Week 4:- Basic CS and Python for ML (2.11):

Python: # - for comments

Don't hardcode, capital letters ⇒ Constant

PAGE No.

DATE / /

Don't do lazy programming, don't let it be error prone, understand, comment

try: except: (In case you expect a wrong user input) '_' variable name for waste variables

* import numpy as np

np.random.random([size]) , np.transpose(A) or A.T

A.T.dot(A) and A*A are different np.linalg.pinv(A).dot(A)

time.perf_counter() (returns time in micro seconds)

Use vectorization as far as possible (Shorter code as well as faster)

Finally time complexity considerations ($O(n)$, $O(\log n)$, $O(n^2)$, $O(2^n)$)

Intro. to Regression (3.1):

- objective of regression
- Decompose loss into bias & variance
- Write objective of Linear regression
- Write expression of analytical sol^n
- Algorithm for comp^nal soln
- Regularization terms to comp^nal

(x_i, t_i) $x \xrightarrow{f} y$ y close to t

MSE. Loss function $= \frac{1}{2}\frac{1}{N}\sum (y_i - t_i)^2$ Mean squared error.

Bias variance decomp. of Regression (3.2):

$$\mathbb{E}(L) = \iint L\,p(x,t)\,dx\,dt = \iint (y-t)^2 p(x,t)\,dx\,dt$$

$$\partial \mathbb{E}(L)/\partial y = 2\int (y-t)\,p(x,t)\,dt = 0$$

$$\Rightarrow \quad y = \int \frac{t\,P(x,t)}{P(x)}\,dt = \mathbb{E}(t|x)$$

$$\mathbb{E}(L) = \int E_D\{y(x;D) - h(x)\}^2 p(x)\,dx \quad \text{(Bias)}$$

$$+ \int E_D\{y(x;D) - E_D\,y(x;D)\}^2 p(x)\,dx \quad \text{(Variance)}$$

$$+ \iint (h(x) - t)^2 p(x,t)\,dx\,dt \quad \text{(Noise)}$$

Finding solution of linear Regression (3.3):

$y = W^T x'$ $x' = [x^T 1]^T$ $= \langle W, x' \rangle$ $t = y + \eta$

$p(t|x, W, \sigma^2)$ $\eta \sim \mathcal{N}(0, \sigma^2)$

$$\boxed{\underset{N\times 1}{y} = \underset{N\times p+1}{X}\ \underset{p+1\times 1}{W}}$$

$\boxed{\max.\ p(t|x)}$

Can add non linear terms to make a linear function

Since the x_i are all i.i.d

$$p(t|x) = \prod p(t_i|x_i)$$

$$= \prod \mathcal{N}(w^T x_i, \sigma^2)$$

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(t_i - w^T x_i)^2\right)$$

$$\boxed{\text{To max this} = \text{Min}\left(\sum_{i=1}^{N}(t_i - w^T x_i)^2\right)} \leftarrow \text{minimize this loss function}$$

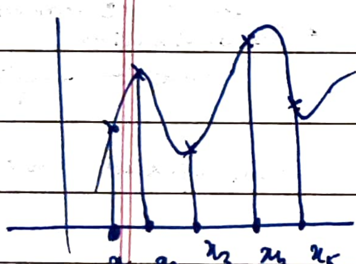
By setting derivative as 0 $\quad W_{ML} = (X^T X)^{-1} X^T t \quad$ (Over determined)

$\underline{(X^T X)^{-1} = \text{pseudo inverse.}}$ $\quad D\times D \quad D\times N \quad N\times 1 \quad N \geq D$

Computing the inverse is very computationally expensive. $O(D^3)$

↳ Therefore we go for <u>gradient descent</u> updates to be more efficient

<u>Linear regression modelling non-linear fns (3.4.)</u>



← Given function

Can approximate a white curve by adding different basis vectors with linear coeff.

Basis set: $\{\phi_1(x), \phi_2(x), \phi_3(x) \ldots \phi_D(x)\}$

eg. $\quad \phi_j(x) = \exp\left[-\frac{(x-\mu_j)^2}{2\sigma^2}\right]$ $\quad$ Can take powers $(x^0, x^1 \ldots)$ or fourier rep also

∴ Just using linear fns we can model non linear ones.

<u>Regularizatn of Linear Regression (3.5)</u>

- We can do this by restricting a models weight from becoming to large

$$y = W_D x^D + \ldots + W_1 x^1 + W_0 x^0$$

- Ways to restrict $- \sum_{1}^{D} W_j^2 \leq \eta$ or reduce no. powers of x.

L2 norm of W or $\|W\|_2$ should be penalized.

So ⟶ Bring y as close to D as possible while the constraint that $\|W\|_2 \leq \eta$ is maintained

We can penalize this by:

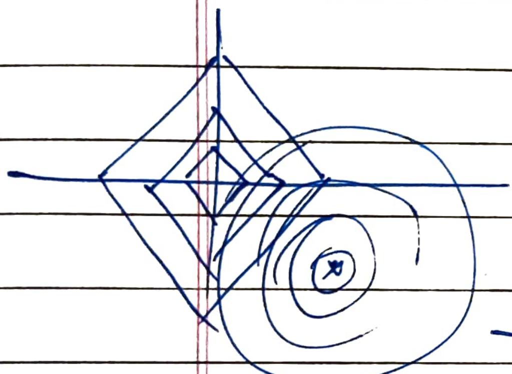
loss fn: $= \frac{1}{2N} \sum (t_i - W^T x_i)^2$

$+ \frac{\lambda}{2} \sum_{j=1}^{D} |w_j|^q$ ($q = 2$ chosen)

λ small $\Rightarrow$ As good as no regulrⁿ. | λ large $\Rightarrow$ Very constrained weights

Lagrange multiplier logic (Adding two convex fns gives us a convex fn)

$q=1$ (L1) Regularization: Here contours are very different.

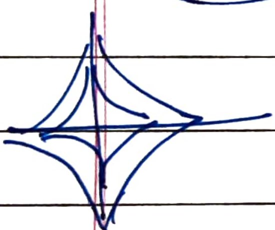


We see the tangential component grazes the corner of the square:

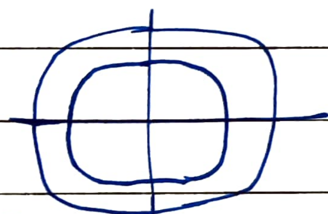
We can drop the feature that doesn't show up at all. (☹)

— Can be used to do feature elimination.

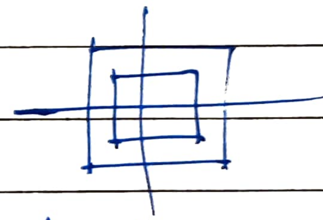
$q = 1/2$



, $q=3$



, $q = \infty$



L2 regularization is called (Ridge Regression / Weight decay)

L1 regularization is called LASSO (Least Absolute Shrinkage & Selection Operator)