Gradient-Descent.

applied to LINEAR Regression

A quick Refresher!

Slope Constant

Recall simple linear Regression. $y = mx + b + \epsilon$

error or Residu

$$\hat{y}_{\text{predicted}} = mx + b$$

$$\text{Error} = y_{\text{actual}} - \hat{y}_{\text{predicted}}.$$

For 'n' observations.

The loss term is mean Sum of Squares of errors.

Thus $$L(m,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right)^2$$

Take gradients]

$$\frac{\partial L}{\partial m} = \frac{2}{N}\sum_{i=1}^{N} -x_i\left(y_i - (mx_i+b)\right) \quad \text{'m' gradient}$$

$$\frac{\partial L}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -\left(y_i - (mx_i+b)\right) \quad \text{'b'\_gradient}$$

Update Rule

New_b = Current_b − αx b_gradient current

New_m = Current m − α x m gradient current

↳ Compute New Line. with New_b, New_m

↳ Compute Loss → get gradients

↳ Iterate. till convergence.

Gradient Boosting For Regression. GBM.

→ Given $(x_1, y_1) \dots (x_n, y_n)$

Fit $F(x)$.

[Initial Model → $F(x)$.]

$F(x)$ can be any Model!

Say we have the following.

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\vdots$$
$$h(x_n) = y_n - F(x_n)$$

Fit a Regression tree to the data.

$$(x_1, y_1 - F(x_1)),$$
$$(x_2, y_2 - F(x_2))$$
$$\vdots$$
$$(x_n, y_n - F(x_n))$$

[Revised Model. $F(x) + h(x)$]

Gradient Descent.

$$\theta_i := \theta_i - \rho \frac{dJ}{d\theta_i}.$$

How is this Related to gradient Descent?

Consider the loss function for the fitted Model $F(x)$

$$L(y, F(x)) = \frac{1}{2}\left[y - F(x)\right]^2$$

We want to minimize

$$J = \sum_i L(y_i, F(x_i))$$

Consider $F(x_1) \dots F(x_n)$ as parameters

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$$

or
$$\frac{\partial J}{\partial F(x_i)} = F(x_i) - y_i \quad \text{— (A)}$$

Re arranging (A)

$$\underbrace{y_i - F(x_i)}_{\text{residual}} = \underbrace{\frac{-\partial J}{\partial F(x_i)}}_{\text{-ve gradient}}$$

∴ We can interpret residuals as negative gradients.

Thus we have the following:

$$\underset{\text{New}}{F(x_i)} := \underset{\text{current}}{F(x_i)} + \underset{\text{current}}{\overbrace{h(x_i)}^{\text{residuals}}}$$

Concept of Boosting

or
$$\underset{\text{new}}{F(x_i)} := \underset{\text{Current}}{F(x_i)} + \underbrace{y_i - F(x_i)}_{\text{residual}} \text{ Current}$$

or
$$\underbrace{\underset{\text{new}}{F(x_i)} := F(x_i) - 1 \frac{\partial J}{\partial F(x_i)}}$$

Compare with Gradient Descent

↳ $$\underset{\text{New}}{\theta_i} = \underset{\text{Current}}{\theta_i} - \rho \frac{\partial J}{\partial \theta_i}$$

[CASE] Regression with Square loss.

Negative gradient:

$$-g(x_i) = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

Algorithm.

- Start with an initial model, say $F(x) = \sum_{i=1}^{n} \frac{y_i}{n}$.
- Iterate till convergence
  - → Calculate -ve gradients $-g(x_i)$
  - → Fit a regression tree h to -ve grad. $-g(x_i)$
  - → $F_{New} = F_{old} + \rho h$, $\rho = 1$

A Fundamental Question arises:

Q why -ve gradient over residuals ?

Consider the types of Loss Functions shown below.

① Square Loss

$$L = \frac{(y_i - F(x_i))^2}{2}$$

← Square Loss Function.
← Outliers are heavily punished

② Absolute loss.

$$L(y, F) = |y - F|$$ ← More Robust to outliers.

③ Huber loss: (More Robust to outliers)

$$L(y,F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| \le \delta \\ \delta(|y-F| - \delta/2) & |y-F| > \delta \end{cases}$$

Negative Gradient vs Residual.

Example: using Huber loss.

Huber loss:

$$L(y,F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| \le \delta \\ \delta(|y-F| - \delta/2) & |y-F| > \delta \end{cases}$$

CASE A

- Update by residual

$$h(x_i) = y_i - F(x_i)$$

CASE B

- update by Negative Gradient.

$$h(x_i) = -g(x_i) = \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| < \delta \\ \underbrace{\delta \, \text{sign}((y_i - F(x_i))}_{} & |y_i - F(x_i)| > \delta \end{cases}$$

Negative gradient → outliers get less attention.

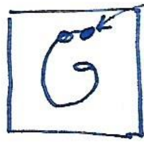
Q] Why should we moderate the effect of outliers?

→ We do not want a 'boost' in the wrong direction !!!

Gradient Boosting for classification

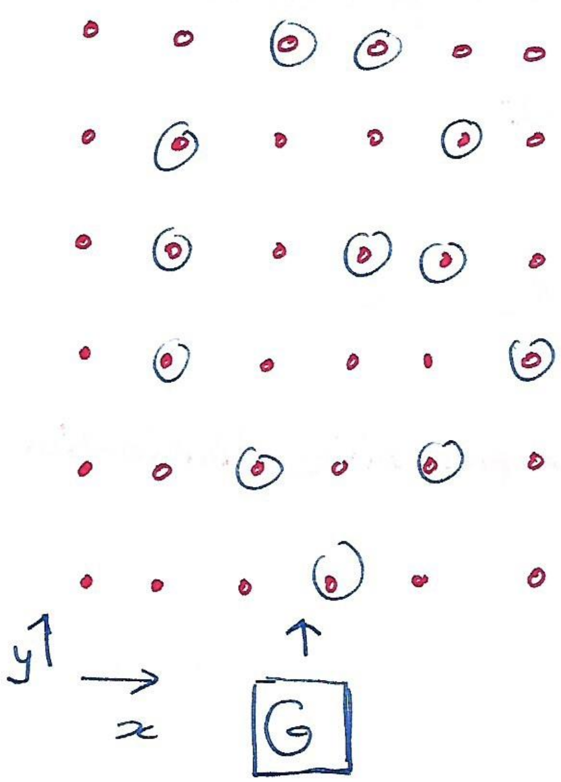
Consider the usecase: Handwritten Capital letter classification

- 26 classes (A to Z)
- 16 features]



an 'ON' Pixel

1] horizontal pos of box
2] Vertical pos " "
3] width of box
4] height of box
5] Total on Pixels.
6] Mean x of 'ON' Pixels
7] Mean y of 'ON' Pixels
8] Mean x variance.
9] Mean y Vari
10] Mean xy correl.
11] Mean of x^2y
12] Mean of xy^2
13] Mean edge count L to R.
14] Correl X edge with y
15] Mean edge cou bottom to top
16] correl. of 'y' edge with



← example of a 36 (6X6) Pixel box.

⊙ denotes an 'ON' Pixel.

Modelling ⊙ scoring

] 26 score functions, one for each class.
$F_A(x)$ assigns a score for class A.

Probabilities:

$$P_A(x) = \frac{e^{F_A(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}} ;$$

$\vdots$

$$P_Z(x) = e^{F_Z(x)} / \sum_{c=A}^{Z} e^{F_c(x)}$$

Consider ith Row $X_i \, [x_1, x_2 \dots x_{16}]$ Label $\bar{y}_i =$ '(

16 features per observation

For each Row | Goal. $P_A(X_i)$ to $P_B(X_i)$.

Predicted label = class that has highest probability.

⊙ Loss function at each data point

Say For $X_5 \, [x_{1,5}, x_{2,5} \dots x_{16,5}]$ $y_5 =$ '6'

Then we can say.

$Y_A(X_5) = 0$

$Y_B(X_5) = 0$

$\vdots$

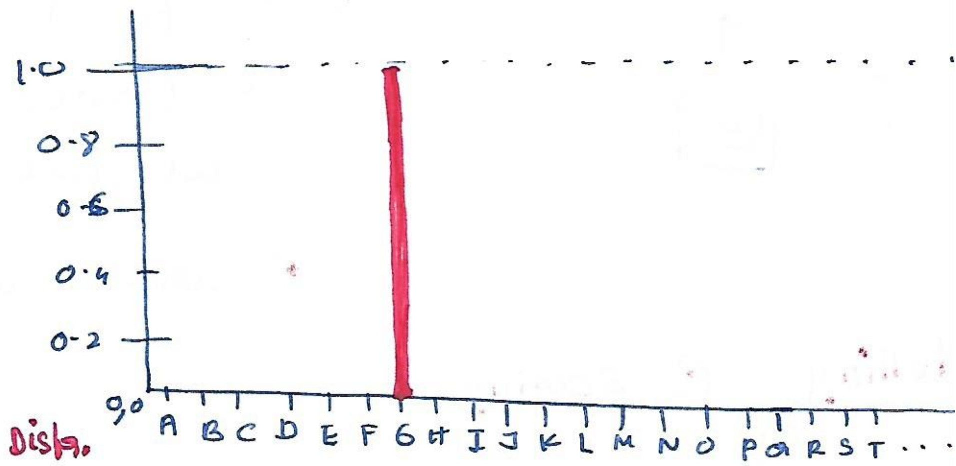
$Y_G(X_5) = 1$

$\vdots$

$Y_{(Z)}(X_5) = 0$

→ Convert this into a true probability distribut

↳ Turn the Label into a 'TRUE' Probability Distn.



Assume: Predicted Prob. at current Model iteration is.

$P_A(X_5) = 0.03$

$P_B(X_5) = 0.05$

$\vdots$

$P_G(X_5) = 0.3$

$\vdots$

$P_Z(X_5) = 0.05$.

Minimize total Loss (KL) - divergenc

→ what is th

Q] ↳ How do we go ahead in this case?

Q] What is our initial guess For $F_{c's}$?

Regression
with Loss Function.

General Procedure.

- Given any differential Loss function L

Start] → Start with initial model F

Iterate until Converge:

- Calculate -ve gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$

Size of the tree
2 Terminal Nodes
(Stump only)

- Fit Regression tree 'h' to -ve gradients. $-g(x_i)$
- update $F := F + \rho h$.

Procedure For alphabet classification case.

→ Start with initial Models For
$F_A = 0, \quad F_B = 0 \quad \dots \quad F_Z = 0.$

Iterate till Convergence

- $-g_A(x_i) = -\frac{\partial L}{\partial F_A(x_i)}$

  $\vdots$

  $-g_B(x_i) = -\frac{\partial L}{\partial F_B(x_i)}$

  $\vdots$

  $-g_Z \quad \dots$

- Fit Regression tree. h_A to -ve grad $-g_A(x_i)$

  $\vdots$

  h_Z.

- update.

  $F_A := F_A + \rho_A h_A$

  $\vdots$

  $F_Z := F_A + \rho_Z h_Z$.

K.L. Divergence → Loss Function.

$$D_{KL}(p \| q) = \sum_{i=1}^{N} p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right)$$

$q(x)$ is approximation.
$p(x)$ is true distribution.

Intuitively. This measures: How much a given arbitrary distribution is away from the true distribution.