- 模型记号. 学习的系统 概率分布或决策函数.

  $F = \{f \mid Y = f(X)\}$

- 决策函数. Loss Fun
  - 0-1 Lf

    $L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases}$
  - 平方 quadratic Lf

    $L(Y, f(X)) = (Y - f(X))^2$
  - 绝对 abs. Lute Lf

    $L(Y, f(X)) = |Y - f(X)|$
  - logarithmic Lf

    ~~$L(Y, P(f(X)))$~~

    $L(Y, P(Y|X)) = -\log P(Y|X)$

- 为防止过拟合 over-fitting 加 $\boxed{\text{Regularization}}$ 的 penalty item
  Regularizer
  - 交叉验证. cross validation

-

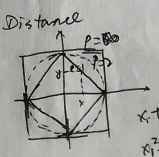
		predict P	predict N
Real	T	TP	FN
	F	FP	TN

guess

$\text{Recall-Rate} = \frac{TP}{\text{Real True}} = \frac{TP}{TP+FN}$

$\text{precious} = \frac{TP}{P} = \frac{TP}{TP+FP}$

- Distance



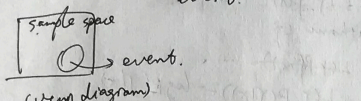
$x+y=1 \quad (y=1-x)$ Manhattan Dist..

$x_i^2+y_i^2=1^2$ Euclidean Dist..

$\sqrt[\infty]{x_i^\infty + y_i^\infty} = \max(x_i, y_i) = 1$

- Random variable / aleator / stochastic
  : a variable whose value depend on outcome of a random phenomenon
- The domain of a ~~ran~~ random variable is $\boxed{\text{SAMPLE SPACE}}$ sample space : the set of possible outcomes of a random phenomenon.
- endowed with a probability distribution
  - probability mass function
  - Probability ~~distribution~~ fun. density
- EVENT: a set of outcomes of an $\boxed{\text{experiment}}$ (a subset of the sample space) to which a probability is assigned.

  Typically. Sample Space is finite $\Rightarrow$ any subset of the sample space is an event.



(venn diagram)

- experiment / ~~Tria~~ Trial
  Any procedure that can be infinitely repeated and has a well-defined set of possible outcomes, known as the sample space.

$P(A)P(B) \quad P(A)P(\bar{B})$

$P(AB) + P(A\bar{B}) = P(AB \cup A\bar{B}) = P(A)$

$P(A\bar{B}) = P(A) - P(AB) = P(A)(1 - P(B)) = P(A)P(\bar{B})$ ✓

pmf.

- Bernoulli (Bernoulli) trial coin flip

  试 A random experiment has exactly 2 outcomes ~~互~~ mutually ~~exclusive~~ exclusive

  single
- Binomial $P(C_1C_2C_3 \cdots C_n) = P(C_1)P(C_2)P(C_3) \cdots P(C_n)$

  $P(x=k) = \binom{n}{k} p^k q^{n-k} \quad k=0,1,2\ldots$

  (伯努利试验)
- Poisson distribution (frequency of buses)

  泊松分布 $\lambda > 0$, assume $np_n = \lambda$

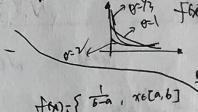
  $P(x=k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad k=0,1,2\ldots$

  $\lim_{n\to\infty} \binom{n}{k} p_n^k (1-p_n)^{n-k} = \frac{\lambda^k e^{-\lambda}}{k!}$

  $E = Var = \lambda$

pdf

$X \sim \text{exponential}(\lambda)$ Waiting time between buses



$f(x) = \begin{cases} \frac{1}{\theta} e^{-\frac{x}{\theta}}, & x > 0 \\ 0, & \text{otherwise} \end{cases}$

$X \sim \text{Normal}(\mu, \sigma^2)$ Height

$X \sim \text{uniform}(a, b)$ random $f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a,b] \\ 0, & \text{elsewhere} \end{cases}$

$\text{Exp}(\lambda)$. $f(x) = \lambda e^{-\lambda x}$ $\theta = \frac{1}{\lambda}$ default $\lambda = 1$ $Var = \frac{1}{\lambda^2}$

- chi-squared distribution distribution of a sum of the squares of k independent standard normal random variables.

  $X_1, X_2 \cdots X_n$ 来自 $N(0,1)$ 独立.

  则. $\chi^2 = X_1^2 + X_2^2 + \cdots X_n^2$

  $\sim \chi^2(n)$

  freedom degree

- $X \sim N(0,1)$, $Y \sim \chi^2(n)$ mutual independent.

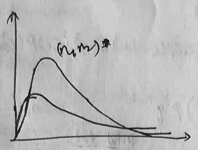
  $t = \frac{X}{\sqrt{Y/n}} \sim t(n)$

  Student's t distribution is any member of a family of ~~distribution~~ continuous probability distributions that arises when estimating the mean of a normally distributed population when sample size is small

· F distribution

$U \sim \chi^2(n_1)$ $V \sim \chi^2(n_2)$ mutually independent.

$$F = \frac{U/n_1}{V/n_2} \sim F(n_1, n_2)$$



Linear Regression

IID independent and identically distributed.

The sum of IID also Gaussian Distribution.

$$N(\Sigma\mu, \Sigma\sigma^2)$$

scale. $N(w\mu, (w\sigma)^2)$

$y_i = f(x) + \varepsilon$

$\varepsilon \sim N(\mu, \sigma^2)$

P

ML-III

Gradient Decent.

Sum of squared residual $= \Sigma(\text{observed} - \underline{\text{predicted}})^2$

(intercept + $w^T x_i$)

$\frac{d\,SSE}{d\,\text{intercept}}$ = slope

step

derivative $SSE = \Sigma(y_i - c - wx_i)^2$

$$\frac{dSSE}{dc} = \Sigma[2(y_i - c - wx_i)] \cdot (-1)$$

$$= -\Sigma[2(y_i - c - wx_i)]$$

Stochastic Approximation
- batch
- single point
- mini batch

$X_i \xrightarrow{ML} \boxed{y_i}$ only output

predicted guess_i ERROR

$$\text{Cost} = \sum(\underbrace{y_i - \text{guess}_i}_{\text{error}})^2 = \sum(y_i - mx - b)^2$$

to make it decrease we need to

$m = m + \Delta m$

$b = b + \Delta b$

$$\frac{\partial \text{Cost}}{\partial m} = \sum 2 \cdot \text{Error} \frac{\partial \text{Error}}{\partial m}$$

What change would be caused if m changed at 1

And to make it efficient. Δm should be at the fastest speed, that's $\frac{\partial \text{cost}}{\partial m} \to 0$

$$\frac{\partial(y_i - mx - b)}{\partial m} = -x$$

$$= \sum 2 \cdot \text{Error} \cdot (-x)$$

because the constants could be realized in the learning rate. so the crucial part is:

$(-1) \cdot -\text{Error} \cdot x$ * learning rate

opposite direction

That's the direction to go in which we could get cost increased.

step big or small?

$\therefore$ $\Delta m = (\Sigma + \text{Error} \cdot x) \cdot \text{learning rate}$ $\text{Error} = (y_i - \text{guess})$

$\Delta b = (\Sigma + \text{Error} \cdot 1) \cdot \text{learning rate}$

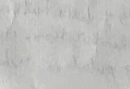
$$\Delta T = \frac{\partial T}{\partial x} + \frac{\partial T}{\partial y} + \frac{\partial T}{\partial z} = \nabla T \cdot (\Delta x, \Delta y, \Delta z)$$

$$= |\Delta T| \cdot |\Delta R| \cdot \cos\theta$$

lively
sweet
supercute
pretty
beautiful
charming
fascinating
endearing
stunning
shockingly
fetching
appealing
enchanting

两向量叉积(向量积) $\vec{a} \times \vec{b}$ 的结果 法向量 (右手法则)

$$\vec{n} = \vec{a} \times \vec{b} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$



例如 $\vec{a} = (-3, 4, -6)$ $\vec{b} = (-2, 3, -1)$

$$\vec{n} = \begin{vmatrix} i & j & k \\ -3 & 4 & -6 \\ -2 & 3 & -1 \end{vmatrix} = 14i + 9j - k$$

该平面(面)方程为 $f(x,y,z)$: $14(x - x_0) + 9(y - y_0) - (z - z_0) = 0$

$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) \Leftarrow (14, 9, -1)$ 是的法向量

方向导数: 沿任意方向

$f(x,y)$ 沿 $\vec{t}$ 方向的方向导数为

沿 m 是任意的

$$\frac{\partial f(x,y)}{\partial \vec{t}} = f_x(x_0, y_0)\cos\alpha + f_y(x_0, y_0)\cos\beta$$

α, β 是向量 $\vec{t}$ 与 x, y 轴的夹角

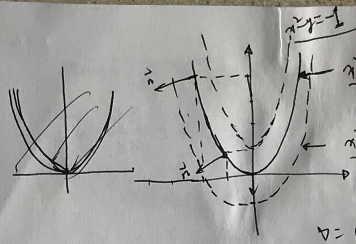
$= [f_x(x_0, y_0), f_y(x_0, y_0)]\,[\cos\alpha, \cos\beta]$

$= \text{grad}\, f(x_0, y_0) \cdot e_t$ → e 单位向量

$= |\text{grad}\, f(x_0, y_0)| \cos\theta$ θ 为梯度与 l 的夹角

梯度: 每一点都有一个量 $f_x(x_0, y_0)\vec{i} + f_y(x_0, y_0)\vec{j}$ 称为梯度 $\text{grad}\, f(x_0, y_0)$

ML-IV



$\nabla = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2x, -1)$

$\nabla(-1, 1) = (-2, -1)$

$\nabla(-2, 4) = (-4, -1)$

$\nabla(0, 0) = (0, -1)$

$\therefore$ grad(f) 的方向为 f(x,y) 增加最快的方向

Linear Regression

IIP

independent. identically. distributed

$$P(x, y, z) = P(x) \cdot P(y) \cdot P(z)$$

$$P(\vec{y}) = \prod_{i=1} P(y_i)$$

$y_i = mx_i + C + \varepsilon_i$

$\varepsilon_i \sim N(0, \sigma^2)$

$(y_i - mx_i - C) \sim N(0, \sigma^2)$

$$P(y_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - f(x_i))^2}{2\sigma^2}\right)$$

Likelihood

That means the probability if not obtain such points is $\prod P(y_i)$

$$P(\vec{y}) = \prod P(y_i) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{\Sigma(y_i - f(x_i))^2}{2\sigma^2}\right)$$

$$L = \underset{\text{ln}}{\log} P(\vec{y}) = -\frac{n}{2}\log(2\pi\sigma^2) - \sum\frac{(y_i - f(x_i))^2}{2\sigma^2}$$

to max(L)

$$\Rightarrow \min(-L) = \frac{n}{2}\log 2\pi + \frac{n}{2}\log\sigma^2 + \frac{\Sigma(\text{Error})^2}{2\sigma^2}$$

$$E = \log\sigma + \frac{\Sigma(\text{Error})^2}{2\sigma^2}$$

object function

Error $= y_i - mx_i - C$

$\min(E)$, ~~= min f~~

then. $\frac{\partial E}{\partial m} = 0$,

$-2\sum x_i(y_i - mx_i - c) = 0$

$m = \frac{\sum (y_i - c)x_i}{\sum x_i^2}$

(2) $\frac{\partial E}{\partial c} = 0$,

$-2\sum (y_i - mx_i - c) = 0$

$c = \frac{\sum (y_i - mx_i)}{n}$

"OLS" vs "MLE"?

Fit	linear regression	statistical model with parameters unknown
Objective	vertical error $\sum (y_i - f(x_i))^2$	$\prod P(x_i, y_i)$ μ, σ^2 statistical parameters

coordinate descent.

gradient descent: compute the gradient at a position and update

coordinate descent: update a parameter at a time to its minimum.

$\frac{\partial E}{\partial c} = 0$

$y = mx + c$

$E(m,c) = \sum (y - mx - c)^2$

$\frac{dE}{dm} = -2\sum (\text{error} \cdot x) = 0$

$-2(y - mx - c)x = 0$

$m = \frac{\sum (y-c)x}{\sum x^2}$

$\frac{dE}{dc} = -2\sum \text{error} = 0$

$\sum (y - mx - c) = 0$

$c = \frac{\sum (y - mx)}{n}$

That's the difference with steepest descent, in gradient descent, each step update is with gradient * learning while here, directly update is to the minimum parameter

$\beta = c^T$

$(AC)^T = CA^T$

$\frac{dAx}{dx} = A^T$

$\frac{dx^TAx}{dx} = (A+A^T)x$

$\frac{dx^TAx}{dA} = xx^T$

$(A^T)^{-1} = (A^{-1})^T$ ~ denoted A^{-T}

$(AB)^T = B^TA^T$

$(AB)^{-1} = B^{-1}A^{-1}$

$(\lambda A)^{-1} = \frac{1}{\lambda}A^{-1}$

$(\lambda A)^T = \lambda A^T$

$(AB)C = A(BC)$

$\frac{d\vec{z}^T\vec{c}\vec{z}}{dz} = \frac{d(c^Tz)^T}{dz}z + (c^Tz)^T\frac{dz}{dz}$

$= \frac{d\,z^Tc}{dz}z + z^T \cdot c$

(inner product)

$\frac{dx^2}{dx} = \frac{d(x \cdot x)}{dx} = \frac{dx}{dx}x +$

$\frac{dx^3}{dx} = \frac{d(x^2 \cdot x)}{dx} = \frac{d(x^2)}{dx} \cdot x + \frac{x^2 dx}{dx}$

$= x \cdot x + x^2$

$= 3x^2$

$(fg)' = f'g + g'f$.

$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$

$\left(f \cdot \frac{1}{g}\right)' = f'\frac{1}{g} + f\left(\frac{1}{g}\right)'$

$= \frac{f'}{g} - f\frac{g'}{g^2}$

derivative of $y = \vec{A}\vec{x}$ with respect to column vector $\vec{x}$. is

$\frac{\partial y}{\partial \vec{x}} = A^T$

$\vec{A}\vec{x} = \sum_i a_i x_i$ (1,n) (n,1)

$\frac{d}{dx_k} = a_k \Rightarrow \frac{d}{d\vec{x}} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = A^T$

Square matrix $\vec{A}$, x column vector $\vec{x}$ $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $\begin{bmatrix} \partial y/\partial x_1 \\ \partial y/\partial x_2 \\ \vdots \\ \partial y/\partial x_n \end{bmatrix}$

$\frac{\partial x^TAx}{\partial x} = (A+A^T)x$

$x^TAx = \sum_i \sum_j x_i x_j a_{ij}$

$\frac{d}{dx_k} = \sum_j a_{kj}x_j + \sum_i a_{ik}x_i$

$\therefore \frac{dx^TAx}{dx} = (A+A^T)x$ (n,1)

$= x^T(A+A^T)$? (1,n)

with respect to A. then

$\frac{\partial x^TAx}{\partial a_{ij}} = x_i x_j$

$\therefore \frac{\partial x^TAx}{\partial A} = xx^T$

The comparison of Batch Gradient Descent Algorithm

mini batch ——

stochastic ——

MAIN Difference is that how much samples are the gradients calculated.

$\frac{\partial J(w)}{\partial w} = \nabla = \frac{\partial \sum^{Ds} (y_i - \text{guess})^2}{\partial w}$, $\Rightarrow w = w - \nabla$

objective function

opposite direction

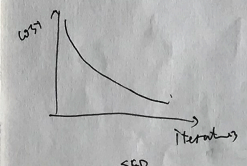
Batch GD	mini batch GD	Stochastic
for i in range(iterations):	for i in range(iterations):	
$\nabla = \frac{\partial \sum^{Ds} J}{\partial w}$	np.random.shuffle(Ds)	
$w = w - \eta\nabla$	for b in random_minibatch(Ds, batchsize=x):	batchsize = 1
	$\nabla = \frac{\partial \sum^{b} J}{\partial w}$	
	$w = w - \eta\nabla$	

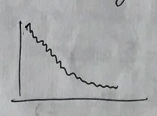
Advantages:

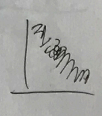
- batchsize < Ds.size ⇒ better generalization error
- randomly batched → avoid redundant examples
- faster than Batch GD

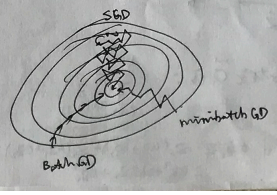
disadvantages:

- won't converge:

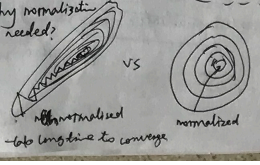








why normalization is needed?



too long time to converge

here lost font page VI about vectors

The comparison of Batch Gradient Descent Algorithm
mini batch —
Stochastic —

MAIN Difference is that how much samples are the gradients calculated.

$$\frac{\partial \overset{\text{objective function}}{\text{Cost}}\, J(w)}{\partial w} = \nabla = \frac{\partial \sum^{Ds} (y_i - \text{guess})^2}{\partial w}, \Rightarrow w = w - \nabla$$

opposite direction

Batch GD

for i in range(iterations):
$\nabla = \frac{\partial \sum J}{\partial w}$
$w = w - \eta \nabla$

mini batch GD

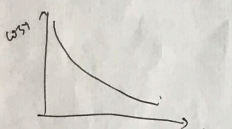
for i in range(iterations):
np.random.shuffle(Ds)
for b in random_minibatch(Ds, batchsize=x):
$\nabla = \frac{\partial \sum^{b} J}{\partial w}$
$w = w - \eta \nabla$

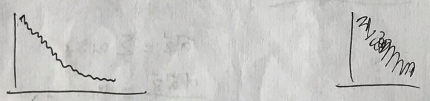
Stochastic

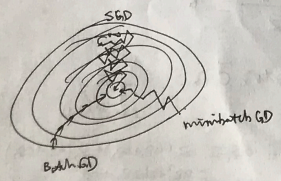
batchsize = 1

Advantages:
- batchsize < Ds size ⇒ better generalization error
- randomly batched → avoid redundant examples
- faster than Batch GD

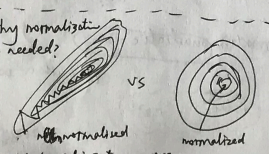
disadvantages:
- won't converge:







why normalization is needed?



→ takes longtime to converge

- dot product.

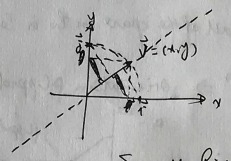
$$\begin{bmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underline{\text{scalar}}$$

in a number line

isn't it tranforming a vector to a line?

EXACTLY!

Simply, let us ~~transfor~~ linear transform a unit vector (x, y) to a number line



So we have to know the transform $\begin{vmatrix} \vec{i} & \vec{j} \\ ? & ? \end{vmatrix}$

And coincidently just transform it to the line which is in the same direction ~~with~~ of $\vec{v}$.

So the ~~linear~~ transform $\begin{vmatrix} \vec{i} & \vec{j} \\ ? & ? \end{vmatrix}$ is $\begin{vmatrix} \vec{i} & \vec{j} \\ \text{where } \vec{i} \text{ lands on the line} & \text{where } \vec{j} \text{ lands on the number line} \end{vmatrix}$

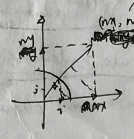
$$= \begin{vmatrix} \vec{i} & \vec{j} \\ x & y \end{vmatrix}.$$

$$\Rightarrow \begin{bmatrix} \vec{i} & \vec{j} \\ x & y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x^2 + y^2 \;!!$$

amazingly same as dot product!

because ~~we~~ $\vec{v}$ is given as an unit vector.
$x = 1\vec{i}$, $y = 1\vec{j}$

And when a non-unit vector given, it's the same.

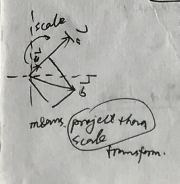


$\vec{u} = (nx, ny)$
$(\vec{u} = n\vec{v})$

$$\begin{bmatrix} \vec{i} & \vec{j} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} nx \\ ny \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{i} & \vec{j} \\ x & y \end{bmatrix} \begin{bmatrix} nx \\ ny \end{bmatrix} = n^2x^2 + n^2y^2$$

linear transformation in the line (u)

$= n(x^2 + y^2)$



arbitrary (x, y) vector is also could ~~under~~ taken as the linear transform in the exact ~~the~~ number line (v) of any x-y coordinate vector

27 Oct

ML — VII

- **rank** is the SPACE DIMENSION of the column vector
  and column R = row R.

- Determinant is the scalar value of the Matrix ~~tran~~ transformation
  ↓
  Area scalar

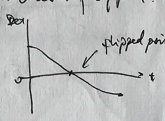
det(A)

so when det(A) = 0 means its dimensions lost

"it squishes all of the space on to a low dimension space."

when det(A) < 0 ⇒ Orientation flipped.

$\det(AB) = \det(A)\det(B)$



- So what is the meaning of a m×n matrix (m≠n)

① $\begin{vmatrix} \vec{i} & \vec{j} \\ a_1 & a_2 \\ b_1 & b_2 \\ c_1 & c_2 \end{vmatrix}$ shape: 3×2

2 basis vector
↘ expressed use a 3-dimension coordinate (landing spots)
It's still a transformed on plane in the n-dimension space.

② $\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & a_1 & a_2 \\ b & b_1 & b_2 \end{vmatrix}$ shape: 2×3

3 basis vector.
↘ expressed only use 2-dimension coordinate (landing spot).
⇒ $\vec{i}, \vec{j}, \vec{k}$ all land on a same plane!

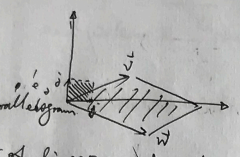
That is PCA: high dimensional basis were projected to a low dimension space

in term of the 2-D basis, a vector actually could be a more diverse and colorful things. That what they can't see. just like our human, only in 3-D creature. time is a one-way dimension. So called schrodinger's cat uncertainty principle, that are maybe just squeezed result from high dimension universe. So sad, for human species. Silly people.

$\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & a_1 & a_2 \end{vmatrix}$ ⇒ land on a number Line!

• cross product / not ~~out product~~ ≠ tensor product ≠ AA^T

$\vec{v} \times \vec{w} = \det([\vec{v}, \vec{w}])$

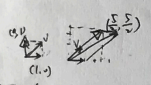


area of the parallelogram

also the scalar of linear transformation.

$= \det\begin{bmatrix}\vec{x}\\ \vec{y}\\ \vec{z}\end{bmatrix}([\vec{v}\times\vec{w}]) = \det\begin{bmatrix} x & v_1 & w_1 \\ y & v_2 & w_2 \\ z & v_3 & w_3\end{bmatrix}$

• basis transform

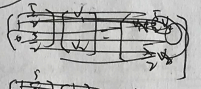


original basis $\begin{bmatrix}1&0\\0&1\end{bmatrix}$

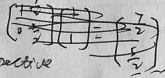
Changed basis $\begin{bmatrix}1&\frac{5}{2}\\0&\frac{5}{2}\end{bmatrix}$ A

$A^{-1} = \begin{bmatrix}1&-1\\0&\frac{2}{5}\end{bmatrix}$

a vector $\vec{v}\begin{bmatrix}v_1\\v_2\end{bmatrix} \Rightarrow$



$\vec{v} = \begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}0\\\frac{2}{5}\end{bmatrix}$



new perspective

$\underbrace{\begin{bmatrix}1&-1\\0&\frac{2}{5}\end{bmatrix}}_{A^{-1}}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\\frac{2}{5}\end{bmatrix}$

original coordinates

transformed basis coordinates

$I\vec{v} = A\,\text{new}\vec{v}$

$\text{new}\vec{v} = A^{-1}\vec{v}$

$\begin{bmatrix}1&-1\\0&\frac{2}{5}\end{bmatrix}\begin{bmatrix}\frac{7}{2}\\\frac{5}{2}\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$

~~A basis = new basis~~

The "same" position the plane with 2 coordinates

$A\vec{v} = \text{new}\vec{v}$

$\vec{v} = A^{-1}A\vec{v} = A^{-1}\text{new}\vec{v}$

basis $\vec{v}$ = new basis new$\vec{v}$ ⇒ $\vec{v}$ = new basis new$\vec{v}$

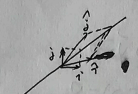
• eigen vector

unaltered matrix

$\underbrace{A}_{\text{transform matrix}}\vec{v} = \lambda\vec{v}$ → scalar

$(A-\lambda I)\vec{v} = 0$

$\Leftarrow \underbrace{\det(A-\lambda I) = 0}_{\text{dimension lost}}$



basis ⇒ basis

$\begin{bmatrix}\hat{i}\\\hat{j}\end{bmatrix}$ $\begin{bmatrix}\hat{i}\\\hat{j}\end{bmatrix}$

number

this line's orientation didn't change.

only scaled λ

$\vec{v}$: eigen vector.

• A 2-D transformation doesn't have to have such a eigenvector like.

everything have changed

$\det(A-\lambda I) = 0$

unaltered basis $\begin{bmatrix}1&0\\0&1\end{bmatrix}$

new basis

$\det\left(\begin{bmatrix}+1&1\\-1&1\end{bmatrix} - \lambda I\right) = \begin{bmatrix}1-\lambda&1\\-1&1-\lambda\end{bmatrix} = \underbrace{(1-\lambda)^2+1 = 0}_{\text{which is impossible in } \mathbb{R}}$

complex $\begin{bmatrix}\text{imaginary}&+i\\&-i\end{bmatrix}$

• eigen ~~vector~~ vector could be more than one! think why.

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

$A\vec{v} = \lambda\vec{v}$

$(A-\lambda I)\vec{v} = 0$

$\det(A-\lambda I) = (1-\lambda)(1-\lambda) = 0$

$\lambda = 1$

$\vec{v} = \begin{bmatrix}1\\0\end{bmatrix}$ — eigen vector

$A = \begin{bmatrix}2&0\\0&0.5\end{bmatrix}$

$(A-\lambda I)\vec{v} = 0$

$(2-\lambda)(0.5-\lambda) = 0$

$\lambda = 2$ or $\lambda = 0.5$

$\vec{v} = \begin{pmatrix}1\\0\end{pmatrix}$ $\begin{pmatrix}0\\1\end{pmatrix}$

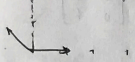
$\boxed{[\text{change of basis}]^{-1} A \underbrace{[\text{change of basis}]}_{\text{eigenbasis}} = A'}$

the matrix which is from the new basis space, but represent the same transformation

$\underbrace{\begin{bmatrix}1&-1\\0&1\end{bmatrix}^{-1}\begin{bmatrix}3&1\\0&2\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}1&-1\\0&1\end{bmatrix}}_{A'} = \begin{bmatrix}3&0\\0&2\end{bmatrix}$

And A' is guaranteed to be a diagonal one.

So the eigen vectors can be used to diagonalise the matrix.



$\begin{vmatrix}3-\lambda&1\\0&2-\lambda\end{vmatrix} = 0$

$\lambda = 3, 2$

$\begin{vmatrix}0&1\\0&-1\end{vmatrix}\begin{pmatrix}x\\y\end{pmatrix} = 0$ $\begin{vmatrix}1&1\\0&0\end{vmatrix}\begin{pmatrix}x\\y\end{pmatrix} = 0$

$y = 0$ $x + y = 0$

$\vec{v} = \begin{bmatrix}1\\0\end{bmatrix}$ $\vec{v} = \begin{bmatrix}1\\-1\end{bmatrix}$ or $\begin{bmatrix}-1\\1\end{bmatrix}$

§ weak LLN

$\lim_{n\to\infty} P(|\bar{X}_n - \mu| > \varepsilon) = 0$

or $\bar{X}_n \xrightarrow{P} \mu$ when $n \to \infty$

"WEAK" means in probability that still might get a $\bar{x}$ beyond ε, but just unlikely.

"unlikely"

For specific large n. $\bar{X}_n$ likely to be near μ. $|\bar{X}_n - \mu| > \varepsilon$ happens an infinite number of times although at a unlikely probability ($\to 0$)

strong LLN.

$P(\lim_{n\to\infty}\bar{X}_n = \mu) = P(\lim_{n\to\infty}|\bar{X}_n - \mu| > 0) = 1$

"strong" → surely

"surely"

$|\bar{X}_n - \mu| < \varepsilon$ for all large enough n.

With specified n, $|\bar{X}_n - \mu| > \varepsilon$ never occur.

§ gambler's fallacy / Monte Carlo fallacy

1913 black ball occurred 26 times in a row. $P = \left(\frac{18}{37}\right)^{26} = \frac{1}{66.6\text{ million}}$

5 times ~~tai~~ head. $P = \left(\frac{1}{2}\right)^5$

while when 4 times tails occured in advance. then the what about the fifth toss?

event "5 heads in a row"

and "4 heads first, then a tails" } equally $= \frac{1}{32}$

$P(A_5 \text{ tail} \mid \underbrace{A_1 \cap A_2 \cap A_3 \cap A_4}_{\text{heads}}) = P(A_5 \cap A_4 \cap A_3 \cap A_2 \cap A_1) = \frac{1}{32}$

BUT this is the case only before the first coin is tossed.

The basis of the fallacy is "the previous four tosses heads, with a run of luck in the past influencing the odds in the future"

example: die. (fair die) → 1 → win; 2, 3, 4, 5, 6 } lose

P{ 1 occurring at least once in 6 rolls }

$= 1-(\frac{5}{6})^6 =$

After one loss has occured in the first roll $(\frac{5}{6})$. According to the gambler's fallacy, they should have a higher chance of winning in the remaing 5 rolls. which should be $1-(\frac{5}{6})^6 =$

however. the probability ~~of at~~ is: $1-(\frac{5}{6})^5 =$ of at least once win in the next 5 rolls

the Pr indeed decreases, because there're fewer ~~trails~~ trials left in which to win (you had 6 times. now you only got 5)

§ Martingale betting strategy: (18th. France)

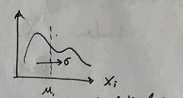
double the bet after every loss, so a eventually win would recover all the previous losses plus and win a equal profit equal to the original stake

exponential growth of the bets would eventually bankrupt "unlucky" gamblers. who choose that tactic.

§ CLT CENTRAL LIMIT THEOREM

ML-X-C

$X \sim$ IID
- independent.
- identical
- Distribution $?(\mu, \sigma^2)$



population distribution

then

$Sum(X_i) = \sum^n X_i \overset{n\to\infty}{\sim} N(n\mu, n\sigma^2)$ (iid)

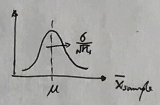
⇓

$\bar{X} = \frac{\sum^n X_i}{n} \overset{n\to\infty}{\sim} N(\mu, \frac{n\sigma^2}{n^2})$

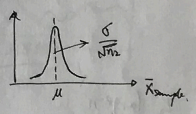
$= N(\mu, \frac{\sigma^2}{n})$

⇓

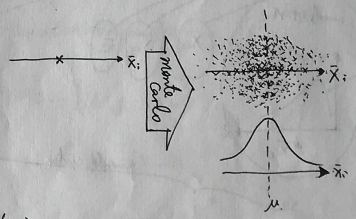
Samples of size n



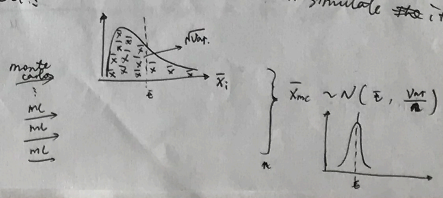
when with a n_2. $(n_2 > n_1)$



What is monte carlo simulation's business:



while if x_i are not in line with $\boxed{iid}$ then x_i might also not distributed ~~two~~ normally. However monte carlo as well ~~ati~~ simulate ~~the~~ its distribution well:



However. According to Lyapunov CLT (Name after Russian Mathematician Aleksandr Lyapnov:

random X_i have to be independent.

~~But~~ not necessarily identically distributed.

$X_1 X_2 X_3 \cdots X_n \sim ?[\mu_i, \sigma_i^2]$

$S_n^2 = \sum^n \sigma_i^2$

$\frac{1}{S_n}\sum^n (X_i - \mu_i) \xrightarrow{d} N(0,1)$

or. $\sum^n X_i \overset{\cdot}{\sim} N(\sum^n \mu_i, S_n)$

$= N(\sum^n \mu_i, \sum^n \sigma_i^2)$

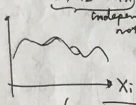
n large enough

$n = 100$ ✗

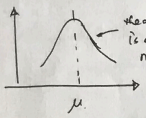
$n \geq 1000+$ ✓

empirically in the assignment.

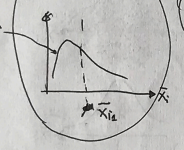
$X_1, X_2 \cdots X_n$ independent not identical



I: large n

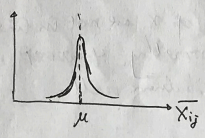


II: modest n.



$\bar{X}_{i2}$ $\bar{X}_{i3}$ $\bar{X}_{i\cdots}$ $\bar{X}_{in}$

monte carlo



Monte Carlo

— 27 oct —

ML-XI

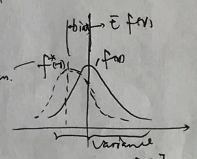
· leave one out error

- leave one point for validation of the model trained by the remaing dataset
- Do this for each point in the train set in turn.
- Average the resulting error

K-fold Cross-validation:

An alternative for leave one error out.

· less time consuming only iterate $\lceil \frac{n}{k} \rceil$ times



$\int f(x)p(x)dx \approx \frac{1}{S}\sum_{i=1}^{S} f(x_i)$

$\{x_i\}_{i=1}^{S}$ are a set of S iid from the distribution $P(x)$

Variance $= E[(f(x) - \bar{E}f(x))^2]$

Bais $= \bar{E}[f(x)] - f^*(x)$

$R(w) \approx \frac{1}{n}\sum^n L(y_i, x_i, w)$

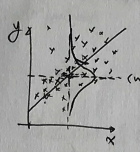
$\approx \frac{1}{n}\sum^n (y_i - f(x_i, w))^2$

$= \frac{1}{n}\sum^n [y_i - w^T\phi(x_i)]^2$

"hold out" a portion of data (validation) and use the model performance on that sub-set of the data as a proxy for the true loss/cost.

Bayes' Rule

model: $y_1, y_2 \cdots y_n$ independent given w.



$y_i \sim N(w^T x_i, a^{-1})$, $a > 0$

$a = \frac{1}{\sigma^2}$ one of the variance. sometimes it's called Precision. Gauss like precision

assumed.

prior: $w \sim N(0, b^{-1} I)$, $b > 0$

Assume a, b are known. $(\theta = w)$

likelihood: $P(y \mid w, X) = \prod P(y_i) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{\sum (y_i - f(x_i))^2}{2\sigma^2}\right)$ (Design matrix) MLE

$$\mathcal{L} = \log P(y) = -\frac{n}{2}\log(2\pi\sigma^2) - \sum \frac{(y_i - f(x_i))^2}{2\sigma^2}$$

$$= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{(y - Xw)^T(y - Xw)}{2\sigma^2}$$

$a = \frac{1}{\sigma^2}$ precision

$$= -\frac{n}{2}\log\left(\frac{2\pi}{a}\right) - \frac{a}{2}(y - Xw)^T(y - Xw)$$

$$= -\frac{n}{2}\log 2\pi + \frac{n}{2}\log a - \frac{a}{2}(y - Xw)^T(y - Xw)$$

Posterior: $P(w \mid y, X) \propto P(y \mid w, X) \cdot P(w)$ ("is proportional to")

likelihood, prior

$$\propto \exp\left(-\frac{a}{2}(y - Xw)^T(y - Xw) - \frac{b}{2} w^T w\right)$$

$P(w) = \frac{1}{(2\pi)^{\frac{n}{2}}} \exp\left(\frac{\sum(w_i^2)}{2\sigma^2}\right)$

$\propto -\frac{b}{2} w^T w$

$$a(y - Xw)^T(y - Xw) + b w^T w$$

$$= a(y^T y - 2yXw + w^T X^T X w) + b w^T w$$

$$= a y^T y - 2a y^T X w + w^T (a X^T X + b I) w$$

$-2a w^T X^T y$

Λ

quadratic form:

$(x - \mu)^T \Lambda (x - \mu)$

$= x^T \Lambda x - 2x^T \Lambda \mu + \text{constant}$

goes to

Bayes' rule

$$P(\theta \mid y) = \frac{P(y \mid \theta)\, P(\theta)}{P(y)}$$

LIKELIHOOD, PRIOR distribution, POSTERIOR, EVIDENCE / marginal likelihood

conditional

(θ is used to represent all the parameters.)

~~y, X~~ PRIOR: $P(\vec{w})$

posterior: $P(\vec{w} \mid y, X) = \frac{P(y \mid \vec{w}, X)\, P(\vec{w})}{P(y, X)}$

(here, X is given as ~~basis~~ input. or basis function of the input.)

$P(w, y) = P(y \mid w)\, P(w)$

$\prod^n N(y_i \mid w^T x_i, \sigma^2)$; $y_i = w^T x_i + \varepsilon_i$

$$P(y, X) = \int p(y \mid \vec{w}, X)\, P(\vec{w})\, d\vec{w}$$

rubbish below this line

~~assume initially, $w_i \sim N(0, \alpha)$~~

~~$w \sim N(0, \alpha I)$~~

~~$p(w \mid y, X, \sigma^2) = N(w \mid \mu_w, C_w)$~~

~~$C_w = (\sigma^{-2} \Phi^T \Phi + \sigma^{-1})^{-1}$~~

~~$\mu_w = C_w \sigma^{-2} \Phi^T y$~~

~~$y = mx + c$~~

~~$x = \begin{bmatrix} \\ \end{bmatrix}$~~

~~$\begin{bmatrix} w_1 & w_2 & \cdots & w_n \\ w_1 & w_2 & \cdots & w_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$~~

~~$w^T x + b$~~

~~$\tilde{w}^T \tilde{x}$~~

~~$\tilde{w} = [w_0 \; w_1 \cdots w_n]$~~

~~$\tilde{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}$~~

conditional expectation

$y \sim N(\beta^T X, \sigma^2 I)$

Response Feature Distribution

Homoscedastic Variance

Posterior of Model Parameters

"MLE" likelihood of the given observations given the model

Prior Probability of Model parameters

$$P(\beta \mid y, X) = \frac{P(y, X \mid \beta)\, P(\beta)}{P(y, X)}$$

$$\text{Posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Normalizing constant independent of the model

"conditional on nothing"

because the Posterior is a function of β.

The evidence will marginalize c by integration. so it's a value independent of c. $\int_a^x x\,dx$

Hence the posterior is proportional to $\propto$ likelihood * prior.

Generative & Discriminative

- Assume some functional form for $P(y)$, $P(x|y)$
- Estimate params of $P(x|Y)$, $P(y)$ directly from training data.
- use Bayes rule to calculate $P(Y|x)$

eg.
- Naive Bayes
- Bayesian Network
- Markov random fields
- Hidden Markov models (HMM)

- Assume some functional form for $P(Y|X)$
- Estimate parameters of $P(Y|X)$ directly from training data.

eg.
- Logistic Regression
- Scalar Vector Machine
- Traditional neutral networks
- Nearest neighbour.
- Conditional Random Field. (CRF)s

$P(x,y)$

	$y=0$	$y=1$
$x=1$	$1/2$	0
$x=2$	$1/4$	$1/4$

give the estimate of y given by what y is given/observed.

$P(y|x)$

	$y=0$	$y=1$
$x=1$	1	0
$x=2$	$1/2$	$1/2$

give the estimate of y only by X.

So we get:

$$\Lambda = aX^TX + bI$$

precision $\frac{1}{\text{covariance}}$ } symmetric and square.

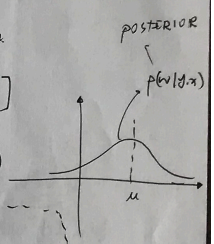
$$aW^TX^Ty = W^T\Lambda\mu$$

$$aX^Ty = \Lambda\mu$$

$$\mu = a\Lambda^{-1}X^Ty \quad \text{------ mean}$$

then: $p(w|y,x) \propto \exp\left[-\frac{1}{2}(w-\mu)^T\Lambda(w-\mu)\right]$

and: $p(w|y,x) \sim N(w|\mu,\Lambda^{-1})$



assume B, a square matrix.

$B + cI$

$Bu = \lambda u$ (eigen value). $cIu = cu$

$(B+cI)u = Bu + cIu = \lambda u + cu = (\lambda + c)u$, } Λ is invertible.

$\lambda > 0,\ c > 0 \Rightarrow \lambda + c > 0$

So, the MAP estimate of $w = \mu$.

$\theta_{MLE} = \max_\theta P(D|\theta)$ (data)

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

$$W_{map} = a\Lambda^{-1}X^Ty$$
$$= a(aX^TX + bI)^{-1}X^Ty$$
$$= (X^TX + \frac{b}{a}I)^{-1}X^Ty$$

regularization parameter

$\left\{\frac{b}{a} = \frac{\sigma_L^2}{\sigma_w^2}\right\}$ likelihood Variance, prior w. variance

Remember MLE: $w_{mle} = (X^TX)^{-1}X^Ty$

The only difference is: $\frac{b}{a}I$

$$\text{Variance} = \Lambda^{-1} = (aX^TX + bI)^{-1}$$
$$= \left(\frac{X^TX}{\sigma_L^2} + \frac{I}{\sigma_w^2}\right)^{-1}$$

Predictive: $P(y|X) = \int P(y|w,x)\cdot P(w)\,dw$

$$= \int N(y|w^Tx, a^{-1})\cdot N(w|\mu,\Lambda^{-1})\cdot dw$$

$$\propto \int \exp\left[-\frac{a}{2}(y - w^Tx)^2\right]\exp\left[-\frac{1}{2}(w-\mu)^T\Lambda(w-\mu)\right]dw$$

$$= \int \exp\left[-\frac{1}{2}\left(ay^2 - 2w^Txya + \cancel{a\,w^TxX^Tw} + w^T\Lambda w - 2w^T\Lambda\mu + \cancel{\mu^T\Lambda\mu}\right)\right]$$

aw^TXX^Tw

$$w^T(axx^T + \Lambda)w - 2w^T(xya + \Lambda\mu) + \underbrace{ay^Ty}_{ay^2}$$

$$Lm = xya + \Lambda\mu$$

$$m = L^{-1}(xya + \Lambda\mu)$$
$$= (axx^T + \Lambda)^{-1}(xya + \Lambda\mu)$$

precision matrix

$(w-m)^TL(w-m)$
$= wLw - 2w^TLm + m^TLm$

$$= wLw - 2w^TLm + m^TLm - m^TLm + ay^2$$
$$= (w-m)^TL(w-m) - m^TLm + ay^2$$

So: $P(y|x) \propto \int \exp\left(-\frac{1}{2}(w-m)^TL(w-m)\right)\exp\left(\frac{1}{2}m^TLm - ay^2\right)dw$

$$\propto \exp\left(\frac{1}{2}m^TLm - \frac{ay^2}{2}\right)$$

$(p(w|y) \propto p(y|w)\cdot p(w))$

$$\propto \exp\left[-\frac{1}{2\sigma_1^2}(y - xw)^2\right]\cdot\exp\left[-\frac{1}{2\sigma_2^2}w^2\right]$$

$$\exp\left[-\frac{1}{2\sigma_1^2}(y - xw)^2 - \frac{1}{2\sigma_2^2}w^2\right]$$

$$-\frac{a}{2}(y - xw)^T(y - xw) - \frac{b}{2}w^Tw$$

$y^T - w^TX^T$

$$-\frac{1}{2}\left[a(y - xw)^T(y - xw) + bw^Tw\right]$$

$$= -\frac{1}{2}\left[ay^Ty - 2ay^Txw - aw^TX^TXw + bw^Tw\right]$$

$2aw^TX^Ty$ $\quad bw^TbI^Tw$

$$ay^Ty - 2ay^Txw - w^T(ax^TX + bI)w$$

Λ

$(X^T - w^T)\Lambda(X - w)$

$\Lambda = aX^TX + bI$ symmetric

$$X^T\Lambda X - 2X^T\Lambda\mu + \mu^T\Lambda\mu$$
$$-2\mu^T\Lambda X$$

$2\mu^T\Lambda = 2ay^TX$

$\mu^T = ay^TX\Lambda^{-1}$

$\mu = a(\Lambda^{-1})^TX^Ty$

$\mu = a\Lambda^{-1}X^Ty$

$2\Lambda\mu = 2a\cdot X^Ty$

$\mu = 2a\Lambda^{-1}X^Ty$

$(w-\mu)^2$

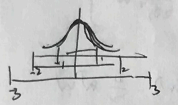
$$p(w|y) \propto \exp\left[-\frac{1}{2}(w-\mu)^T\Lambda(w-\mu)\right] = \exp\left[-\frac{1}{2}\frac{(w-\mu)^T}{\Lambda^{-1}}\right]_{\sigma^2}$$

So: $p(w|y) \sim N(w|\mu,\Lambda^{-1})$

$\frac{1}{n}$ ~~Sample~~

precision

$N(\mu, \sigma^2)$



σ

$\rightarrow \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

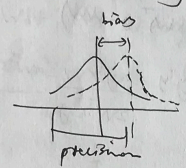
$(-\sigma)^{-\frac{1}{2}}$

$\frac{1}{\sigma}$:

$\sigma^2 = \frac{s^2}{n}$

$\frac{1}{\sigma} = \frac{\sqrt{n}}{s}$

$\sigma +$ precision



$\frac{1}{\sigma^2}$ $\uparrow \in \sigma^2$

$a = \frac{1}{\sigma^2}$ $\frac{1}{a}$

$p(y|w) \cdot p(w)$

$\mathcal{L}_\infty \; \mathcal{L} \; \mathcal{L} \; \mathcal{L} \; \mathcal{L} \; \mathcal{L} \; \mathcal{L}$

auxiliary $\log(\quad)$

$= -\frac{1}{2}\log(2\pi\sigma^2)$

$= -\frac{1}{2}(\log 2\pi + 2\log\sigma)$

$- \left(\frac{(x-\mu)^2}{2\sigma^2}\right)$

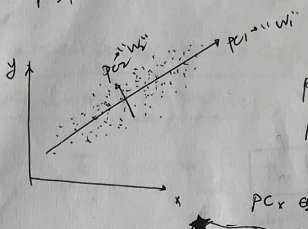
$= -\frac{n}{2}\log 2\pi + \frac{n}{2}\log a$

$- \frac{1}{2\sigma^2}\sum(x-\mu)^2$

$= -\frac{n}{2}\log 2\pi + \frac{n}{2}\log a$

PCA ~~Hotelling~~

ML-XIV



PC1 captures the most variation

PC2 captures the second most variation

PCx ~~capture~~ span the direction of the xth most variation.

X (n×m) — m measurements, n samples

$X = TW^T$

T "scores" $= X$ (n×m) W (m×m) "loading" $= [p_1, p_2 \cdots p_m]$

each of W is a "Principal Component."

$= XV$

$= U\Sigma V^T$

$= U\Sigma$

$I\vec{V} = A\,\vec{w}$ — transformation of basis

$\vec{V} = [v_1\ v_2 \cdots v_n]$ (n dimensions, m samples)

$V^T = \vec{w}^T A^T$

$\vec{w}^T = \vec{V}^T A$

eigen decomposition of (X^TX) (m×m) → W eigen vectors, λ eigen value

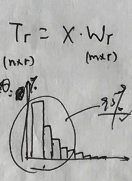
→ ~~total~~ ORDERED cols by the value of λ

W_r = first r cols of W

$\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_m \\ | & | & & | \end{bmatrix}$ r.

then $T_r = X \cdot W_r$ (n×r) (m×r)

Truncated SVD



σ_i

keeping 95% of the original variance.

eigen decomposition of

(X^TX) (m×m) → W, λ

$Av = \lambda v$ — eigen value, eigen vector.

$AQ = Q\Lambda$

$A = Q\Lambda Q^{-1}$

$Q = [v_1, v_2, v_3 \cdots v_n]$

$AQ = [Av_1, Av_2 \cdots Av_n]$

$= [\lambda_1 v_1, \lambda_2 v_2 \cdots \lambda_n v_n]$

$= [v_1, v_2 \cdots v_n]\begin{bmatrix}\lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \lambda_n\end{bmatrix}$

$= Q\Lambda$

$\therefore A = Q\Lambda Q^{-1}$

Orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors.

$Q^TQ = I$

$Q^T = Q^{-1}$

orthogonal unitary → orthogonal (rotate) (stretch) (rotate)

Diagonalization

Σ: $\begin{bmatrix}\sigma_1 & & 0 \\ & \sigma_2 & \\ 0 & & \ddots \sigma_n\end{bmatrix}$ ordered. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$

$A^TA = V\Sigma^TU^TU\Sigma V^T = V(\Sigma^T\Sigma)V^T$ ($U^TU = I$) — eigenvector matrix for A^TA; After A^TA σ^2 for λ

$AA^T = U\Sigma V^TV\Sigma^TU^T = U\Sigma\Sigma^TU^T$ ($V^TV = I$) — eigen matrix for AA^T

$B^TAB = \Lambda$ — diagonal

eigen-decomposition

That's why put data in the eigen basis

$\det(A) = 0$ or have no A^{-1} is singular.

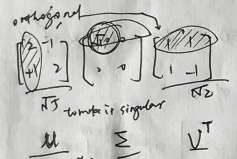
Singular Vector Decomposition "SVD"

$A = U\Sigma V^*$ (matrix)

Unitary matrix

$UU^* = I$ — identity matrix

$A = \begin{bmatrix}2 & 2 \\ 1 & 1\end{bmatrix} =$



orthogonal

$\sqrt{5}$ — torsion is singular

U: eigen matrix U for AA^T

Σ: λ for AA^T

V^T: eigen matrix V for A^TA

A $(2-\lambda)(1-\lambda)-2=0$

$\lambda^2-3\lambda=0$

$\lambda=3$ or $\lambda=0$

$A^*A^T = \begin{bmatrix}2 & 2 \\ 1 & 1\end{bmatrix}\begin{bmatrix}2 & 1 \\ 2 & 1\end{bmatrix} = \begin{bmatrix}8 & 4 \\ 4 & 2\end{bmatrix}$

$(8-\lambda)(2-\lambda)-16=0$

$\lambda^2-10\lambda=0$

$\lambda=10$ or $\lambda=0$

$\begin{pmatrix}8-10 & 4 \\ 4 & 2-10\end{pmatrix}\begin{pmatrix}x \\ y\end{pmatrix}=0$

$\begin{cases}-2x+4y=0 \\ 4x-8y=0\end{cases} \Rightarrow \begin{cases}x-2y=0 \\ \cancel{x-2y=0}\end{cases} \begin{bmatrix}2 \\ 1\end{bmatrix}$

$\begin{pmatrix}8 & 4 \\ 4 & 2\end{pmatrix}\begin{pmatrix}x \\ y\end{pmatrix}=0$

$\begin{cases}8x+4y=0 \\ 4x+2y=0\end{cases}$ $x+2y=0$ $\begin{bmatrix}-1 \\ 2\end{bmatrix}$ Hurray!

ML-XIV-C

PCA

actually is to project a high-dimensional basis into a low dimensional space (p) $p<q$

$T_r = X\,W^T$

scores n×p; X n×q; W^T loading q×p (w.slope) p×q

q basis is projected or loaded in p-dimensional basis space

W: (n components, q features)

$p \times q$

dimension reduction

$X^TX = W\Lambda$

$W^TX^TXW = \Lambda$

~~$AV = \lambda V$~~

$A \cdot W = W\begin{vmatrix}\lambda_1 & & \\ & \lambda_2 & \\ & & \ddots\end{vmatrix}$ (matrix)

$= W\Lambda$

$A^TAW = W\Lambda = \Lambda$

~~$AW = U\Sigma W$~~

$W^TA^TAW = W^TW\Lambda$

$AW =$

$X_{original}$ (q×n) $= A\,X_{new}$ (p×n)

A: q×p — means p basis vector represented by q-dimensional vector

inversely:

$A^{-1}X_{original} = X_{new}$ (p×q, q×n)

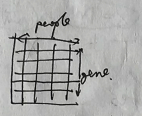
When A is orthogonal $A^{-1} = A^T$ and $A^T \cdot A = I$ — matrix of orthogonal unit vectors

$\therefore A^T X_{original} = X_{new}$

$X^T_{original} A = X^T_{new}$ (n×q, q×p, n×p)

When you have a design matrix X which is in basis of $x_1, x_2 \cdots x_q$, it is not orthogonal. like $\begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}$ $\uparrow x$, so its basis is not $\begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}$, but a covariance matrix. $\begin{bmatrix}\sigma_1 & & \text{cov}(\ ) \\ & \sigma_2 & \\ \text{cov}(\ ) & & \ddots \\ & & & \sigma_q\end{bmatrix}$

Application e.g.



PCA.

Find the Singular Value Decomposition of $\underset{A}{\begin{pmatrix}1&2\\3&4\end{pmatrix}} = \underset{U}{(\quad)}\,\Sigma\,\underset{V}{(\quad)}^T$

$A^TA = V\Sigma^TU^TU\Sigma V^T = V\Sigma^T\Sigma V^T$

$\begin{pmatrix}1&3\\2&4\end{pmatrix}\begin{pmatrix}1&2\\3&4\end{pmatrix} = \begin{pmatrix}10&14\\14&20\end{pmatrix}$

$V = \begin{pmatrix} v_1 & v_2 \end{pmatrix}, \Sigma = \begin{pmatrix}\sqrt{\lambda_1} & 0\\ 0 & \sqrt{\lambda_2}\end{pmatrix}$

to find eigen value of A^TA

$\begin{vmatrix}10-\lambda & 14\\14 & 20-\lambda\end{vmatrix} = 0$

$\lambda^2 - 30\lambda + 200 - 196 = 0$

$\lambda = 15 \pm \sqrt{225}$

$\lambda = \Sigma^T\Sigma$

$(A^TA - \lambda I) = \begin{pmatrix}-5-\sqrt{225} & 14\\14 & 5-\sqrt{225}\end{pmatrix}, \quad v_1 = \begin{pmatrix}1\\ \frac{5+\sqrt{225}}{14}\end{pmatrix}\Big/|v_1|$

or $\begin{pmatrix}-5+\sqrt{225} & 14\\14 & 5+\sqrt{225}\end{pmatrix}, \quad v_2 = \begin{pmatrix}1\\ \frac{5-\sqrt{225}}{14}\end{pmatrix}\Big/|v_2|$

$V = \begin{pmatrix}0.576 & -0.817\\0.817 & +0.576\end{pmatrix}$

$AV = U\Sigma$

$\begin{pmatrix}1&2\\3&4\end{pmatrix}\begin{pmatrix}v_1 & v_2\end{pmatrix} = U\begin{pmatrix}\sqrt{\lambda_1} & 0\\0 & \sqrt{\lambda_2}\end{pmatrix}$

$U = \begin{pmatrix}0.405 & -0.915\\0.915 & 0.405\end{pmatrix}$

$AA^T = \begin{pmatrix}1&2\\3&4\end{pmatrix}\begin{pmatrix}1&3\\2&4\end{pmatrix} = \begin{pmatrix}5 & 11\\11 & 25\end{pmatrix}$

$\lambda^2 - 30\lambda + 125 - 121 = 0$

$\lambda = 15 \pm \sqrt{225}$

$(AA^T - \lambda I)$:

$\begin{pmatrix}-10-\sqrt{225} & 11\\11 & 10-\sqrt{225}\end{pmatrix}, \quad u_1 = \begin{pmatrix}1\\ \frac{10+\sqrt{225}}{11}\end{pmatrix}\Big/\sqrt{u_1}$

$\begin{pmatrix}-10+\sqrt{225} & 11\\11 & 10+\sqrt{225}\end{pmatrix}, \quad u_2 = \begin{pmatrix}1\\ \frac{10-\sqrt{225}}{11}\end{pmatrix}\Big/\sqrt{u_2}$

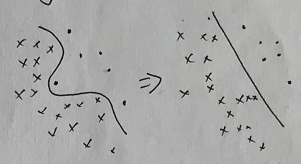
eigen vectors:

---

12 Nov

ML-XV

Feature scaling.



"To avoid that one feature with a broad range of values govern the distance (Euclidean)."

1. Rescaling (min-max normalization)

$x' = \frac{x - \min(x)}{\max - \min}$ $\quad [0,1]$ or $[-1,1]$

2. Mean normalization

$x' = \frac{x - \text{average}(x)}{\max - \min}$ $\quad [0,1), (-1,1)$

3. Standardization (z-score normalization)

$x' = \frac{x - \bar{x}}{\sigma}$ $\quad \sim N(0,1)$

4. Scaling to unit length

$x' = \frac{x}{\|x\|}$

Principal Component Analysis - PCA

1. "Rotate". change of basis so that new features are orthogonal
2. keep the dimension with biggest variance for DR. (max-variance)

capture the essence of the original data

Logistic Regression — 28 Nov

ML-XVI

$\sigma(z) = \frac{1}{1+e^{-z}}$ $\quad$ domain $(-\infty, \infty)$, range $(0, 1)$

- when z goes to infinity $e^{-z} = 0$, $\sigma(z) = 1$
- when z goes to minus infinity e^{-z} goes to infinity. $\sigma(z) \to 0$
- And $\sigma(0) = 0.5$

That qualify σ to be our targeted activity function

$z = wx + b$

weight. $\quad$ bias (bias)

if increase it, the σ evolves faster as a function of x and gets sharper.

INPUT x

$z = wx + b$

σ

activity function. $\quad$ Sigmoid

when bias(b) is very large the neuron always be activated $\sigma(z)$

vise versa, b is very small. the neuron dies.

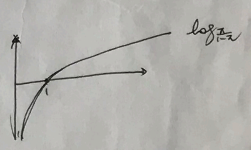
ODDS: the ratio of π, the probability of a positive outcome $P(y=1|x)$ to $(1-\pi)$, the probability of a negative outcome $P(y=1-x)$

$\frac{\pi}{1-\pi}$

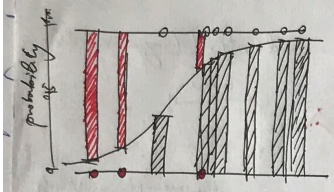
probability: $[0,1]$

odds: $[0, \infty)$

so log odds: $[-\infty, \infty]$

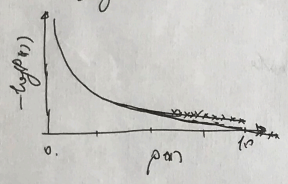


Cross Entropy



The bars represent the predicted probabilities associated with the corresponding TRUE class of each points.

log Loss

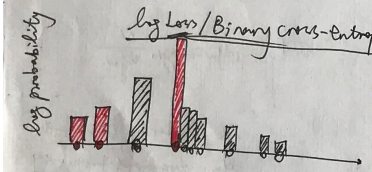


To penalize bad predictions:

If the probability associated with the true class is 1.0, ~~we~~ then the loss ~~is~~ is 0.

~~if~~ Conversely if the probability is low. Say 0.01, we need its loss → huge.

log Loss / Binary Cross-Entropy



$$= -\sum^{c} t_i \log[f(s_i)]$$

$P(y_i)$ is sigmoided proba.

$$M_p(y) = \underbrace{-y_i \log(P(y_i)) - (1-y_i)\log(1-P(y_i))}_{\text{Binary}}$$

y_i: is the label. {1 for black points, 0 for red points}

$P(y_i)$ is the predicted probability of the point being black for all points

That is

~~for~~ each black points: $y=1$. it adds $\log[P(y_i)]$ to the loss $\boxed{y_i \log(P(y_i))}$

conversely for each point $y=0$ (red points) it adds $\log[1-P(y_i)]$ to the loss

$$\boxed{(1-y_i)\log(1-P(y_i))}$$

$$\overbrace{(1-0)}\log(1-P(y_i))$$

CNN

Kernel & Filter.

a matrix of w which multiplied with the input to extract relevant features

the dimension of "kernel" is how the convolution gets its name.

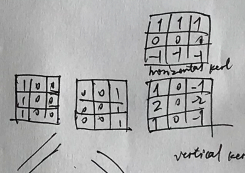
conv2d → kernel (2d matrix)

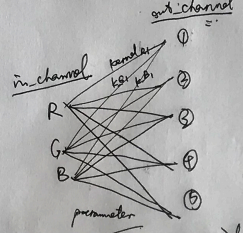
a ~~concatenation~~ collection of multiple kernels each assigned to a particular channel of the input. So it is always one dimension more than kernels

conv2d → kernel (2d matrix) (k_h, k_w) → filter (3d) (d, k_h, k_w)

in_channel × out_channel

- each kernel for every single input channel and every single output channel.
- each filter respond to a single output channel





kernel_a, kernel_g, kernel_b } + bias → output channel_i

So the size of a conv2d(in_c, out_c, kernel_h, kernel_w, ...) is in_c × out_c × (kernel_h × kernel_w)

A essential design choice of any CNN architecture is that the input size grow smaller and smaller ~~and~~ while the number of channels grow deeper and deeper.

This is for the receptive field: from details to big picture.

The expansion of receptive field allows conv layers to combine the low level features (line, edge) into high level features (curves, textures).