mathcal{H}}{\operatorname{argmin}}\ \hat{\varepsilon}_s(h_i)$

Want to show that ERM helps our general^{n} error

Strategy: (i) $\hat{\varepsilon} \approx \varepsilon$

(ii) implies bound of $\varepsilon(\hat{h})$

Fix any $h_j \in \mathcal{H}$

Define $Z_i = \mathbb{1}\{h_j(x^{(i)}) \neq y^{(i)}\} \in \{0, 1\}$ ↑ misclassify

Z_i is Bernoulli rand var

$$P(Z_i = 1) = \varepsilon(h_j)$$

$\therefore Z_i$ are iid

$$\hat{\varepsilon}(h_j) = \frac{1}{m}\sum_{i=1}^{m} Z_i = \frac{1}{m}\sum_{i=1}^{m} 1\{h_j(x^{(i)}) \neq y^{(i)}\}$$

↑ mean $\varepsilon(h_j)$; ie. a

By Hoeffding inequality:

$$P(|h_j - \hat{\varepsilon}(h_j)| > \gamma) \leq 2e^{-2\gamma^2 m}$$

Proves that for fixed h_j, my training error for large m, will be bounded

Ng 9

New Prove for all h_j

A_j = event that $|\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma$

$P(A_j) \leq 2e^{-2\gamma^2 m}$

$P(\exists h_j \in \mathcal{H} \ldots)$ (hypothesis that I make a large error)

$= P(A_1 \cup A_2 \cup \ldots A_k) \leq \sum_{i=1}^{k} P(A_i)$

$\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) = -2ke^{-2\gamma^2 m}$

1 − both sides: i.e. true hypothesis = no error

$P(\text{not } \exists h_j \in \mathcal{H} \ldots)$

$= P(\forall h_j \in \mathcal{H} \; |\varepsilon(h_j) - \hat{\varepsilon}(h_j)| \leq \gamma)$

$\geq 1 - 2ke^{-2\gamma^2 m}$

* So with prob $1 - 2ke^{-2\gamma^2 m}$, we have $\hat{\varepsilon}(h)$ will be within γ of $\varepsilon(h)$ for all $h \in \mathcal{H}$

"Uniform convergence result"; as $m \uparrow$, $\hat{\varepsilon}(h)$ will all converge to $\varepsilon(h)$ (gen = error)

Highly probable

Remark uniform convergence bound

Given γ, δ, what is m? (training set size)

$\delta = 2ke^{-2\gamma^2 m}$, solve for m:

So long as $m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$

then w.p $1-\delta$, we have $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$

"Sample complexity" bound ($\forall k$, $\log k \leq 30$)

i.e. number of training examples don't grow fast as $\log$.

"error bound!"

Solve for γ, for fixed m, δ

w.p $1-\delta$, we have that $\forall h \in \mathcal{H}$

$|\hat{\varepsilon}(h) - \varepsilon(h)| \leq \sqrt{\frac{1}{2m} \log \frac{2k}{m}}$

2nd step of overall proof.

Lets assume $\forall h \in \mathcal{H} \; |\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ eqn (1)

(unif. converg)

Can we prove sth about $\varepsilon(\hat{h})$? (gen error)

$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \hat{\varepsilon}(h)$ ERM eqn (2)

Ng 9

$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ \varepsilon(h)$: best hypothesis for (3) gen. error.

$$\varepsilon(\hat{h}) \leq \hat{\varepsilon}(\hat{h}) + \gamma \quad \text{by (1)}$$
$$\leq \hat{\varepsilon}(h^*) + \gamma \quad \text{by (2)}$$
$$\leq \varepsilon(h^*) + \gamma + \gamma \quad \text{by (1)}$$

n.b.
$\hat{\varepsilon}(h)$ = training error of h
$\varepsilon(h)$ = gen error of h

Theorem: Let $|\mathcal{H}| = k$, & let m, δ be fixed, then w.p $1-\delta$,

$$\varepsilon(\hat{h}) \leq \underbrace{\left(\min_{h \in \mathcal{H}} \varepsilon(h)\right)}_{\varepsilon(h^*)} + 2\sqrt{\frac{1}{2m}\log\frac{2k}{\delta}} \quad \text{(eqn)}^*$$

To prove:

$\gamma = \sqrt{\frac{1}{2m}\log\frac{2k}{\delta}}$ We know eqn (1) holds w.p $1-\delta$

which implies eqn (*)

Result helps deal with bias – tradeoff error

Let's say have,

$\mathcal{H}$ ↑ $\mathcal{H}'$

linear → quadratic

considering switching to quadratic, tradeoff.

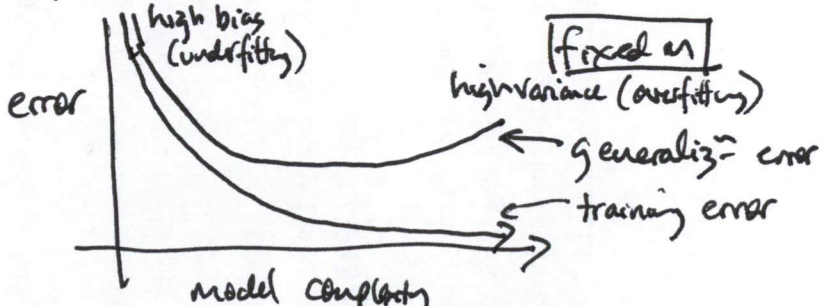
$\varepsilon(h^*)$ will become better (best quadratic)

but k will increase (i.e. risk of not fitting is worse)

In eqn (*), 1st term is bias; 2nd is variance

So complex class will make variance worse.

(for classif^n problems, no generally accepted defn of bias & variance – n.b)



Often preferable to have intermediate complexity

Ng's

Corollary: Let $|\mathcal{H}| = k$; Let any δ, γ be fixed

Then, suppose I want to guarantee

$$\varepsilon(\hat{h}) \leq \min_{h \in \mathcal{H}} \varepsilon(h) + 2\gamma$$

w.p $1 - \delta$. It suffices that:

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$$

$$= O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta}\right)$$

* convinces yourself that this is a true fact & does follow from the other one

Ng 10 Models

Vtrain = 25,089

VC dimension

Model selection -

Bias-var - cross-valid^n

- feature selection

Bayesian statistics & regulariz^n

In order to get guarantee, if suffices

$$m \geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta}\right)$$

$$= O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta}\right)$$

Number of training examples is roughly linear in number of parameters in $\mathcal{H}$.

Recap

Let $|\mathcal{H}| = k$, & let γ, δ be fixed,

then for $\varepsilon(\hat{h}) \leq \min_{h \in \mathcal{H}} \varepsilon(h) + 2\gamma$, w.p $1-\delta$, suffices

that $m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta} = O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta}\right)$

ERM : empir. risk. min : Choos $\hat{h}$ that min ~~training~~ generaliz^n error

$\varepsilon(\hat{h})$: generaliz^n error

m : training set size

k : size of hypothesis class

gives bound on the number of training examples needed

To prove formally,

defn: Given a set of points $S = \{x^{(1)} \dots x^{(d)}\}$

$\mathcal{H}$ "shatters" set S if $\mathcal{H}$ can realize any labelling on it: i.e. I can choose perfect labels for S.

e.g $\mathcal{H} = \{$linear classifiers in 2D$\}$

e.g. S, 2 points — 4 possible labellings, $\exists \hat{h}$ that labels perfectly $\therefore$ $\mathcal{H}$ shatters S

e.g. S, 3 points

eg. S, 4 points ⇐ cannot be shattered by $\mathcal{H}$

Generalize to "∞ $\mathcal{H}$ class":

e.g. log reg has $\mathbb{R}$; ∞ $\mathcal{H}$ class

Say $\mathcal{H}$ is parameterized by d real numbers

e.g. $\mathcal{H}$ is class of all linear decision boundaries i.e. by 64d bits;

$k = |\mathcal{H}| = 2^{64d}$;

Ng 10

Defn : V-C dimension (Vapnik-Chervonenkis) : $VC(\mathcal{H})$ is the size of the largest set shattered by $\mathcal{H}$.

E.g. $\mathcal{H}$ is {class of linear classifier in 2D},

$VC(\mathcal{H}) = 3$

More generally, in n dim, the VC of {linear classifier in n-d} $= n+1$

Arguably best known result in learning theory (0:18.00)

Let $\mathcal{H}$ be given, & let $VC(\mathcal{H}) = d$.

Then w.p $(1-\delta)$, we have that for $\forall h \in \mathcal{H}$

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq O\left(\sqrt{\frac{d}{m}\log\left(\frac{m}{d}\right) + \frac{1}{m}\log\left(\frac{1}{\delta}\right)}\right)$$

($\varepsilon(h)$: gen error; $\hat{\varepsilon}(h)$: train error)

Thus, w.p $(1-\delta)$, we have that

$$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

Corollary

To guarantee that $\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$ w.p $(1-\delta)$ it suffices that

$$m = O_{\gamma,\delta}(d)$$

Training egs needed are roughly linear with (2) VC dimension; usually similar to number of parameters; "upper-bounded by VC dim".

How come SVM don't overfit?

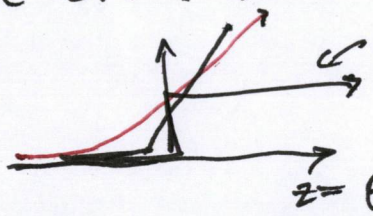
i.e. $VC \to \infty$? If $\|x\|_2 \leq R$

$\mathcal{H}$ of large margin separ^{n} ($\geq \gamma$)

$$VC(\mathcal{H}) \leq \left[\frac{R^2}{\gamma^2}\right] + 1$$

By using large margin, $\mathcal{H}$ of small VC dim;

Tie ERM back to algorithms:



Choose θ to min this step f^{n}; i.e. $\mathbb{1}(h_\theta(x)) \approx 0$

but this is non-convex

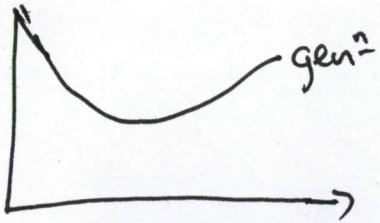
Lin reg & SVM can be viewed as convex approx to this step f^{n}; (red line)

SVM is precursor loss: "hinge-loss f^{n}"

∴ All the theory of ERM gives good intuition for what log reg & SVM are doing; & vice-versa

Ng 10 **Model selection (M.S)**

M.S algs automatically tradeoff b/w bias & variand



e.g.

choose $\begin{cases} \theta_0 + \theta_1 x \\ \theta_0 + \theta_1 x + \theta_2 x^2 \\ \vdots \\ \theta_0 + \theta_1 x \dots \theta_0 x^{10} \end{cases}$

e.g. choose τ - bandwidth param in LWLR

e.g. choose C in SVM in $\min \frac{1}{2}\|w\|^2$

How to choose these models? automatically

$\mathcal{M} = \{M_1, M_2, M_3 \dots\}$

↑ diff models

idea: train them all & look for training error min^n

- no, overfit;

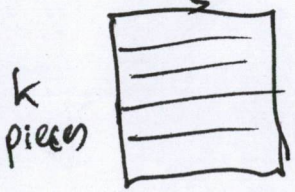
Better ideas (B)

Hold-out cross-valid^n

- Split S into S_{train} (70%)
  S_{cv} (30%) ← hold-out
- train each model on S_{Train}, & test on S_{cv};
- pick lowest error on S_{cv} model.
- optionally; retrain this best model on the whole S.

HCV works out great, but you often have very few training e.g^s. Painful to hold out cv.

∴ **k-fold CValid^n**



- Train on k-1 pieces; test on the remaining piece
- Average over k-results & their error

- usually go back & train on whole S
- k = 10 is common; holdout 10%
- more expensive 0; ~~fpg~~

~~&~~ **leave-one-out CVal^n**

k = m: just leave one out each time;

- 0 is high; use for scarce training S.

(Ng 10)

Disc"

Learning theory bounds are often very loose, & hold for any prob dist"; actually learning theory bounds will bring out extremely pessimistic numbers;

- Attempts to use LT bounds as guidelines, but I tend to use them as shape of intuition.
- Trying to use logreg, if m is $10 \times p$, then probably in good shape

Feature selection

Many ML have high dim feature space, e.g. text class^{fn} ~ 50×10^3 p.

Reducing number of features might reduce variance, & risk of overfitting

∴ Select feature subset that are relevant, & reduce risk of overfitting

With n features, 2^n possible subsets

Use "search heuristics" to find subset of features

- Forward search / select^{n} : 1 feature at a time
  - start with $\mathcal{F} = \emptyset$
  - Repeat ε
    (i) For $i = 1 \dots n$, try add feature i to $\mathcal{F}$ & evaluate model using CV^{n} (any flavor)
    (ii) Set $\mathcal{F} = \mathcal{F} \cup \{$ best feature found in (i) $\}$ → lowest CV error
  - Output best hypothesis found.

"Wrapper" feature selection : repeatedly calls learning algorithm; comput^{n}ally expensive; often work well;

Backward selection

- start ~~$\mathcal{F} =$~~ $\{1 \dots n\}$
  delete features one at a time
- if you have 100 n, 10^4 p, then don't use this

- NP-hard problem to find best feature set;

Ng ⑩

Less computationally expensive:

"filter" feature selection method

for each feature i, compute some measure of how informative x_i is about y.

e.g. $\text{corr}(y, x_i)$; pick top k most correlated features;

For discrete features:

e.g. $MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$

($p(x_i, y)$: estimate from training data)

MI: mutual information

$= KL\left(p(x_i, y) \,\|\, p(x)p(y)\right)$

KL: KL divergence; formal measure of similarity of distributions

MI: how diff joint dist^n of $P(x,y)$ is from independent dist^n; how non-independent are they;

Then pick top k features — largest MI.

To choose k, use cross-validation to decide how many features to use;

Ng 11

28,891
Jul 22, 2008

Bayesian stats & regularizⁿ
Online learning (digression)
applying ML

Regulⁿ : as a way of preventing overfitting

e.g. linear regression

choose θ using max likelihood

: $\max_\theta \prod_i p(y^{(i)} | x_i, \theta)$

i.e. frequentist procedure ; envision a true θ governing ~~housing~~ prices ; θ is not random;

vs Bayesian school

We don't know θ:

$P(\theta)$ = prior E.g $\theta \sim N(0, \tau^2 I)$

$S = \{x^{(i)}, y^{(i)}\}_{i=1}^m$: training set

Calculate

$P(\theta|S) \sim \prod p(y^{(i)}|x^{(i)}, \theta) P(\theta)$

↑ posterior

To predict a new price on x

$$P(y|x,S) = \int_\theta P(y|x,\theta) P(\theta|S) d\theta \quad (a)$$

$$E[y|x,S] = \int_y y P(y|x,S) dy \quad (b)$$

↑ i.e, take an expectⁿ of y, given posterior distⁿ.

for many problems, this is comput= difficult. Integrals over high-d spaces; θ is high-D.

Instead of computing full posterior distⁿ

$$\theta_{MAP} = \arg\max_\theta P(\theta|S)$$

$$= \arg\max_\theta \left(\prod p(y^{(i)}|x^{(i)}, \theta)\right) P(\theta)$$

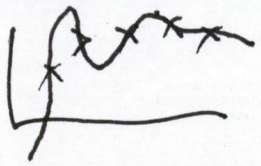
To make a predⁿ

predict $h_{\hat\theta_{MAP}}(x) = \hat\theta_{MAP}^T x$

So, instead maxing the prior?

Intuition : if ur prior is $\theta \sim N(0, \tau^2 I)$, then most of θ will be close to 0. This is like feature selection. The Most mass is close to zero

In pictures



: max likelihood, gives high order with lots of oscill, vs Bayesian regularizⁿ gives a smoother fit

Ng 11

intuition: whereas MLE

$$\min_\theta \sum_i \| y^{(i)} - \theta^T x^{(i)} \|^2$$

with Bayesian reg

$$\min_\theta \sum_i \| y^{(i)} - \theta^T x^{(i)} \|^2 + \lambda \|\theta\|^2$$

- so that $\lambda\|\theta\|^2$ smooths, again $\because$ most of mass close to zero

for text classifier, with Bayesian regul^n, logistic regression becomes effective classifier (see BayesLogit)

DIGRESSION : Online Learning

All algo so far are batch; online make pred^n while learning.

$x^{(1)}$ → guess → $\hat{y}^{(1)}$ → $y^{(1)}$ → $x^{(2)}$ → $\hat{y}^{(2)}$ → $y^{(2)}$ → $x^{(3)}$ → $\hat{y}^{(3)}$ → ...

Total online error:

$$\sum_{i=1}^{m} 1\{\hat{y}^{(i)} \neq y^{(i)}\}$$

Adapt SGDes, a perceptron to do online learning: (2)

initialize $\theta = 0$

after i^{th} eg, update, run 1 step of SGD:

$$\theta = \theta + \alpha (y^{(i)} - h_\theta(x^{(i)})) \cdot x^{(i)}$$

You can prove that: even $\because$

$x^{(i)} \in \mathbb{R}^\infty$ (infinite feature)

so long as some margin γ separates (+) & (-) eg, perceptron will converge to perfect & sep^n, even tho' you are in an ∞-dim space.

Applying ML algo 00:25

I want to give you advice of how to apply really powerfully. Convey to you methods on how to get them to work well. You just want to get it to work.

3 key ideas : diagnostics
- error & ablative analysis

how to get started

- premature statistical optimiz^n

Ng 11

~~Debugging learning~~ Debugging ML algs

e.g. anti-spam: Choose 100 features;
Use Bayesian log reg with grad descent.
& you get 20% ~~true~~ test error

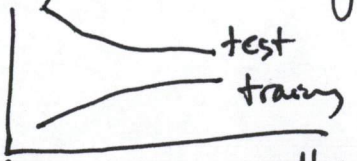
Bayes log reg:
$$\max_{\theta} \sum_{i=1} \log p(y^{(i)} | x^{(i)}, \theta) - \lambda \|\theta\|^2.$$

Do 8 things - endless options
- Get more training sets?
- try diff feature set, ... fixing diff problems

But takes a lot of time.
Better to figure out what to fix:

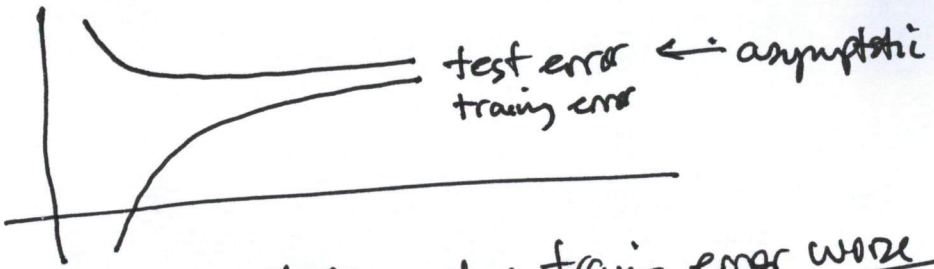
e.g. high bias or high variance
diagnostic: high variance: training error will be much lower than test error
high bias: train error will be higher

For high variance, training error will grow with higher m?



So more training data ~~won't help~~ will help. Also smaller feature set

for high bias

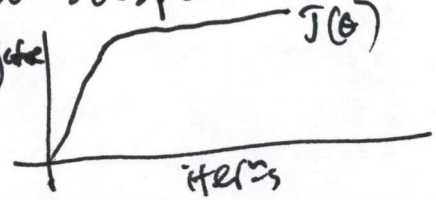


more training data makes training error worse

∴ try more features, not more data, since that can't help: This happens a lot. You could have told them 6 months ago. Somewhat depressing.

Bias vs variance comes up often. A common diagnostic
Often need to construct your own diagnostic

Another example: SVM outperforms log reg
Bayesian log reg: 2% rejection of non-spam; is too high; 2% " " spam error
SVM gives 10% of spam; 0.01% of non-spam
But you want to use Bayes log reg.
You suspect it is not converging



has it converged?

Ng 4

Ask - what do you care about? Are you optimizing for the right thing? i.e. the objective

SVM outperforms BLR, but you want to use BLR

Weighted accuracy: you care about:

$$a(\theta) = \max_\theta \sum_i w^{(i)} 1\{h_\theta(x^{(i)}) = y^{(i)}\}$$

θ_{SVM} outperforms θ_{BLR}. So:

$$a(\theta_{SVM}) > a(\theta_{BLR})$$

BLR tries to maximize

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)} | x^{(i)}, \theta) - \lambda \|\theta\|^2$$

Diagnostic

$$J(\theta_{SVM}) > J(\theta_{BLR})?$$

Case 1:

$$a(\theta_{SVM}) > a(\theta_{BLR})$$
$$J(\;\;\;) > J(\theta_{BLR})$$

But BLR was trying to max $J(\theta)$. $\therefore \theta_{BLR}$ didn't maximize $J(\theta)$. $\therefore$ Optimiz^n didn't converge

Case 2: $a(\theta_{SVM}) > a(\theta_{BLR})$

$$J(\theta_{SVM}) < J(\theta_{BLR})$$

$\therefore$ BLR maxed $J(\theta)$, but SVM does better on weighted accuracy $a(\theta)$. $\therefore$ $J(\theta)$ is the wrong f^n to max; the wrong objective.

- fix optimiz^n algorithm — more iterations / change method
- " " objective — use SVM

Example of diagnostic for helicopter 00:52:00

Choose cost f^n: $J(\theta) = \|x - x_{desired}\|^2$

Use reinforcement learning:

$$\theta_{RL} = \arg\max_\theta J(\theta)$$

Say it gives bad performance, what do you do? ($\vee \theta_{human}$)

Diagnostics

1. If θ_{RL} works in simulator but not actually then the simulator is broken
2. Let θ_{HUMAN}. If $J(\theta_{human}) < J(\theta_{RL})$, then the problem is the RL algorithm. θ_{RL} is not small enough
3. If $J(\theta_{HUMAN}) > J(\theta_{RL})$, then the problem is the cost function

Ng 12

More on diagnostics

Use to figure out what is happening in an algorith

"Some of you will graduate from Stanford & get fabulously high jobs applying ML to significant economic appln" 1:01:12

> You should get an intuitive understanding of the problem; your own personal understandi

> Some Silicon valley outsourced their ML. Often a terrible idea. You don't keep the knowledg

> Diagnostics & error analysis help justify the problem & your analysis

Error ~~Ana~~ Analysis

Actually, many applic^{n} combine learning components in a pipeline. e.g. ~~Insta~~ face-recog^{n} analysis.

How much error is attributed to each compnt?

- plug in ground truth for each component, & see how accuracy changes; make table,

& looks for major sources of error

(5)

Ablative analysis

> tries to explain differnce btw baseline & current performance.

> e.g. anti-spam classifier, has lots of compnts → 94.9

LR → 94%

then remove components & see how performa drops.

Getting started on a learning problem

#1 : careful design - best for research

#2 : build & fix : a hack

- works quickly
- use error analysis & diagnostics
- faster to market
- avoids premature optimiz^{n}

Step 1 : plot the data every way you can think of; half the time you find something wrong w the data

I use LR most often

Ng 11

Danger of over-theorizing

Papadimitriou 1995

Theory of VC dimension
NP hardness
] very important & impactful

1:22:00

I personally only work on ML with theory if I can see a link;

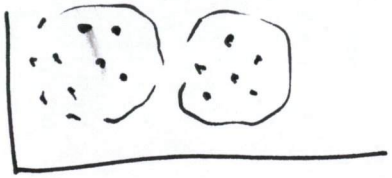
Spend 30-50% on diagnostics

Ng 12

Unsupervised learning (UL) 28,406

- clustering (k-means)
- mixture of Gaussians
- Jensen's inequality
- EM (Expect. Max^n)

In UL, you are given a dataset with no labels. The ML algo has to discover structure



Applications

biology – genes.
Market research – customers
news.google.com – group related news
image segmentn

K-means algorithm

Input $\{x^{(1)}, x^{(2)} \dots x^{(3)}\}$

(no class labels, y)!

1. Initialize cluster centroids

$\mu_1 \dots \mu_k \in \mathbb{R}^n$: vectors in $\mathbb{R}^n$

2. Repeat until convergence:

(i) Set $c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|$: assign point

(ii) $\mu_j := \dfrac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$: shift centroid

Shows:
Anim^n from Michael Jordan

Guaranteed to converge

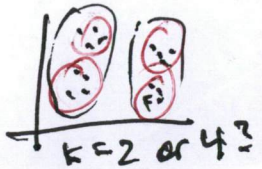
Distortion fn:

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Can show that k-means is coordinate ascent on J. (Go home & prove this yourself!)
$\therefore J(c, \mu)$ decreases monotonically. K-means can oscillate b/w two diff. clusters. Also local optima.

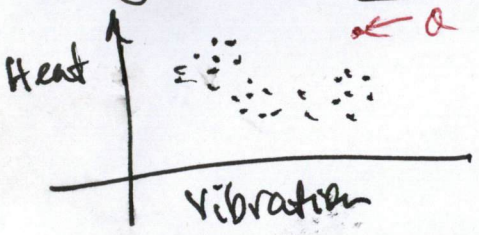
Choose k?

try a few different k;
The true number is ambiguous



Ng 12

Density estimation



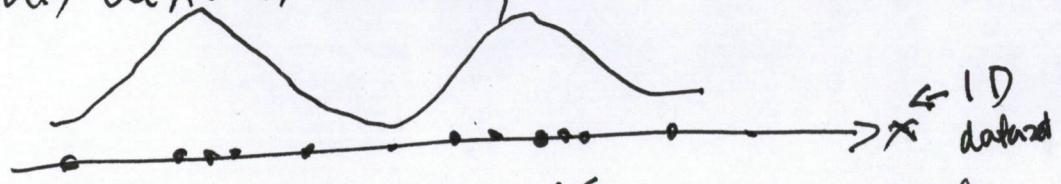
for aircraft engines: you want to estimate joint density to find outliers

Look at typical dist^n, build model of $P(x)$, then if $P(x)$ is very small, that would raise a red flag. i.e. Anomaly detection

Take training set $\{x^{(1)}, \dots, x^{(m)}\}$, build $P(x)$, & use that to detect anomaly

Used in security applic^{ns}

This dist^n is unusual, not a standard one



could be mixture of 2 N: i.e. 2 separate Gaussians that generated this. But I don't know which Gaussian each data point came from.

Gaussian mixtures

(2)

Imagine there's a latent (hidden, unobserved) random variable z. And $x^{(i)}, z^{(i)}$ have a joint dist^n:

$$P(x^{(i)}, z^{(i)}) = P(x^{(i)} | z^{(i)}) P(z^{(i)})$$

by the chain rule

$z^{(i)} \sim \text{Multinomial}(\phi)$; (Bernoulli for 2 Gaussians)

$(\phi_j \geq 0;\ \sum_j \phi_j = 1)$

$x^{(i)} | z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$

(Almost same as Gaussian LDA, but y is replaced by z)

If we knew $z^{(i)}$s, then use MLE:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)$$

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}$$

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}$$

But we don't know z. So guess z using bootstrap, + MLE : EM

Ng 12

EM - Expect^n max algorithm

Repeat {
until converges

E-Step { guess values of $z^{(i)}$'s)

Let $w_j^{(i)} := p(z^{(i)} = j \mid x^{(i)}; \phi, \mu, \Sigma)$

- compute probability

$$= \frac{p(x^{(i)} \mid z^{(i)} = j)\, p(z^{(i)} = j)}{\sum_{l=1}^{k} p(x^{(i)} \mid z^{(i)} = l)\, p(z^{(i)} = l)}$$

$$= \frac{\frac{1}{2\pi |\Sigma_j|} \exp\left((x^{(i)} - \mu_j)^2 \Sigma_j^{-1} (x^{(i)} - \mu_j)\right) \cdot \phi}{\sum_{l=1}^{k} (\ldots}$$

↑ plug in formula for Gaussian

M-Step: { update parameters

$$\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$$

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$$

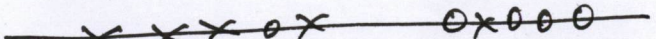
(3)

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^2}{\sum_{i=1}^{m} w \Sigma_j^{(i)}} \quad \}$$

Covariance matrix for different Gaussian can differ (cf GDA)

Recall - [tries to explain max] [00:39:00a]

In GDA: we know the labels:



MLE of ϕ_j (the probability of getting a point from class j) $\phi_j = \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}$

In Mix of Gaussians, no labels: We guess $z^{(i)}$'s, In E-step, $w_j^{(i)}$ is the best guess for point i coming from j. In M-step, $w_j^{(i)}$ is the soft version of the $1\{z^{(i)} = j\}$. - the indicator fn.

Ng 12 **Broader view of EM algorithm**

(Mix of N is a special case)

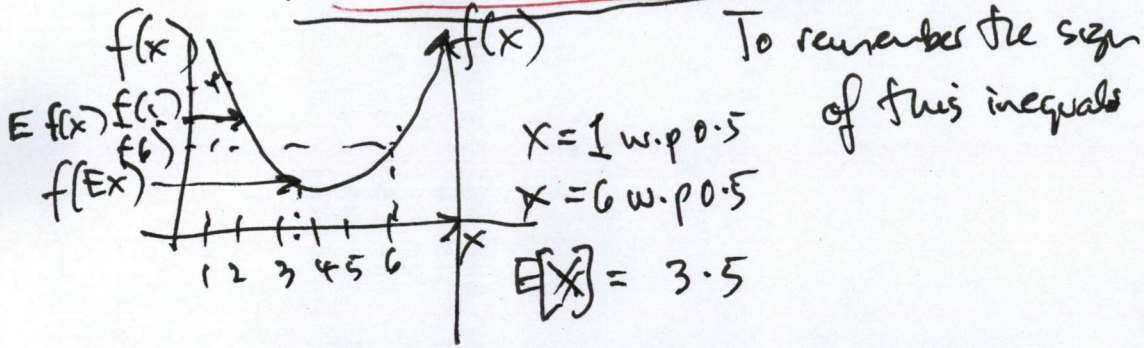
Jensens inequality

Let f be a convex f= (ie. $f''(x) \geq 0$)

Let X " " ranvar

Then $f(E[X]) \leq E[f(x)]$ — Jensen's inequality

To remember the sign of this inequality



Further, if $f''(x) > 0$, (f is strictly convex),

then $E[f(x)] = f(E[x])$

$\Leftrightarrow$ X is a constant w.p. 1

ie $X = E[X]$ w.p. 1.

If $f''(x) < 0$ (concave), then

$f(EX) \geq E[f(X)]$; etc.

Derivation of general EM

Have some model for $P(x, z; \theta)$ - joint dist^{n}.

Observe only X.

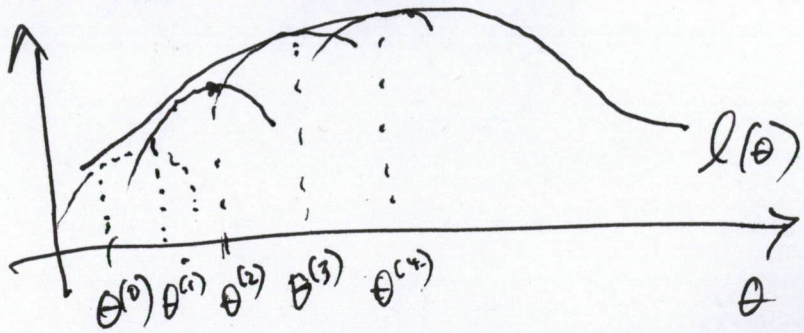
Goal to maximize:

$$l(\theta) = \sum_{i=1}^{m} \log(P(x^{(i)}; \theta))$$

$$= \sum_{i=1}^{m} \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

↑ ie. marginalizing out $z^{(i)}$

EM is a way of doing the ML above



Ng, (2)

$$\max_\theta \sum_i \log P(x^{(i)};\theta)$$

$$= \sum_i \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)};\theta)$$

$$= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}$$

$$\left[Q_i(z^{(i)}) \geq 0; \sum_{z^{(i)}} Q(z^{(i)}) = 1\right]$$

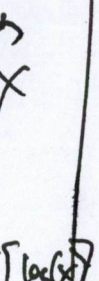
Q_i : a prob dist^n over $z^{(i)}$

$$= \sum_i \log E_{z^{(i)} \sim Q_i}\left[\frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\right]$$

[I'm seeing some frowns. Questions about this? 00:59:03]

$$\geq \sum_i E_{z^{(i)} \sim Q_i}\left[\log \frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\right]$$

← Jensen's inequality



$$= \sum_i \sum_{z^{(i)} \sim Q_i} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}$$

$\log E(x) \geq E[\log(x)]$

(5)

Notice what we've done

We've constructed a lower bound for $\ell(\theta)$. We've shown that:

$$\ell(\theta) \geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}$$

The last thing we want is an equality for $\ell(\theta)$; i.e. a tight lower bound

Need Jensen's inequality to be an equality,

- choose Q so that equality holds for θ

(reminder: if X is a constant, JI becomes "=")

Want:

$$\frac{P(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} = \text{constant}, \quad \text{for all values of } z^{(i)}$$

$Q_i(z^{(i)}) \propto P(x^{(i)}, z^{(i)};\theta)$

$$\sum_{z^{(i)}} Q_i(z^{(i)}) = 1$$

Ng 12

$$Q_i(z^{(i)}) = \frac{P(x^{(i)}, z^{(i)}; \theta)}{\sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)}$$ [see lecture notes for more steps]

$$= \frac{P(x^{(i)}, z^{(i)}; \theta)}{P(x^{(i)}; \theta)}$$

$$= P(z^{(i)} | x^{(i)}; \theta).$$

EM algorithm overview

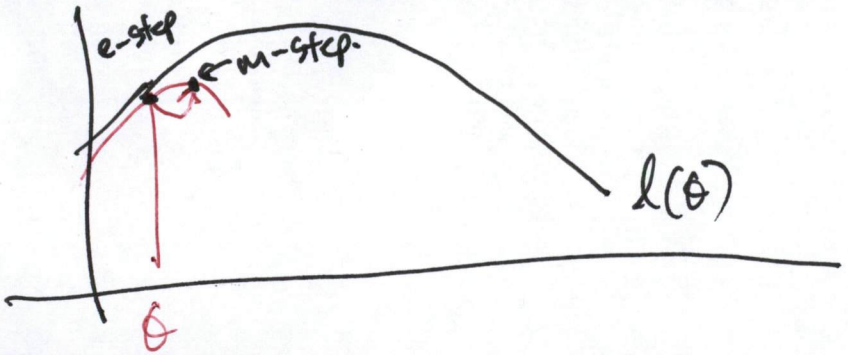
E-step:

Set $Q_i(z^{(i)}) := P(z^{(i)} | x^{(i)}; \theta)$

- i.e tight lower bound

M-step

$$\theta := \arg\max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

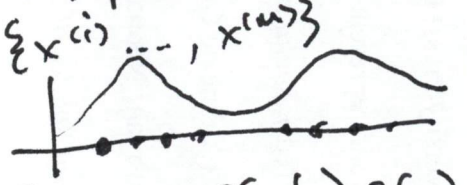


Ng 13

EM
- Mixture of Gaussians
- mixture of naive Bayes
- factor analysis
- Digression: Gaussians dists

23,382

Recap: unsupervised learning



$P(x) = \sum_z P(x|z) P(z)$ ← z is the latent variable that indicates which of 2 Gaussians it came from

$z \sim \text{Multinomial}(\phi)$

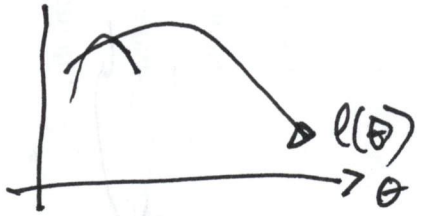
$x|z_j \sim N(\mu_j, \Sigma_j)$

EM $\max_\theta \sum_i \log P(x^{(i)}; \theta)$

$\max_\theta \sum_i \log \sum_z P(x^{(i)}, z^{(i)}; \theta)$

E-step ....

M-step ....



Why do we bother w the EM algo? (1)

Because the M-step is tractably analytically

ie. in closed form.

e.g. for exponential family, $P(x^{(i)}, z^{(i)}; \theta)$ is tractable

One other way of thinking about EM

Define $J(\theta, Q) = \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$

$\ell(\theta) \geq J(\theta, Q)$ (by Jensen's inequal)

Co-ord ascent:

EM is coord ascent on J:

E-step: Max w.r.t. Q

M-step: " " θ

MoG: Mixture of Gaussians

E-step

$w_j^{(i)} \quad Q_i(z^{(i)} = j) = P(z^{(i)} = j | x^{(i)}, \phi, \mu, \Sigma)$

Use Bayes rule:

$= \frac{P(x^{(i)}|z^{(i)}=j) P(z^{(i)}=j)}{\sum_k P(x^{(i)}|z^{(i)}=k) P(z^{(i)}=k)}$

$P(x^{(i)}|z^{(i)}=j)$: we know $= N(\mu_j, \Sigma_j)$

$P(z^{(i)}=j) \to \phi_j$

Ng 13

M-step

$$\max_{\phi,\mu,\Sigma} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)}{Q_i(z^{(i)})}$$

Concretely,

$$= \sum_i \sum_j w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_j)^T \Sigma_j^{-1}(x^{(i)}-\mu_j)\right) \phi_j}{w_j^{(i)}}$$

$w_j^{(i)} = Q_i(z^{(i)}=j)$

Take deriv.

$$\nabla_{\mu_j}(\dots) \stackrel{\text{set}}{=} 0$$

$$\Rightarrow \mu_j = \frac{\sum_i w_j^{(i)} x^{(i)}}{\sum_i w_j^{(i)}}$$

| solve for μ_j, ϕ then for ϕ, Σ, do the updates

Since ϕ is multinomial, $\sum_j \phi_j = 1$ is a constraint.

Constraint maxim: for ϕ_j: construct the Lagrangian:

$$\mathcal{L} = (\dots) + \beta\left(\sum_j \phi_j - 1\right)$$

(…) optimize object; β Lagrange multiplier

$$\frac{\partial}{\partial \phi_j}\mathcal{L} = 0$$

All in the lecture notes:

Ex: text clustering as in news.google.com — group related docs

apply EM:

Mixture of Naive Bayes

train set $\{x^{(1)}, \dots, x^{(m)}\}$

$x^{(i)} \in \{0,1\}^n$ $x_j^{(i)} = 1\{\text{word } j \text{ in doc } i\}$

$z^{(i)} \in \{0,1\}$ 2 clusters

Assume $z^{(i)} \sim \text{Bernoulli}(\phi)$

$$P(x^{(i)} | z^{(i)}) = \prod_{i=1}^n P(x^{(i)} | z^{(i)} = j)$$

$$P(x_j^{(i)} = 1 | z^{(i)} = 0) = \phi_{j|z=0}$$

} basically naive Bayes

E-step:

$$w^{(i)} = P(z^{(i)} = 1 | x^{(i)}; \phi_{j|z}, \phi)$$

M-step

$$\phi_{j|z=1} = \frac{\sum_{i=1}^m w^{(i)} 1\{x_j^{(i)} = 1\}}{\sum_{i=1}^m w^{(i)}}$$

$$\phi_{j|z=0} : \frac{\sum_{i=1}^m (1-w^{(i)}) 1\{x_j^{(i)} = 1\}}{\sum_{i=1}^m (1-w^{(i)})}$$

$$\phi_z = \frac{\sum_i w^{(i)}}{m}$$

document cluster

The intuition here

E-step: compute weights that say whether doc came from cluster 1 or 1

M-step: no hard assignment, but how much do I think it is in the cluster.

Factor analysis model

Not as widely used as MoG, or NBayes.

But mathematically interesting; ∵ continuous z.

of MoG

What about:

$n \approx m$

$n >> m$: high dim → 1000 dim, but only 20 measurements?

What can you do? Fit a Gaussian

$x \sim N(\mu, \Sigma) \quad \Sigma \in \mathbb{R}^{n}$

$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$: eg use MLE.

$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$.

You find the covar matrix will be singular*, not full rank & non-invertible

$n = m = 2$

⇐ infinite thin Gaussian

Contours of Gaussian density

$\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(\frac{1}{2}(x-\mu)\Sigma^{-1}(\ldots)\right)$

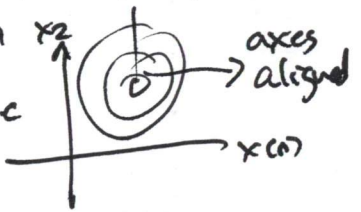
↑ non-invertable :

∴ Not a good model.

What to do?

constrain Σ to be diagonal $x \sim N(\mu, \Sigma)$

$\Sigma = \begin{pmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \ddots & \\ 0 & & & \sigma_n^2 \end{pmatrix} \in \mathbb{R}^{n \times n}$ i.e.



$\sigma_j^2 = \frac{1}{m}\sum_{i} (x_j^{(i)} - \mu_j)^2$

But you've thrown away all the correlations, even though axes aligned

So not a good model

*Singular: determinant is zero

Ng 13

Or constrain $\Sigma = \sigma^2 I = \begin{pmatrix} \sigma^2 & 0 & \cdots \\ 0 & \sigma^2 & \\ \vdots & & \end{pmatrix}$ i.e. circular Gaussian

FA attempts to model correlations b/w variables

Assume latent $z \sim N(0, I)$, $z \in \mathbb{R}^d$

model | $x|z \sim N(\mu + \Lambda z, \Psi)$ $d < n$

Equivalently

$x = \mu + \Lambda z + \varepsilon$ where $\varepsilon \sim N(0, \Psi)$

Parameters: $\mu \in \mathbb{R}^n$, $\Lambda \in \mathbb{R}^{n \times d}$

$\Psi \in \mathbb{R}^{n \times n}$, diagonal.

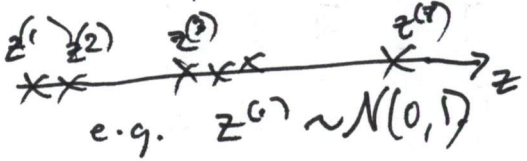
So, e.g.:

$z \in \mathbb{R}^1$, $x \in \mathbb{R}^2$

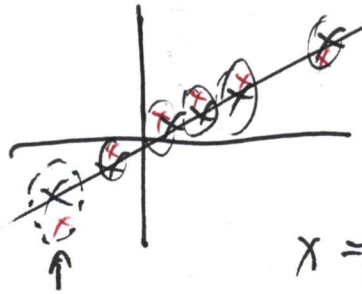
$\Lambda = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$\Psi = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



e.g. $z^{(i)} \sim N(0,1)$



$x = \mu + \Lambda z + \varepsilon$ $\varepsilon \sim N(0, \Sigma)$

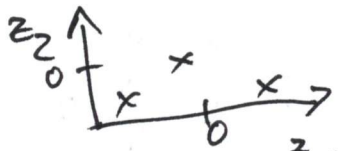
Σ: Gaussians: draw examples from them

In reality, you only see the xs. Informally, think of FA as modelling the data as coming from lower-dim space plus some noise.

So, example

$z \in \mathbb{R}^2$; $x \in \mathbb{R}^3$: z lies in 2D

0:54:01



$\mu + \Lambda z$ moves off the page into the room; a dataset like a fat pancake.

That's the model. How to fit the parameters?

- rewrite Gaussian

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $x_1 \in \mathbb{R}^r$, $x_2 \in \mathbb{R}^s$

$x \in \mathbb{R}^{r+s}$

↑ partition vector.

Ng (3

$x \sim N(\mu, \Sigma)$ where

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

$\Sigma_{12} \in \mathbb{R}^{r \times s}$

Specifies a joint dist^n:

What are the marginal & cond^n dist^n?

$P(x)$

$P(x_1) = \int P(x_1, x_2)\, dx_2$

$x_1 \sim N(\mu_1, \Sigma_1)$ ← marginal is a Gaussian

Conditional dist^n?

$$P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)} \quad \begin{matrix} \leftarrow N(\mu, \Sigma) \\ \leftarrow N(\mu_2, \Sigma_{12}) \end{matrix}$$

Plug in the formula:

$x_1|x_2 \sim N(\mu_{1|2}, \Sigma_{1|2})$

$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$

$\Sigma_{1|2} = \dots$

doesn't show derivⁿ, but in the notes

It turns out that you can go back, [1:02:00]

Under FA model

$$\begin{pmatrix} z \\ x \end{pmatrix} \sim N(\mu_{zx}, \Sigma)$$

$z \sim N(0, I)$, $x = \mu + \Lambda z + \varepsilon$, $\varepsilon \sim N(0, \Psi)$

$E[z] = 0$

$Ex = E[\mu + \Lambda z + \varepsilon] = \mu$ (since $E[z] = 0$, $E[\varepsilon] = 0$)

$$\mu_{zx} = E\begin{bmatrix} z \\ x \end{bmatrix} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix} \begin{matrix} \} d \\ \} n \end{matrix}$$

For covar matrix: $\Sigma = \begin{bmatrix} E(z - Ez)(z - Ez)^T & \dots \\ \dots & E(x - Ex)(z - Ez)^T \end{bmatrix}$

$\Sigma_{11} = \mathrm{Cov}(z) = I$

$\Sigma_{21} = E(x - Ex)(z - Ez)^T$

$= E[(\mu + \Lambda z + \varepsilon - \mu) \cdot z^T]$

$= E[\Lambda z z^T] - E[\varepsilon z]$ ($= 0$)

$= \Lambda E[z z^T] = \Lambda I$

$\Sigma_{22} = E[(\mu + \Lambda z + \varepsilon - \mu)(\mu + \Lambda z + \varepsilon - \mu)^T]$

$= \dots$

$= \Lambda\Lambda^T + \Psi$

Ng 13

So $\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$

Want to find parameters μ, Λ, Ψ

$P(x^{(i)})$ $\quad x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$

Write $\ell(\theta) = \sum \log P(x^{(i)})$

Take derivatives to max $\ell(\theta)$. But no analytic soln for MLE in this form.

∴ Use EM algorithm:

E-step: $Q_i(z^{(i)}) = P(z^{(i)} | x^{(i)}, \theta)$: $z \in \mathbb{R}$ so use dist^n

M-step: $\theta = \arg\max_\theta \sum_i \int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$

Ng 14

EM

29,622

Factor analysis

Principal Component Analysis (PCA)

FA cont: recision

$\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$

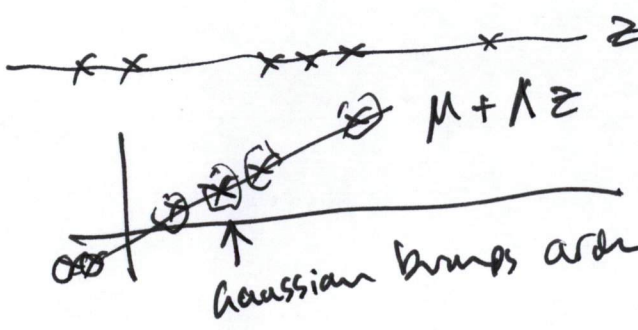
$z \sim \mathcal{N}(0, I)$, $z \in \mathbb{R}^d$: latent variable ≡

$x = \mu + \Lambda z + \varepsilon$: cond^{n} dist^{n} of x given z

$x|z \sim \mathcal{N}(\ldots)$

$\varepsilon \sim \mathcal{N}(0, \Psi)$

Parameters: $\mu \in \mathbb{R}^n$, $\Lambda \in \mathbb{R}^{nd}$ $\Psi \in \mathbb{R}^{nd}$ diagnol



$$\begin{pmatrix} z \\ x \end{pmatrix} \sim \mathcal{N}(\mu_{xz}, \Sigma)$$

$$\sim \mathcal{N}\left(\begin{pmatrix} \vec{0} \\ \mu \end{pmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix}\right)$$

What is x?

$P(x) = ?$

$x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$: marginal dist^{n} of x under my model

Given training set of labelled ex.

$$\prod_{i=1}^{m} P(x^{(i)}; \Lambda, \mu, \Psi) = \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2} |\Lambda\Lambda^T + \Psi|} \cdot \exp\left(-\frac{1}{2}(x-\mu)^2 (\Lambda^T + \Psi)^{-1}\right)$$

↑ likelihood of ϕ (params)

But can't solve for MLE: $\frac{\partial}{\partial \phi} = 0$ is unsolvable

∴ Use EM algo

EM algo

E step:

Find $Q_i(z^{(i)}) = p(z^{(i)} | x^{(i)}, \theta)$

M step:

$$\theta := \arg\max_{\theta} \sum_i \int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} d(z^{(i)})$$

So it turns out that to implement E step.

1. $z^{(i)}$ is a continuous value var;

Ng 14

E-step

$z^{(i)}|x^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}|x^{(i)}}, \Sigma_{z^{(i)}|x^{(i)}})$

where

$$\mu_{z^{(i)}|x^{(i)}} = \vec{0} + \Lambda^T(\Lambda\Lambda^T + \Sigma)^{-1}(x^{(i)} - \mu)$$

$$\Sigma_{z^{(i)}|x^{(i)}} = I - \Lambda^T(M^T + \Psi)^{-1}\Lambda$$

$Q_i(z^{(i)})$: compute vectors of $\mu_{z^{(i)}}$ & $\Sigma_{z^{(i)}}$

- they represent mean & covariance

M-step : approach

key trick for computing these integral:

$$\int_{z^{(i)}} Q_i(z^{(i)})\, z^{(i)}\, dz^{(i)}$$

(a) write out $\int_{z^{(i)}} \frac{1}{2\pi^{d/2}|\Sigma_{z^{(i)}|x^{(i)}}|} \exp\left(-\frac{1}{2}(z^{(i)} - \mu_{z^{(i)}|x^{(i)}})^T \Sigma_{\mu^{(i)}|z^{(i)}} (z^{(i)} - \mu_{z^{(i)}|x^{(i)}})\right) \cdot z^{(i)}\, dz^{(i)}$

or

(b) far simpler to observe that this is an expected value.

$$E_{z^{(i)}\sim Q_i}[z^{(i)}] = \mu_{z^{(i)}|x^{(i)}}$$

M step

$$E_{z^{(i)}\sim Q_i}\left[\log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}\right]$$

$$= E_{z^{(i)}\sim Q}\left[\log P(x^{(i)}|z^{(i)};\theta)\right] + E_{z^{(i)}\sim Q_i}\left[\log \frac{P(z^{(i)})}{Q_i(z^{(i)})}\right]$$

For FA, only this one depends on the parameters. So just maximise that one in the M-step.

$$\max \sum_{i=1}^{m} E_{z^{(i)}}\left[\log p(x^{(i)}|z^{(i)}, \Lambda, \Psi, \mu)\right] \quad \text{w.r.t } \Lambda$$

has Gaussian density $\because$ $\frac{1}{2\pi^{d/2}}$ exp.

$x^{(i)}|z^{(i)} \sim \mathcal{N}(\mu, \Lambda z, \Psi)$

$$\rightarrow \sum_{i=1}^{m} E\left[\text{constant} - \frac{1}{2}(x^{(i)} - \mu\Lambda z^{(i)})^T \Psi^{-1}(x^{(i)} - \mu\Lambda z^{(i)})\right]$$

If you Adiff

∇_Λ (expression above) $\stackrel{\text{set}}{=} 0$

Ng 14

$$\Lambda = \left(\sum_{i=1}^{m} (x^{(i)} - \mu) E z^{(i)T}\right) \left(\sum_{i=1}^{m} E[z^{(i)} z^{(i)T}]\right)^{-1} \quad *$$

Expectations are w.r.t $z^{(i)} \sim Q_i$

What are they?:

$$E[z^{(i)T}] = \mu_{z^{(i)}|x^{(i)}}^T$$

If $z \sim N(\mu, \Sigma)$, then $\Sigma = E z z^T - E z (E z)^T$

$$\Rightarrow E z z^T = \Sigma + (E z)(E z)^T$$

Substitute these into above: *

$$\Sigma_{z^{(i)}|x^{(i)}} = \mu_{z^{(i)}|x^{(i)}} \mu_{z^{(i)}|x^{(i)}}^T$$

Derivn of EM is more complicated than taking the expected values of the random var & subst into formula

I just wrote down a ton of math, & so, I stripped a lot of steps of the derivn: 0:33:44

This is probably the most maths I'll ever do in a lecture

00:39:00

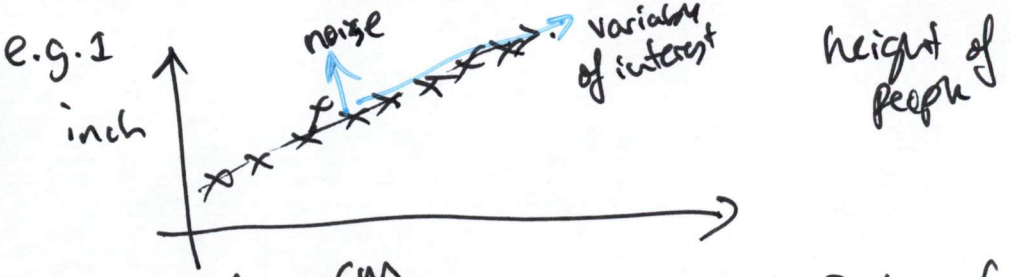
Principal component analysis algorithm (3)

unsupervised

Given $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$

Reduce it to k. dim dataset ($k \ll n$)

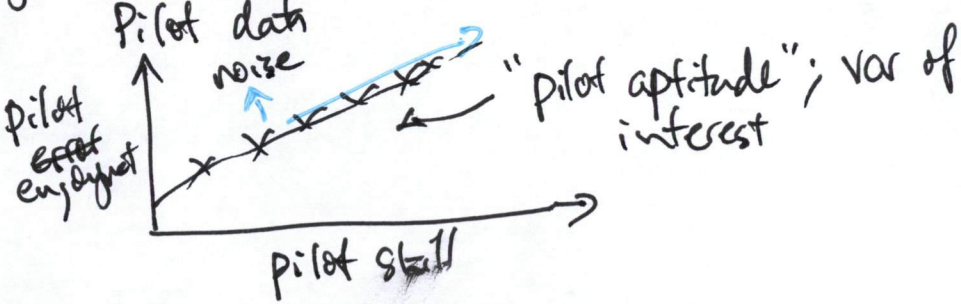
e.g. 1



can

rounding off, but close to straight: Reduce from 2D → 1D to get rid of noise.

e.g. 2 Work on autonomous helicopters



Here's the algorithm

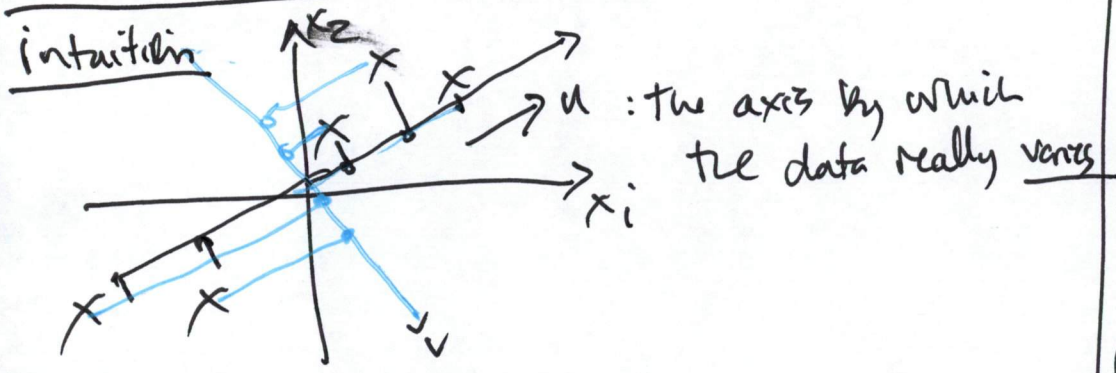
Pre-processing

1. Set $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$ → zero out mean
2. replace $x^{(i)} : x^{(i)} - \mu$
3. Set $\sigma_j = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)})^2$
4. replace $x_j^{(i)}$ with $x_j^{(i)} / \sigma_j$ → normalize to unit variance

Ng 14 PCA (cont)

Preproc deals ⊂ different scales
To find principal axes of variation:

intuition



u : the axis by which the data really varies

The projn of training set onto u has large variance:
Or v, the clustering is tighter:

Find the direc^n that allows greatest variance or spread: that a/c for the variance

If $\|u\| = 1$, ~~length of~~ $x^{(i)}$ projected onto u has length $= x^{(i)T} u$

Choose u:

$$\max_{u:\|u\|=1} \frac{1}{m} \sum_i ~~(x^{(i)T} u)^T~~ (x^{(i)T} u)^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (u^T x^{(i)})(x^{(i)T} u)$$

$$= u^T \left[\frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T} \right] u \qquad \text{s.t } \|u\| = 1$$

$\Rightarrow u$ must be principal eigenvector of $\Sigma = \frac{1}{2} \sum_{i=1}^{m} x^{(i)} x^{(i)T}$

$A u = \lambda u$
matrix vector ↑ ↑ eigenvector of A
eigenvalue of A

$\max u^T \Sigma u$ s.t $\|u\| = 1$; i.e $u^T u = 1$

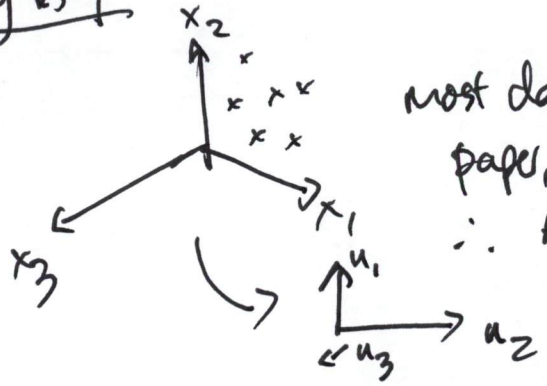
$\mathcal{L}(u, \lambda) = u^T \Sigma u - \lambda (u^T u - 1)$
↑ Lagrange multiplier

constraint optimiz^n:
$\nabla_u \mathcal{L} = \Sigma u - \lambda u$ set $= 0$

To summarize: to find principal axis, construct covar matrix Σ, find principal eigenvector. This is the best 1D subspace

For k ~~D subspace~~
choose $u_1, \ldots u_k$ to be k top eigenvectors of Σ

Ng 14 15



most data in the plane of the paper, but some lifting off
∴ A small u_3

Have $x^{(i)} \in \mathbb{R}^n$, new representation of data in $\{u_1 \dots u_k\}$ basis

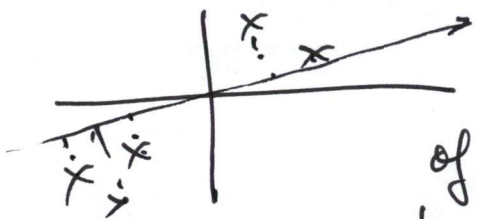
$y^{(i)} = (u_1^T x^{(i)}, u_2^T x^{(i)}, \dots, u_k^T (x^{(i)}))$ $y^{(i)} \in \mathbb{R}^k$

00:59:00

Q: "What do you mean by trivial eigenvectors?"

A: Some matrixes are deficient, but Σ is symmetric so never deficient:
Sometimes there is ambiguity in u_i. u_i can rotate freely in subspace. Think of u_i as basis to represent data; sometimes dangerous to look at them individually. Tiny numerical differences can change them. Look at the whole subspace.

Another interpretn of PCA



Think of it as min the sum of squares of distances from points to line.

There are 9-10 interpretns of PCA.

Applicns of PCA

(a) Visualizn
- hard to look at high dim dataset
- project into 2D/3D plot
- e.g. 50D monkey brain measurements in time series; bring down to 3D

(b) Compression
- don't want to deal c high D data

(c) Learning
-
- in practice, high D datasets are lying in a subspace;
- if ML takes a longtime, use PCA to compress

(5)

Ng

PCA applic^{n}s (cont)

High D, more features, → complex hypothesis class. Prone to overfitting. Use PCA to reduce. In industry, this is done too much.

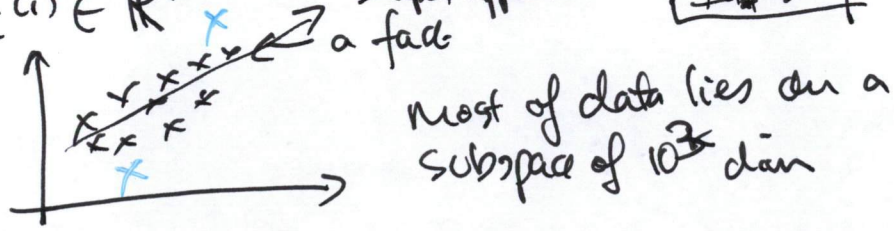
(d) Outlier / anomaly detection

- find subspace for data; then look for things that don't fit.
- not good for this

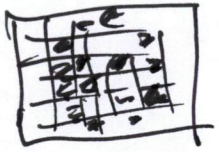
(e) Matching / distance calc^{n}s

e.g. face recognition 100×100

$x^{(i)} \in \mathbb{R}^{10^3}$ shape/appearance of a face



most of data lies on a subspace of 10^3 dim



Use PCA to select 50 D subspace that

Give you a face; find other faces that are similar:

Measure distance: project both faces onto subspace, & measure similarity

Given

$x^{(i)} \in \mathbb{R}^{10^3}$ [e]

$u^{(i)} \in \mathbb{R}^{10^3}$ [e] eigenfaces:

Look at diff eigenvectors to see shade, outline ...

Alex / Sandy Pentland MIT

PCA is useful to know. I see it used more often than it should be. Try to use the training data without compression

---

Lecture 15 — PCA — LSI

SVD implementation

ICA: related, but different

- end of unsupervised learning

Ng 15

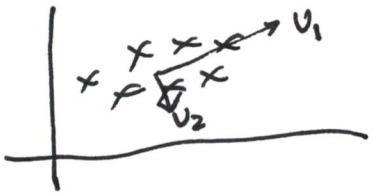
25,933

PCA

- Latent semantic indexing (LSI)
- Singular Value Decomposition

Independent component analysis

Review of PCA



Algorithm

1. Normalize to zero, unit variance
2. $\Sigma = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} x^{(i)T}$
3. Find top k eigenvectors of Σ

The difficulty of doing PCA, is that covar Σ becomes huge : $\Sigma \in \mathbb{R}^{n \times n}$

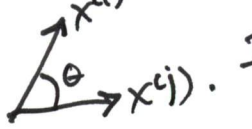
Application to text data : "LSI"

text as high dim vector $\begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}$ a, aardvark, ..., zip

In LSI, usually skip pre-processing step: don't normalize, It would scale up the weight of rare words.

Problem: how similar are two documents $x^{(i)}$, $x^{(j)}$

Sim $(x^{(i)}, x^{(j)})$?

One measure is 

. If θ is small, they are similar

$$\text{Sim}(x^{(i)}, x^{(j)}) = \cos\theta = \frac{x^{(i)T} x^{(j)}}{\|x^{(i)}\| \, \|x^{(j)}\|}$$

Here's the intuition behind LSI:

PCA will remove the noise vector.

Numerator of sim:

$$x^{(i)T} x^{(j)} = \sum_k x^{(i)}_k x^{(j)}_k$$

$$= \sum_k 1\{\text{Docum } i \text{ \& } j \text{ both contain word } k\}$$

e.g. $x^{(i)}$ → "study" $x^{(j)}$ → "learn"

: LSI recognizes similarity of "study" & "learn"

e.g. 100 d subspace

Implementation

$x^{(i)} \in \mathbb{R}^{50,000}$

$\therefore \Sigma \in \mathbb{R}^{50,000 \times 50,000}$

If $A \in \mathbb{R}^{m \times n}$

$$A = U D V^T$$

$(m \times n)\ (m \times n)\ (n \times n)\ (n \times n)$

(SVD) ← linear algebra decomposition, often used

$$D = \text{diagonal} = \begin{bmatrix} \sigma_1 & & 0 \\ & \sigma_2 & \\ & & \ddots \\ 0 & & \sigma_n \end{bmatrix}$$

σ_i = singular values of A

e.g. $\begin{bmatrix} A \\ m \times n \end{bmatrix} = \begin{bmatrix} U \\ m \times n \end{bmatrix} \begin{bmatrix} D \\ n \times n \end{bmatrix} \begin{bmatrix} V^T \\ n \times n \end{bmatrix}$

U's cols : eigenvectors of AA^T

V's cols : " " A^TA

Compute using "svd" in Matlab/Octave

svd is like a sqrt fn, it is very stable because it's well implemented

Ng 15

Nice thing: use svd to compute eigenvectors in PCA

$$\Sigma = \sum_{i=1}^{n} x^{(i)} x^{(i)T}$$

"Design matrix":

$$X = \begin{bmatrix} - & x^{(1)T} & - \\ - & x^{(2)T} & - \\ & \vdots & \\ - & x^{(m)T} & - \end{bmatrix}$$

$\Sigma = X^T X$: covariance matrix

$$= \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$$

To get top k eigenvectors of Σ:

$X = UDV^T$

top k columns of V are top k eigenvectors of $X^TX = \Sigma$

$X \in \mathbb{R}^{m \times 50{,}000}$

For 100 dim data, Σ can be computed directly

In practice:

Turns out if X is wide $\begin{bmatrix} X \\ m \times n \end{bmatrix} = \begin{bmatrix} U \\ m \times m \end{bmatrix} \begin{bmatrix} D \\ m \times n \end{bmatrix} \begin{bmatrix} \\ n \times n \end{bmatrix}$

$\begin{matrix} \\ \\ 0 \end{matrix}$ $\begin{matrix} \\ \\ 0 \end{matrix}$

- cutting off the 0 values, by convention that vary on how they cut things off.

CONTEXT

	model density $P(x)$	not probabalistic
subspace	Factor analysis	PCA
clumps / groups	mixture of Gaussians	k-means

Avoid using FA to reduce dim'

ICA

Find independent component not Principal

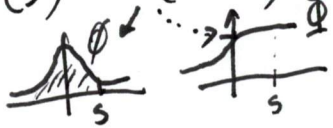
Shows slides of cocktail party problem, & audio files of mixed sound

Prop:

CDF - cumulative dist f^{n}

random variable s, density $P_s(s)$

CDF $F(s) = P(s \leq s)$



e.g. Φ cdf

$$F(s) = \int_{-\infty}^{\infty} P_s(t)\,dt$$

$s \rightarrow$ specifying $P_s(s)$ or specifying $F(s)$

Ng 15

$P_s(s) = F'(s)$

cdf has to be like, e.g

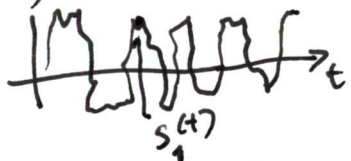
Just a sanity check:

~~Data from~~

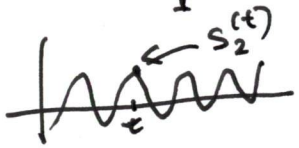
Original sources (speakers) (s)

$s \in \mathbb{R}^n$

$s_j^{(i)}$ = signal from speaker j at time i



speaker 1



speaker 2

We observe ~~3~~ microphones x:

$x^{(i)} = A s^{(i)}$ $\quad x^{(i)} \in \mathbb{R}^n$

(n microphones)

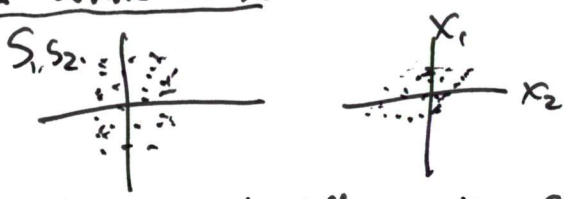
each recording overlapping combⁿs of what s_i are saying

ie $x_j^{(i)} = \sum_k A_{j,k} s_k^{(i)}$

Goal: find $W = A^{-1}$, so that

$s^{(i)} = W x^{(i)}$ $\quad W = \begin{bmatrix} - & w_1^T & - \\ - & w_2^T & - \\ & \vdots & \\ - & w_n^T & - \end{bmatrix}$

e.g $s_{1,2}$ for white noise:



2 Ambiguities: can't tell which speaker is which; permutation, " " the sign of the signal

...

$s_j^{(i)}$ – non-Gaussian

Suppose $s^{(i)} \sim \mathcal{N}(0, I)$

rotational symmetry prevents learning orientⁿ. ICA relies critically on non-Gaussian

Derivⁿ of algorithm

Let $s \in \mathbb{R}^n$: joint vector of speakers

density of $s = p_s(s)$

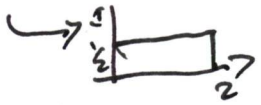
$x = As = W^{-1}s \qquad s = (Wx)$

$p_x(x) = p_s(Wx) \cdot |W|$ ← determinant of W

eg $p_s(s) = 1\{0 \le s \le 1\}$ $\quad (s \sim \mathcal{U}[0,1])$

$p_s(s)$ Let $x = 2s$, $A = 2$, $W = \frac{1}{2}$

$p_x(x) = 1\{0 \le x \le 2\} \cdot \frac{1}{2}$



Ng 15

ICA model

$p(s) = \prod_{i=1}^{n} p_s(s_i)$ (s_i is indep)

$p(x) = \left[\prod_{i=1}^{n} p_s(w_i^T x)\right] \cdot |W|$

$W = A^{-1} = \begin{bmatrix} \cdots w_1^T \cdots \\ \cdots w_n^T \cdots \end{bmatrix}$

$s_i = w_i^T x$

Choose $p_s(s) = ?$

- can't be Gaussian cdf, so convenient to use

sigmoid: $F(s) = \frac{1}{1+e^{-s}}$; happens to increase for $0-1$, & is not Gaussian

> Sigmoid has fatter tails, decays to zero slowly

> Also use $P(s) = \frac{1}{2} e^{-|s|}$: Laplace distn will also work fine.

Algo:

Given $\{x^{(1)} \ldots x^{(m)}\}$

$\ell(w) = \sum_i \log\left(\prod_j p_s(w_j^T x^{(i)})\right) \cdot |W|$

use Stochastic grad desc : $p_s(s) = F'(s)$

$w_i = w + \alpha \nabla_w \ell(w)$

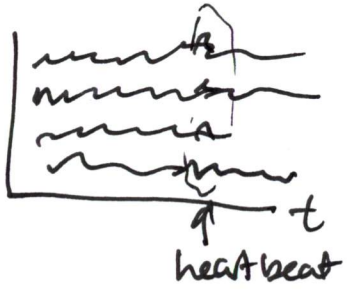
$\nabla_w \ell(w) = \begin{bmatrix} 1 - 2g(w_1^T x) \\ \vdots \\ 1 - 2g(w_n^T x) \end{bmatrix} \cdot (w^T)^{-1}$

$s^{(i)} = w x^{(i)}$ ← separated out data sources

Applicns

EEG cap

- records heart, eye blinks, etc summed.



- find IC of brain signal – heart beats, – eye blink
- remove artifactual components to obtain clean data

Aapo Hyvarinen

Image patches make up natural images: ICA Method of how brain processes images

Brief discussion of number of microphones & number of speakers

Ng 16 | Reinforcement Learning | 23, 506

- MDPs : Markov Decision Process
- Value function
- value iteration
- policy iteration

Context

For supervised learning, we had the right answer

For unsupervised " , we had the data

In between, some supervision : reinforcement.

Picture of autonomous helicopter from lecture 1 :
- the football fields at stanford, helicopter is outside upside down
- difficult aerobatic manoeuvre.

Learning problem : input 10 tuple/sec - pos^n of helicopter.
o/put control numbers

We don't know what the right action is - no training set for it. Instead, use reward signal as feedback on how well it is doing

e.g. 2 : chess or othello. - like training a dog

Harder than supervised learning, because no simple right/answer. You have to keep making decisions.:

Credit assignment problem : you don't know which moves were right, & which wrong.

Formalize this more

- robotics examples are easy to show;

MDP - Markov decision problem

$MDP(S, A, \{P_{sA}\}, \gamma, R)$ 5-tuple

S : set of states

A : set of actions

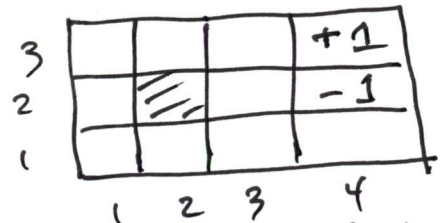
P_{sA} : state transition distn

$\sum_{s'} P_{sA}(s') = 1, \quad P_{sA}(s') \geq 0$

γ : discount factor $0 \leq \gamma < 1$

R : reward function $R : S \mapsto \mathbb{R}$

Ng 16 Example of an MDP



adapted from P. Norvig, textbook

- robot trying to reach top right (4,3)

11 states, $A = \{N, S, E, W\}$

Model of stochastic

State transition dist^n:

$P_{(3,1)}(N((3,2)) = 0.8$

$P_{(3,1)}(N((4,1)) = 0.1$

$P_{(3,1)}(N((2,1)) = 0.1$

$P_{(3,1)}(N(3,3)) = 0$

Reward function:

$R((4,3)) = +1$

$R((4,2)) = -1$

$R(s) = -0.02$ for all other states

The destin^n is the end of the problem: a 12th state; "a zero-cost absorbing state."

MDP works like this:

At state s_0

choose a_0

Get to $s_1 \sim P_{s_0, a_0}$

choose a_1

Get to $s_2 \sim P_{s_1, a_1}$

⋮

After a while, we have/can apply R to s_0...

$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$: total payoff

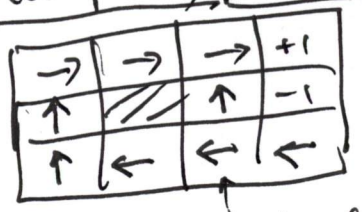
So γ diminishes the reward at each timestep. $\therefore 0 \leq \gamma < 1$. Wins in the future weigh less than near term.

Goal is to choose actions over time that max total payoff

$E[R(s_0) + \gamma R(s_1) + \gamma^2 (R(s_2) + \ldots]$

Policy $\pi : S \mapsto A$: for every state, what action to take

Optimal policy/strategy



why left here? To avoid veering into -1. Short but risky!

Ng 16

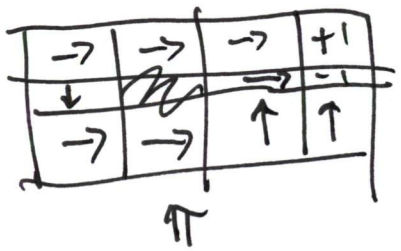
If turns out that MDPs are good at making subtle tradeoffs

Some definitions:

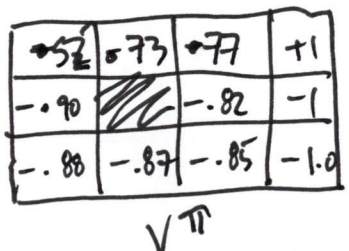
V^{π}, V^{*}, π^{*} ← optimal policy

for any π, define value function $V^{\pi} : S \mapsto \mathbb{R}$ s.t. $V^{\pi}(s)$ is expected total payoff starting in state s & execute π

$$V^{\pi}(s) = E\left[R(s_0) + \gamma R(s_1) + \cdots \mid \pi, s_0 = s\right]$$



π



V^{π}

$$V^{\pi}(s) = E\left[R(s_0) + \gamma\left(R(s_1) + R(s_2) + \cdots\right) \mid \pi, s_0 = s\right]$$

$s_0 \to s$, $s_1 \to s'$

$V^{\pi}(s_1)$ ← random state

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^{\pi}(s')$$

Bellman's equation — used in solving MDP

$P_{s\pi(s)}(s')$ ← probability of getting to s' from s using action $\pi(s)$

(3)

Let's try & say this one more time :... he goes thru it again

~~$V^{\pi}(s) = R(s) + \gamma$~~

Bellman's eqn gives a way to solve for V^{π} in closed form. It imposes linear constraints on the value fn. The linear system of eqns can be solved for the value fn.

e.g. $V^{\pi}((3,1)) = R((3,1)) + \gamma[0.8\, V^{\pi}((3,2)) + 0.1\, V^{\pi}((4,1)) + 0.1\, V^{\pi}((2,1))]$ $\quad \pi((3,1)) = N$

write eqn for each state in MDP.

∴ there are 11 variables, 11 eqns;

Optimal value function

$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

Bellmans eqn for $V^{*}(s)$ rather than $V^{\pi}(s)$

$$V^{*}(s) = R(s) + \max_{a} \gamma \sum_{s'} P_{sa}(s') V^{*}(s')$$

↑ optimal expected payoff

↑ max over all actions a

Ng 16

$$\pi^*(s) = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s)$$

Algorithm to compute π^*

If I can compute V^*, then can get π^*, the optimal policy. But $V^*(s)$ is not nice. Hard to compute - huge possible number of policies.

To ground the notation:

"Value iteration" algorithm

initialize $V(s) = 0 \quad \forall s$ ← array of elements

For every s, update:

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

This will make $V(s) \to V^*(s)$: converge

2 ways of doing update:

(a) Synchronous update $V := B(V)$ ← Bellman-Back operator
- all states at once

(b) asynchronous update :
- $V_1(s)$, $V_2(s)$, $V_3(s)$,

Run this algorithm on MDP

.86	.90	.93	+1
.82		.69	-1
.78	.75	.71	.49

Plug these numbers in to get π^*
- e.g. expected total payoff is higher if you go west rather than north.

Other standard algorithm: Policy iteration

- initialize π randomly.
- repeat:
  - Let $V := V^\pi$ (solve Bellman's eqns)
  - Let $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

then $V \to V^*$, $\pi \to \pi^*$

The computationally expensive step is solving the linear eqns in Bellman's. e.g. fine for ≤ 1000 states. But for millions of states, many eqns. So use value iteration.

What if you don't know P_{sa}?

e.g. for helicopter:

5-tuple $(S, A, \{P_{sa}\}, \gamma, R)$

- S, A: you know
- $\{P_{sa}\}$: usually estimate from data
- γ: you choose
- R: you come up with one

Ng 16

$$P_{sa}(s') = \frac{\#\text{ times took action "a" in } s \text{ got to } s'}{\#\text{ times took action "a" in } s}$$

$$\left(\text{or } \frac{1}{|S|} \text{ if } \frac{0}{0}\right)$$

So, in reinforcement learning

- more of a mix-match way of doing things; not one right way.

Putting it together:

Repeat {

- Take action using π to get experience in MDP
- Update estimates of P_{sa}
- Solve Bellman's eq^{n}s using value iteration to get V^*
- Update $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

"Aren't you excited? Our first MDP algorithm"

1:12:50