7/7/21

# Linear Neural Networks

## 3.1 Linear Regression - Using Deep learning Book

- Regression - Methods of modeling relationship between one or more independant variable and a dependent variable.

- The purpose of Regression in ML is for prediction.

eg/ Predict a numerical value.
- Predicting prices (of homes & stocks)
- Predicting length of stay (for patients in hospital).
- Demand forecasting (for retail sales).

### 3.1.1 (Basic Element of Linear Regression)

example :-
- Estimate the prices of houses (in dollors) based on area (in square feet) and age (in years).

⇒ Dataset = Training dataset / Training Set.
⇒ Each Row (corresponding to one sale). = Example / data point, data instance, Sample.

⇝ Predict (Price) = Label / target.
⇝ Independant Variables = $\underbrace{\text{Age \& Area.}}_{}$
based on the independant variable
called [features (or covariates)]

⇝ n = Number of examples in our datset.
examples are indexed such as :-

$$X^{(i)} = \left[x_1^{(i)}, x_2^{(i)}\right]^T$$ - total corresponding $y^{(i)}$

Linear Model

$$\text{price} = \underbrace{w_{area}}_{\text{weights}} \cdot \text{area} + \underbrace{w_{age}}_{\text{weights}} \cdot \text{age} + \underbrace{b}_{\text{bias.}}$$

bias. (offset / intercept).

- weights determine the influence of each feature on our prediction.
- bias give what value the predicted price should be when all of the features take value 0.
- Without bias we will limit expressivity for our model.

$$\hat{y} = w_1 x_1 + \ldots + w_d x_d + b$$

hat

estimates

=

Can be expressed as.

$$\hat{y} = w^T x + b$$

Recap Vectors

- Vectors are list of scalar values
- it is denoted by lowercase letters $x, y, z$.
- Refering element :-

for $x$, we can use $x_i$ - scalar.

(in/consist of)

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

Vector $x$

Real Values scalar

training dataset

$$\hat{y} = Xw + b$$

- Goal - To find the weight vector $w$
  - The bias term $b$

## Loss Function

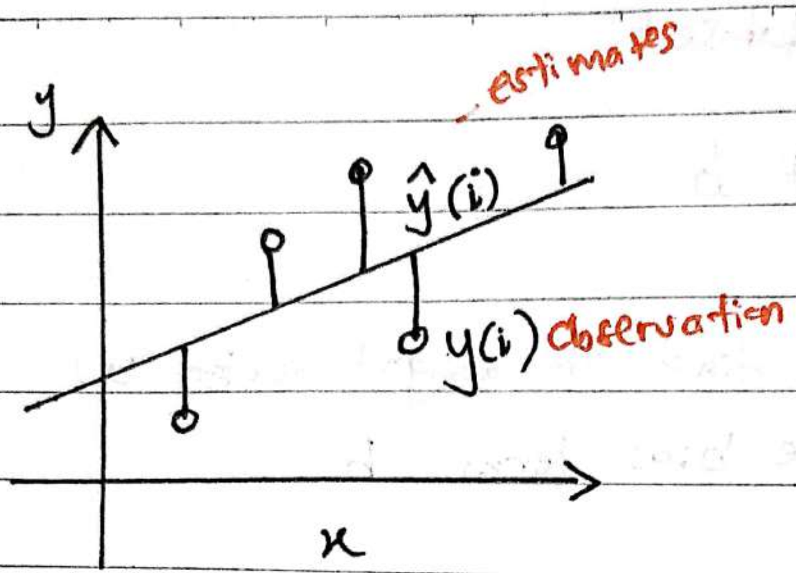
Loss function - measure the distance between the real & predicted value.

- Loss - usually non-negative number
  - perfect predictions incur a loss of 0.

- Squared error

$$l^{(i)}(w, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- $\left(\frac{1}{2}\right)$ will ~~im~~ improve notationally, canceling out when we take derivates loss.




(Fit data with a linear model)

- difference between $\hat{y}(i)$ & $y^{(i)}$ increase
  ⤳ Larger loss due to quadratic depencence.

- quadratic dependance -

$$L(w,b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\underbrace{\left(w^T x^{(i)} + b - y^{(i)}\right)^2}_{\text{with bias.}}$$

Derivation for the equation.

$$\hat{y} = w_1 x_1 + \dots + w_d x_d + b.$$

$$\hat{y} = w^T x + b \quad \text{—} \; (1)$$

Loss function $\quad l^{(i)}(w,b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$

$$= \frac{1}{2}\left(w^T x + b - y^{(i)}\right)^2 \quad \text{—} \; (2)$$

Average $\quad \frac{1}{n}\sum_{i=1}^{n} \quad \text{—} \; (3)$

Combine

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

(1) into (2)
2 into (3)

$$w^* b^* = \operatorname{argmin} \, L(w,b)$$

$\downarrow$

To minimize the total loss across all training examples.

# * Gradient Descent

- An algorithm that consist of iteratively reducing the error by updating the parameter in the direction that incrementally lowers the loss function.

- Average of the losses computed on every single example in the dataset.

- Problem : Extremely slow, we must pass entire dataset before making a single update.

- Solution :- Use a random minibatch of examples every time we need to compute the update.
  → Called minibatch stohastic gradient descent.

→ In each iteration, we sample a minibatch $\underbrace{\mathcal{B}}_{\text{consist of fixed number of training examples.}}$

→ Compute the derivative (gradient) of the average loss on the minibatch.

→ We multiply the gradient by a predetermined positive value $\eta$.

$$(w,b) \leftarrow (w,b) - \frac{\eta}{|\beta|} \sum_{i \in \beta} \partial_{(w,b)} l^{(i)} (w,b)$$

$$w - \frac{\eta}{|\beta|} \sum_{i \in \beta} \partial_w l^{(i)}(w,b) = w - \frac{\eta}{|\beta|} \sum_{i \in \beta} x^{(i)} \left( w^T x^{(i)} + b - y^{(i)} \right)$$

- we initialize the values of the model parameters, at random.

$$b - \frac{\eta}{|\beta|} \sum_{i \in \beta} \partial_b l^{(i)} (w,b) = b - \frac{\eta}{|\beta|} \sum_{i \in \beta} \left( w^T x^{(i)} + b - y^{(i)} \right)$$

- we iteratively sample random minibatches from the data.
- Updating the parameters in the direction of negative gradient.

$|\beta|$ = Number of examples in each minibatch (the batch size)
$\eta$ = Learning rate.

8/7/21

# Linear Regression

Equation : $Y = a + bx$

- $a$ : intercep
- $x$ : explanatory variable.
- $Y$ : dependant Variable
- $b$ : slope (gradient)

old eq :

$y = mx + c$

## Least-Squares Regression

- Used for fitting a regression ~~a regression~~ line.

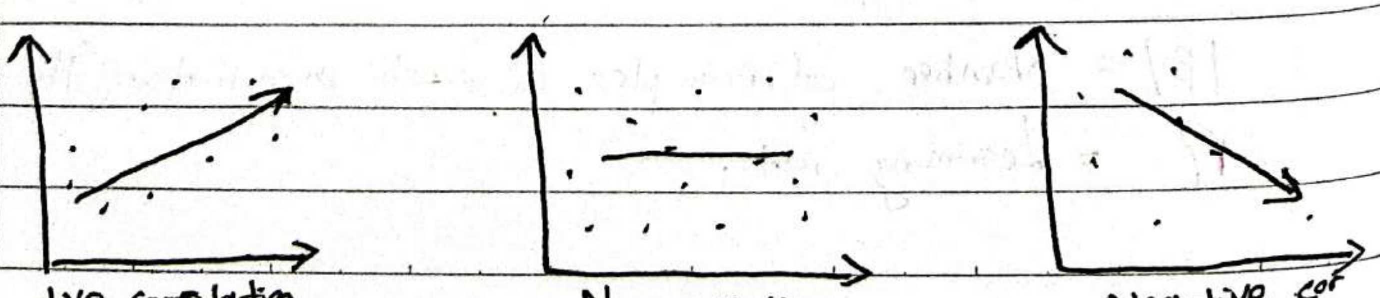
~ calculates the best fitting line

- by minimizing the sum of squares of vertical deviations from each data points to the line.

## Tips

**Correlation** . ~ measure how strong a relationship between two variables.

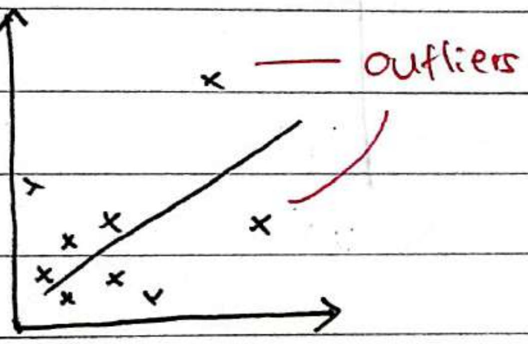
eg



+ve correlation     No correlation     Negative cor example : dataset 'Television, Physician & Life Expectancy'.

- Contains 40 ~~countries~~ countries.
- 8 removed 32 remaining
- correlation coefficient, $r = 0.852$
- $r^2 = 0.726$

= 72.6% of variation in one variable.




- Most of the data points are at the left and the ones on the right called outliers.

- ~~line~~ Point lies far from other data in the horizontal direction :- Influential observation.
- After the influential observation removed :-

correletation, r dropped to 0.427.

$r^2 = 0.182$

=< 20% of variation in number of people per physician

11/7/21

# Recall Supervised Learning

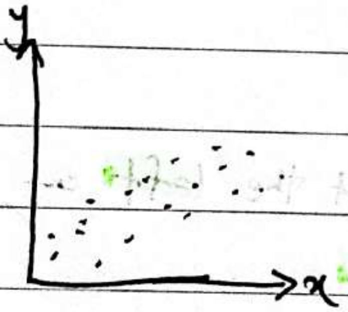
$\xrightarrow{\quad x \quad}$ $\longrightarrow$

## example

Living areas and prices of 47 houses.

| $x$ (Living area) | $y$ (1000 \$s) |
|---|---|
| data | data. |

Produce scatter plot. (+ve correlation)



$x^{(i)}$ = input (living area) called ~~inpt~~ input ~~features~~

$y^{(i)}$ = output (price) target variable

A pair of $\underbrace{(x^{(i)}, y^{(i)})}_{\text{Used to learn}}$ - Called ~~traing~~ training example.

a list of n training examples. ~~n~~.

index for traing set.

Traning Set. $(x^{(i)}, y^{(i)})$ -①

$\{(x^{(i)}, y^{(i)}) ; i=1 \dots n\}$ -②

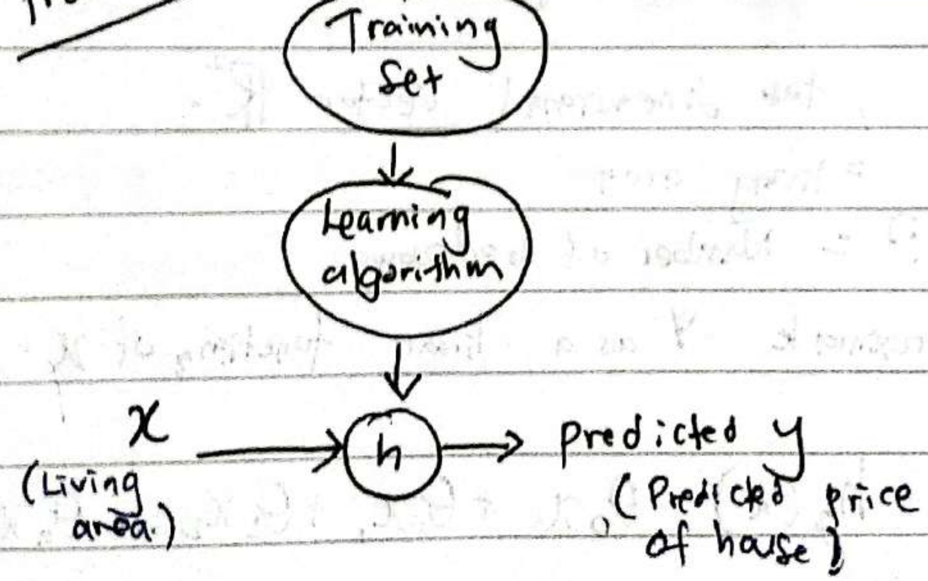
$X$ = Input ; $Y$ = Output

$X = Y = \mathbb{R}$ - real numbers.

Given Training Set $h : X \longrightarrow Y$

$h(x)$ predictor of value $y$.

**Process**




- Predict continuous problem ~ Regression Problem.

- y can only take small discrete values

  - we call it as classification problem.

# Based on Lecture Notes Andrew NG

## Linear Regression

Using a data set

| Living area $(feet)^2$ | # bedrooms | Price (1000$s) |
|---|---|---|
| | | |

- $x$'s, two dimensional vector $\mathbb{R}^2$.
- $x_1^{(i)}$ = living area
- $x_2^{(i)}$ = Number of bedrooms.
- Approximate $Y$ as a linear function of $x$.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \dots \theta_n x_n^{(i)}$$

assume $x_0 = 1$ $\therefore$
(intercept term)

$\theta_i$ = parameters / weights.

~~We take~~

We take the eq as :

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

- make $h(x)$ close to $y$.

linear eq:

$$y = mx + c$$

$$h(x) = \theta x$$

~~We~~ - We use loss function (least square function)

- square error.

$$\frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2 \quad -① $$

estimates

In this case :

$$\hat{y} = h_\theta(x) \quad -② \quad \text{sub}$$

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

≠ We find sum $h_\theta(x) = \sum \theta_i x_i$

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

#.

# Least Mean Square LMS Algorithm

~ To make $J(\theta)$ minimize we need to chose $\theta$.

~ Algorithm that repeatedly changes $\theta$ to make $J(\theta)$ smaller.

~ We use Gradient Descent Algorithm.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta). \quad (1)$$

$\alpha$ = Learning rate. $\alpha \in \mathbb{R}^+$ (range between 0 to 1).

$\theta_j$ = Parameters.

$j = 0, \ldots, n.$

~~$\alpha$~~

~ This algorithm repeatedly decrease the steepness of J.

We take the eq from $J(\theta)$.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2.$$

# Recap Partial derivative

## Basic differentiation concept

$$\frac{\partial}{\partial x}(ax+b)^n = n(ax+b)^{n-1}(a)$$

$$=(n)(a)(ax+b)$$

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

Apply basic concept :

$h_\theta(x) = \theta_0 x_0 + \ldots$

$= \sum \theta_j x_j$

We do this to sub into $\theta_j$

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

$$\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{\cancel{2}}(\cancel{2})\left[\frac{\partial}{\partial\theta_j}(h_\theta(x)-y)\cdot(h_\theta(x)-y)\right]$$

$$=\left[\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{d}\theta_j x_j\right)\cdot(h_\theta(x)-y)\right]$$

$$=\left[x_j\cdot(h_\theta(x)-y)\right]$$

$$=(h_\theta(x)-y)x_j \quad \#.$$

(2)

Sub eq (1) - into (2)

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta) \quad - (1)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \left(h_\theta(x) - y\right) x_j \quad - (2)$$

$$\theta_j = \theta_j - \alpha \left(h_\theta(x) - y\right) x_j .$$

For Single training example.

- The rule is called LMS update rule.
  (Least Mean Squares).
- Widrow - Hoff learning rule.
- directly proportional to the error term
  $\left(y^{(i)} - h_\theta(x^{(i)})\right)$
- if the training set have more than one example :-

- Repeat until convergence :-

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}, \text{ for every } j$$

succinct = short / compact.

$$\theta = \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)}.$$

- method looks at every example in the entire training set on every step.

- Called batch gradient descent.

- Gradient descent always converges to the global minimum.

- J is a convex quadratic function

-

The normal equations

J other than

- Another way of optimizing gradient descent.
- This method will minimize J by taking its derivatives with respect to $\theta_j$ & setting them to zero.

Least Square ~~Matrix~~ Using Matrix Derivatives

Training Examples :-

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(n)})^T- \end{bmatrix}$$ design matrix

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

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

$$h_\theta(x^{(i)}) = (x^{(i)})^T\theta.$$

$\cancel{X\theta =}$

$$X\theta - \vec{y} = \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(n)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(n)}) - y^{(n)} \end{bmatrix}$$

$$\frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= J(\theta)$$

$\cancel{(X\theta - y)^T(X\theta - \vec{y})}$

Eq:

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

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