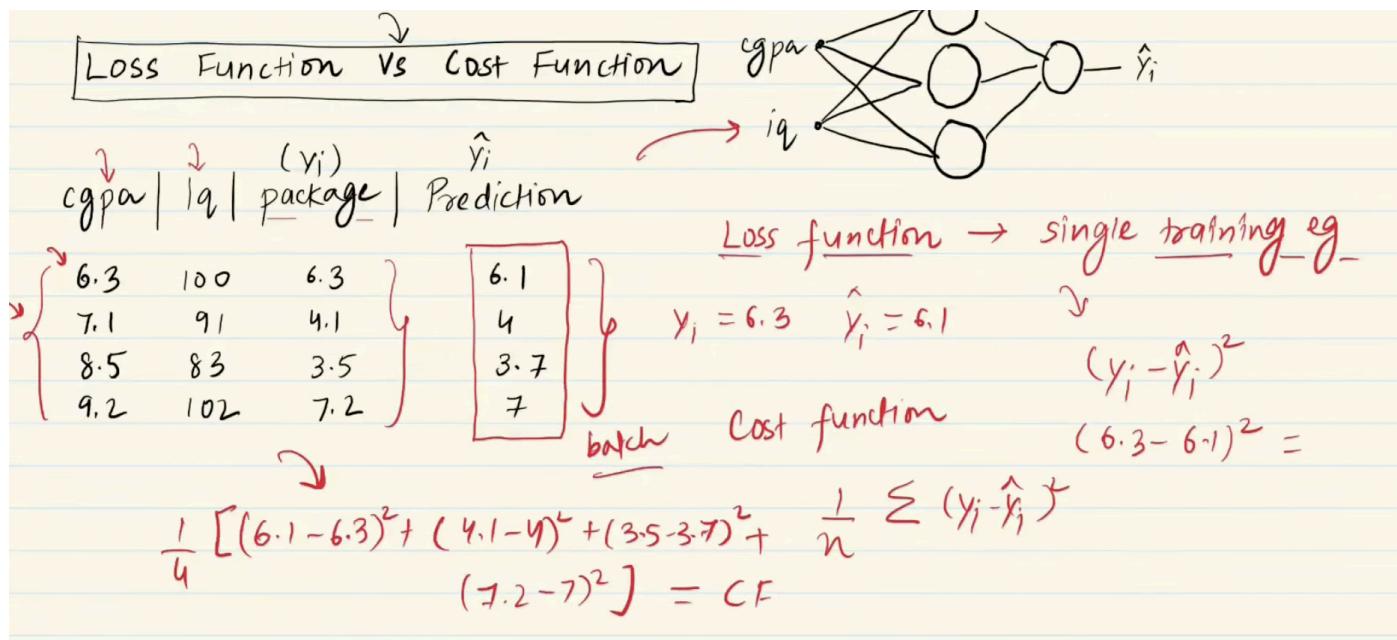
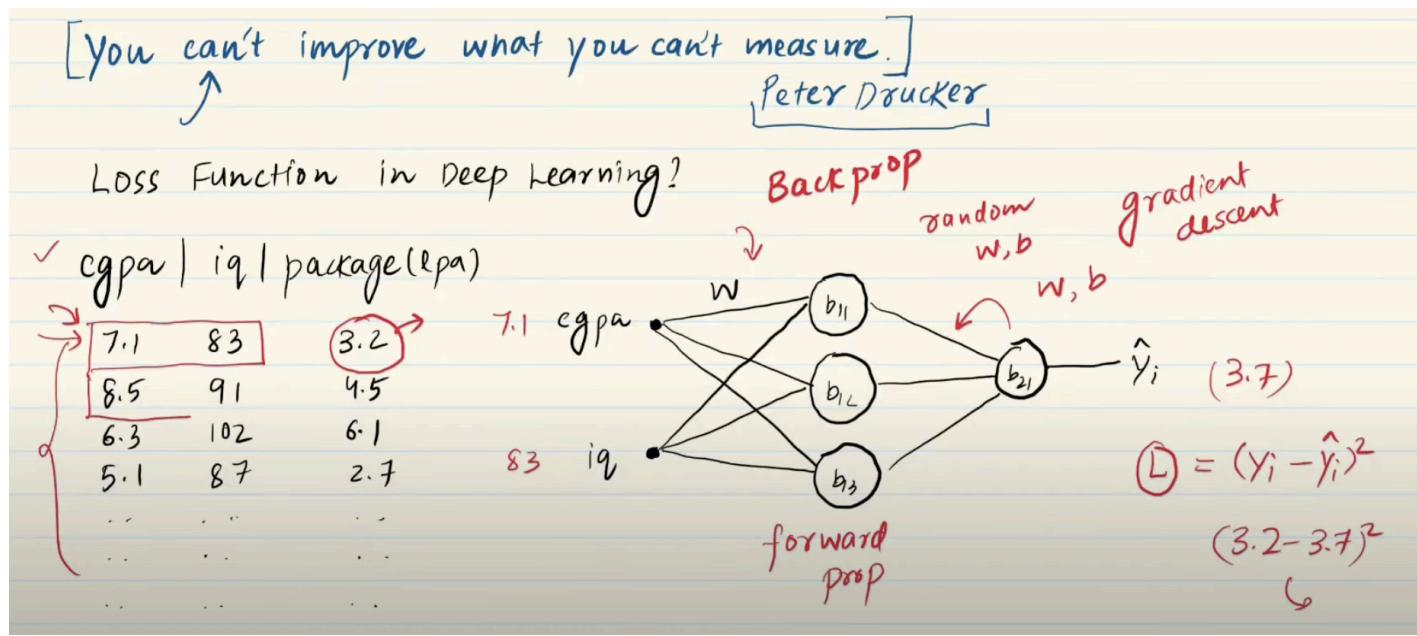


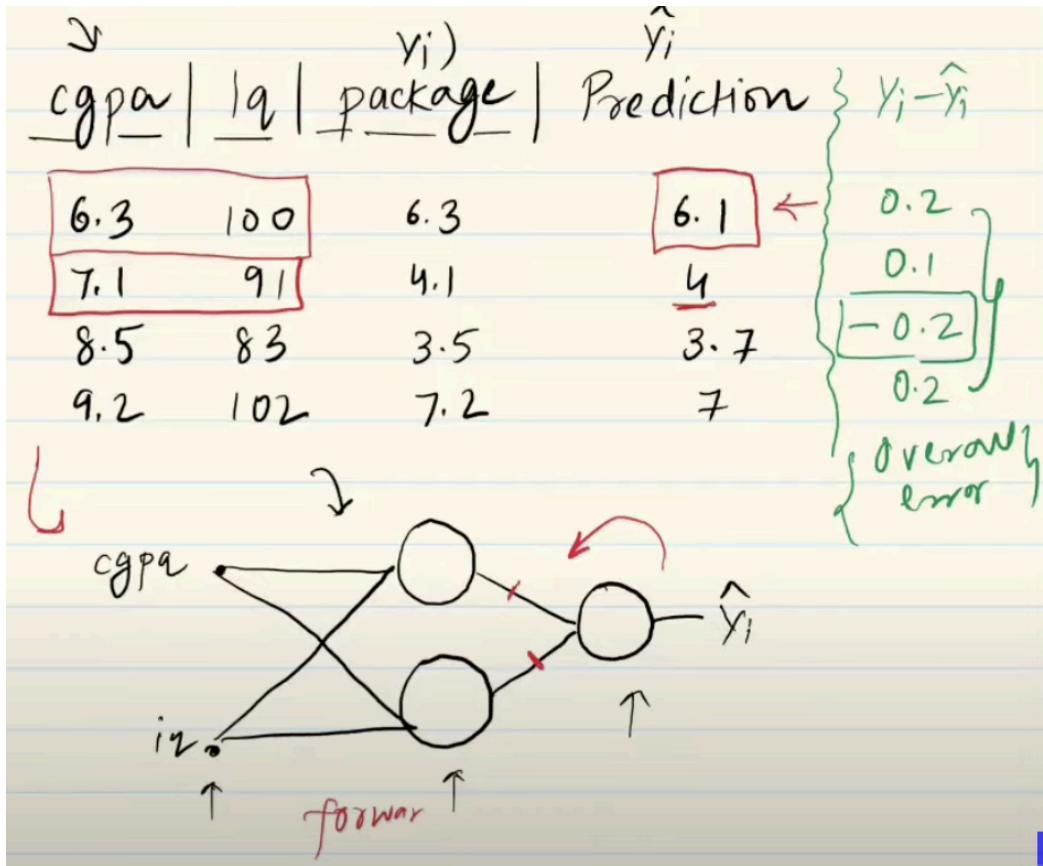
Lecture 4
Loss functions

Read from: <https://medium.com/@ibtedaazeem/loss-functions-in-deep-learning-e4bd353ea08a>



Mean squared error (L2 loss)

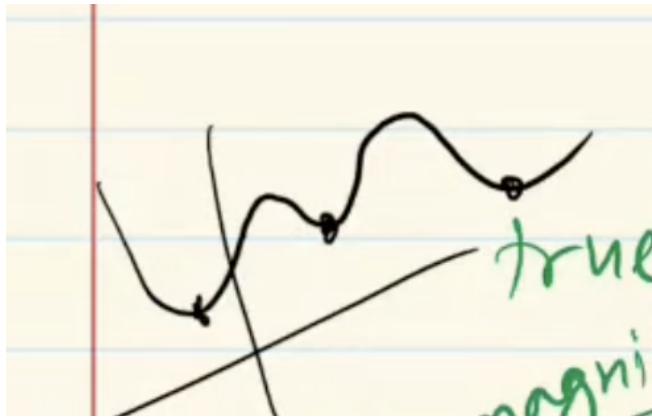
Why squared?



Advantages

- 1) Easy to interpret \nearrow
- 2) Differentiable (GD)
- 3) 1 local minima

1 local minima means: there exists one value of weight and bias for which loss is minimum
 Unlike this, where multiple solutions exist.



Drawback

- 1) Error unit (squared) \rightarrow diff
- 2) Robust to Outliers (Not)

Error unit is squared:

$$\begin{aligned}
 & \text{true-predict} \rightarrow 1^{\text{unit}} \rightarrow 1^{\text{unit}} \\
 & 2^{\text{unit}} \rightarrow 4^{\text{unit}} \\
 & 4^{\text{unit}} \rightarrow 16^{\text{unit}}
 \end{aligned}$$

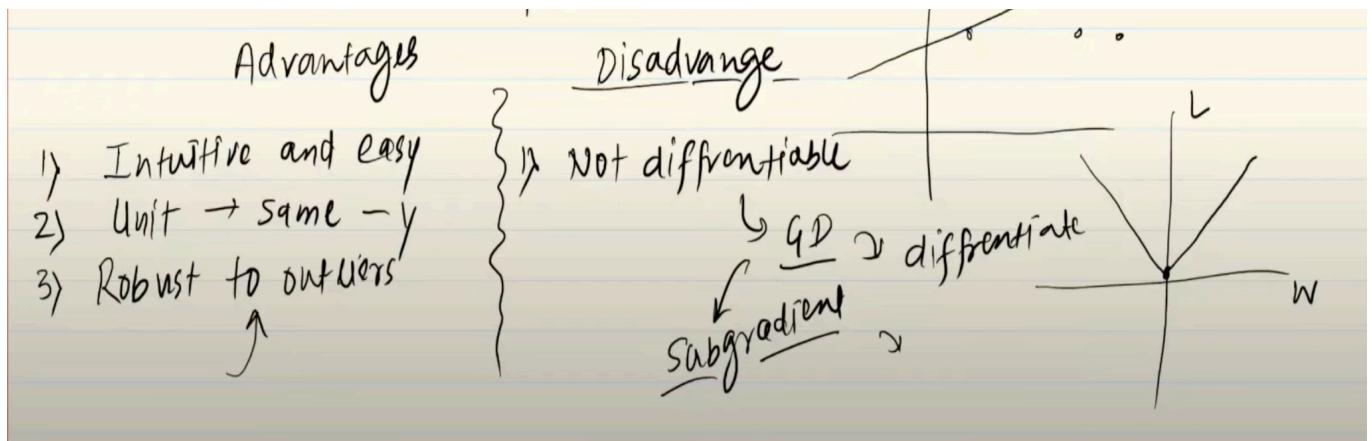
Error gets magnified due to the squared unit , weight updates will be drastic for points away (these points will be punished more) from the line, and causing the line to move further away.

Not robust to outliers:

A student has a Package high, error becomes too high due to squared difference. The line deviated from the true nature of the data.

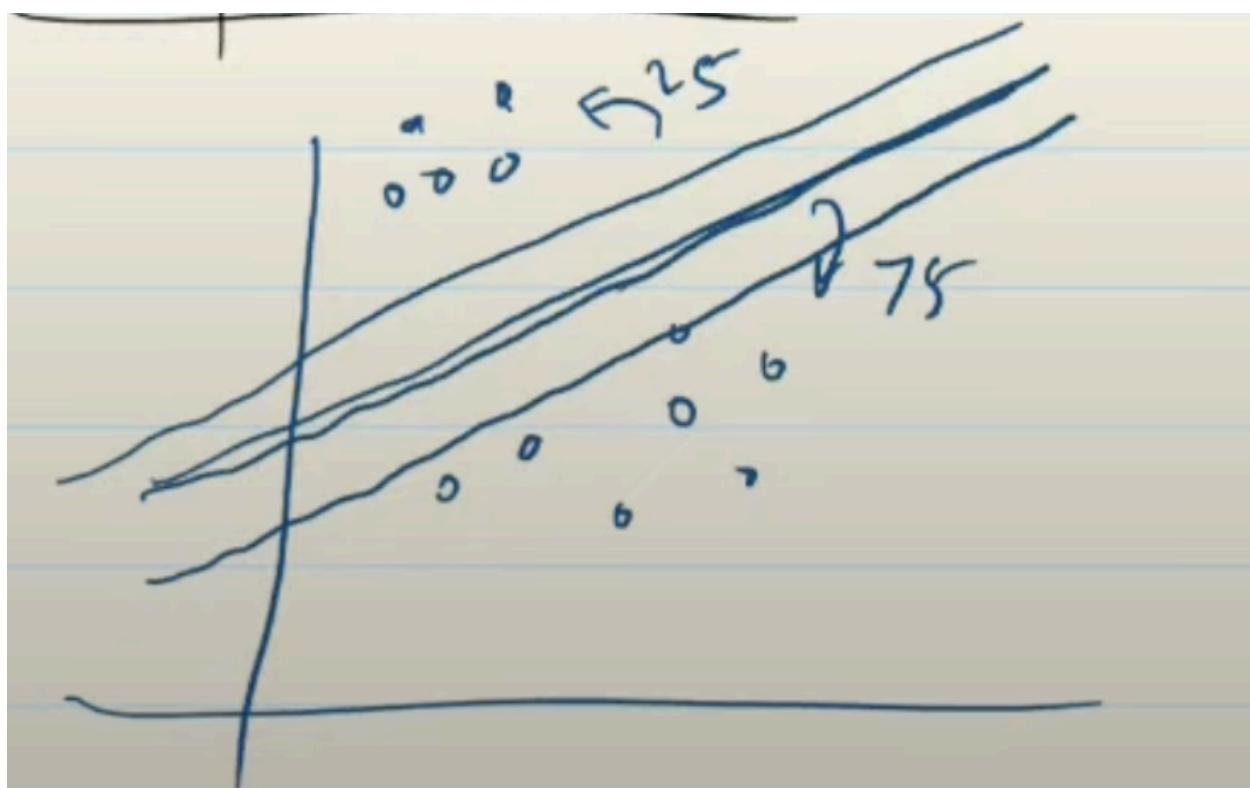
Note: the last layer neuron must use a linear activation function for regression problems.

Mean absolute error (L1 loss)



Huber loss

Assume our data has 25% outliers the above methods won't behave well



3. Huber Loss ✓

$$L = \begin{cases} \frac{1}{2} (y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta |y - \hat{y}| - \frac{1}{2} \delta^2 & \text{otherwise} \end{cases}$$

mse mae

4. Classification loss function

4. Binary Cross Entropy

- classification
- Two classes

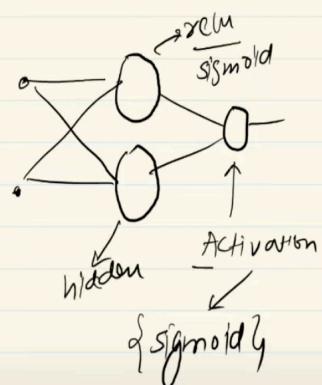
cgoal/q placement
1 0

8	80	1
7	70	0
6	60	0

$$\text{Loss function} = -y \log(\hat{y}) - (1-y) \log(1-\hat{y})$$

y → actual value/target

\hat{y} → NN prediction



$$\text{cost function} = -\frac{1}{n} \left[\sum_{i=1}^n y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right]$$

Where K is # classes in the data

1 point $\rightarrow 3$

$$L = -y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$$

$$f(z) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$f(z) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Activation \uparrow
neurons
class \leftarrow categories

Sparse Categorical Cross Entropy

$L \sim$

$$-1 \times \log(0.1)$$

$$-1 \times \log(0.4)$$

$$-1 \times \log(0.2)$$

$$\begin{bmatrix} 7 & 70 \\ 8 & 80 \\ 6 & 60 \end{bmatrix}$$

cgpa | iq placed DHE

7	70	Yes	1
8	80	No	2
6	60	Maybe	3

$$\begin{bmatrix} 0.1 & 0.4 & 0.6 \end{bmatrix}$$

$$\begin{bmatrix} 0.6 & 0.2 & 0.2 \end{bmatrix}$$

Why integers (1, 2, 3...) don't add extra weight

The integers here are not treated as numeric values like regression targets.
Instead, they are indices that point to a class.

Example:

Suppose model output (softmax) for 3 classes is [0.1, 0.7, 0.2].

- If the true label is 1 (class index), the loss only looks at the probability of index 1 (here 0.7).
- Loss = $-\log(0.7)$.

It doesn't matter that the label is 2 or 3 numerically larger. The loss function does not treat "3" as three times "1" — it just uses it as a lookup index.

So why use Sparse Categorical Crossentropy?

- Saves memory and computation (no need for one-hot vectors).
- Convenient when you have many classes.
- Both methods give the same result.