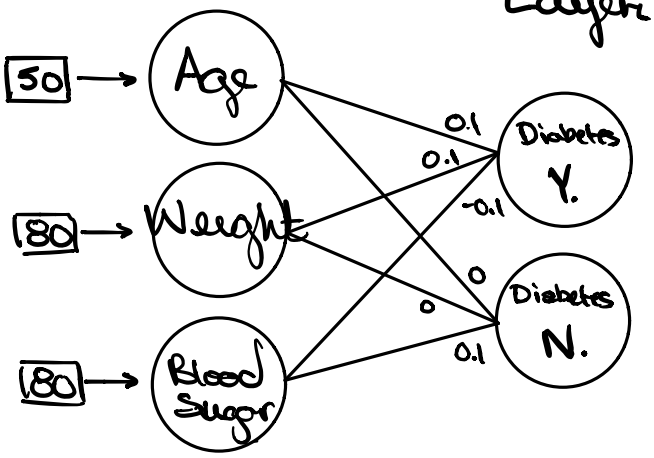
Input Layer Prediction Layer



Stochastic Gradient Descent can be used to minimize the loss function by changing the model weights, a iterative process.

Yes Score:

$0.1(50) + 0.1(80) - 0.1(80) = 5$

No Score:

$0.1(80) = 8$

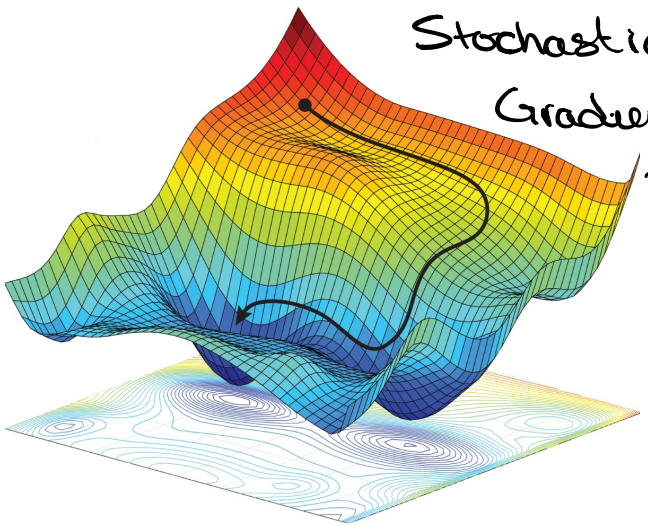
These functions can have loss values

→ These scores get put into softmax func. to produce probabilities.

Batch size – the number of rows or images in a step.

Epoch – one cycle through all the data.

Learning Rate – proportionate steps.



Stochastic Gradient Descent.

Backpropagation –

the mechanism for calculating the weight changes in SGD. (ReLu).

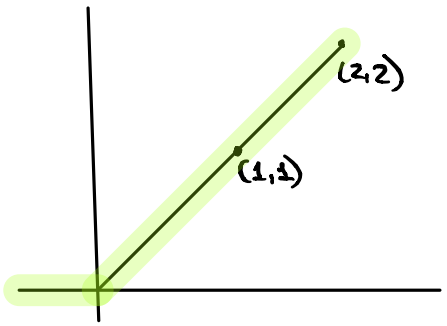
⟹ Work backwards through the layers.

From right to left w- calculus chain rule.

Rectified Linear Unit (or ReLu).

→ activation function for Deep L..

The equation,



$f(x) = \max(0, x)$

Purpose of ReLu,

① Account for Interaction effects
when a variable's effect is conditioned by another variable.

② Account for non-linear effects
when the prediction or reality can not be expressed through solely linear operations based on feature data.
I.E Graphing a variable on x-axis and the prediction of y will result in a line that is not straight.

How-it-works,

Interactions:
ReLu activation function captures the opposition or cooperation of the involved nodes → based on the weights of the connected nodes.

Non-Linearity
ReLu is a mechanic for creating non-linear behaviour from linear operations. While ReLu is simple $\frac{dy}{dx} = \{0, 1\}$, the two slopes constitute a non-linear function - and in binary options allow one to include the input of former nodes. This can be tuned by node bias.

The construction and layering of multiple ReLu nodes allow for creating sophisticated non-linear neural nets.

Impact to Gradient Descent,

The more layers which are added to a model analytically maintain or reduce the degree to which the models weights are changed. For sigmoid-like functions like $\tanh(x)$ which produce small derivatives in most of their ranges often produce very computationally inefficient, perhaps unworkable, layers.

This is known as the vanishing gradient problem.

Since ReLu has $\frac{dy}{dx} = \{0, 1\}$, the combinations and hierarchies of ReLu nodes produce compositional derivatives that do not have an affinity to (0). → Assuming reasonable sized batches.

Alternatives

Generally ReLu is best; not many industry/academic uses of other functions.

Leaky ReLu has slightly more information integrity, as all it's outputs reflect it's input.

$$f(x) = \max(\alpha \cdot x, x).$$