

Perceptrons, Backprop, and Gradient Descent

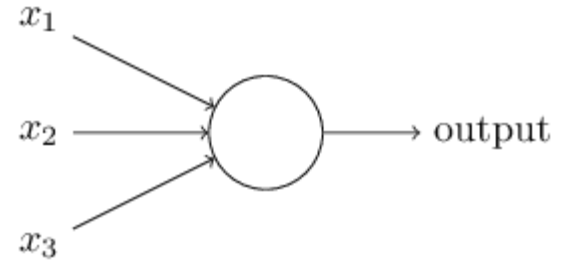
RRC Summer School '23

[illegible]

Perceptron

Takes in several binary inputs

Gives out one binary output

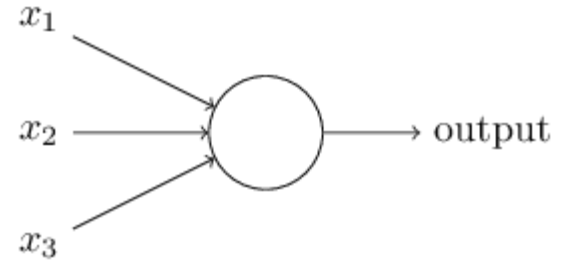


Perceptron

Takes in several binary inputs

Gives out one binary output

- How does it know what to output?

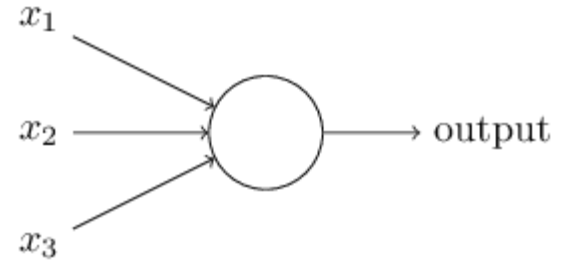


Perceptron

Takes in several binary inputs

Gives out one binary output

- How does it know what to output?
- Preferred listening to inputs
 - Weights assigned to inputs

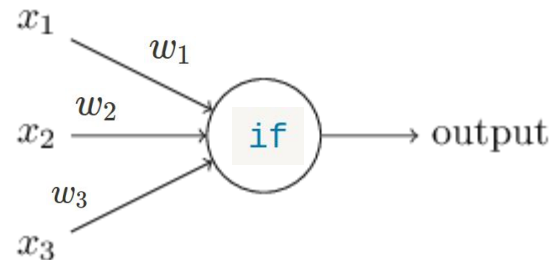


Perceptron

Takes in several binary inputs

Gives out one binary output

- How does it know what to output?
- Preferred listening to inputs
 - Weights assigned to inputs



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

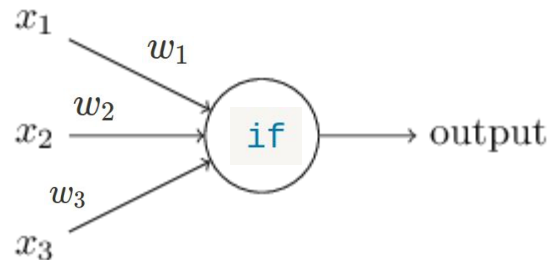
Perceptron

Takes in several binary inputs

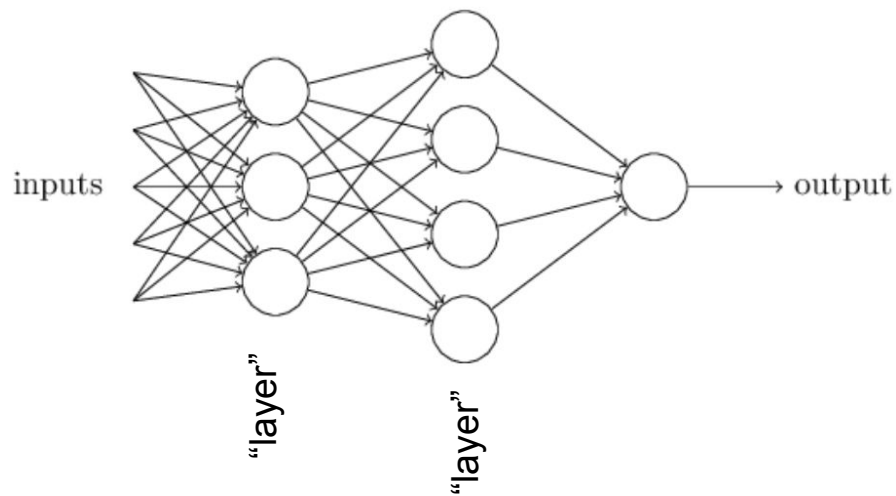
Gives out one binary output

- How does it know what to output?
- Preferred listening to inputs
 - Weights assigned to inputs
- Knobs: weights and thresholds

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$



Scaling Up



Cascading Layer after Layer

- The first layer of perceptrons - is making three very simple decisions, by weighing the input evidence.

Cascading Layer after Layer

- The first layer of perceptrons - is making three very simple decisions, by weighing the input evidence.
- Second perceptrons is making a decision by weighing up the results from the first layer of decision-making.

Cascading Layer after Layer

- The first layer of perceptrons - is making three very simple decisions, by weighing the input evidence.
- Second perceptrons is making a decision by weighing up the results from the first layer of decision-making.
- Thus complex decision boundaries can be learnt

Bias

Gives a feel of what it takes for a neuron to fire

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

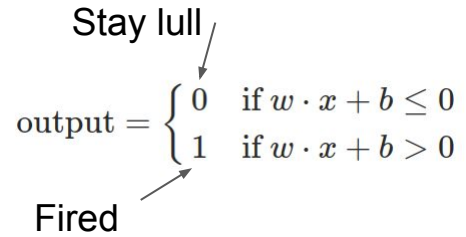
Bias

Gives a feel of what it takes for a neuron to fire

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Stay lull

Fired

The diagram illustrates the role of the bias term in a neuron's activation function. It shows a piecewise function where the output is 0 if the weighted sum of inputs plus the bias is less than or equal to zero, and 1 otherwise. An arrow points from the text 'Stay lull' to the '0' case, and another arrow points from the text 'Fired' to the '1' case.

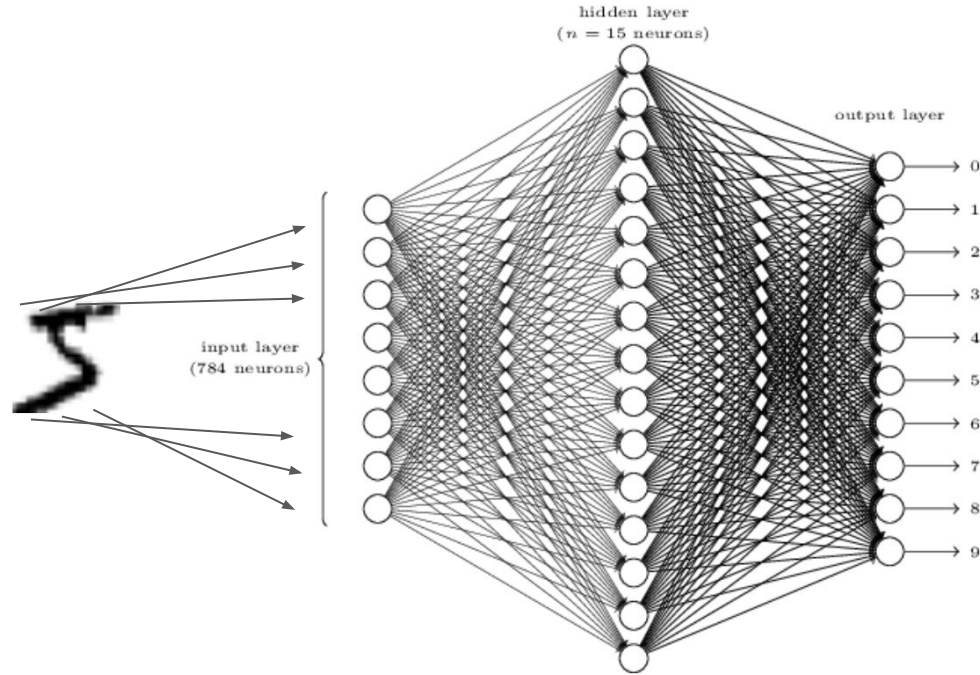
Learning

- Is about automatically tuning the weights and biases of neurons
- So that model's output is in line with the training data

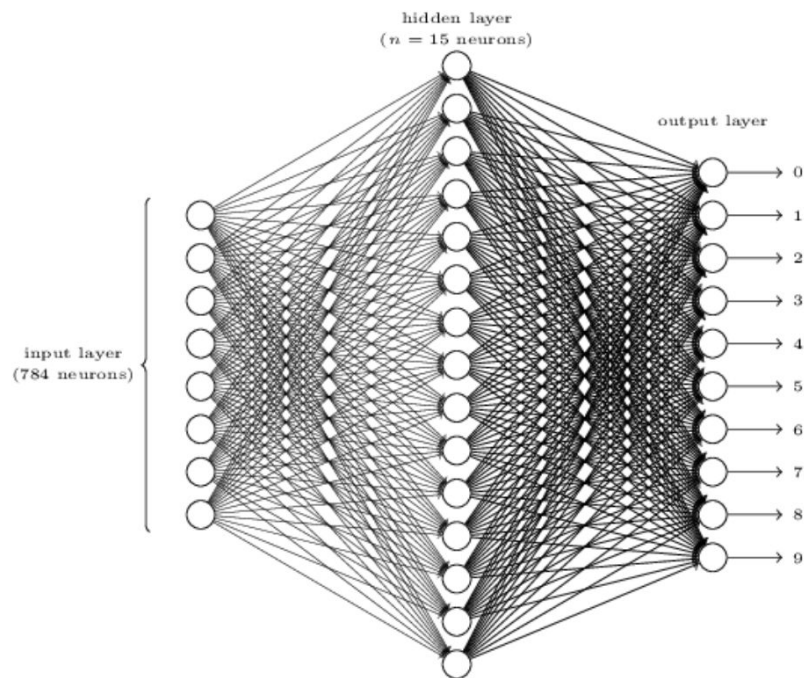
What exactly does a neural net do?

- Learns/ tries to converge on a function which would most closely model training input to label transformation

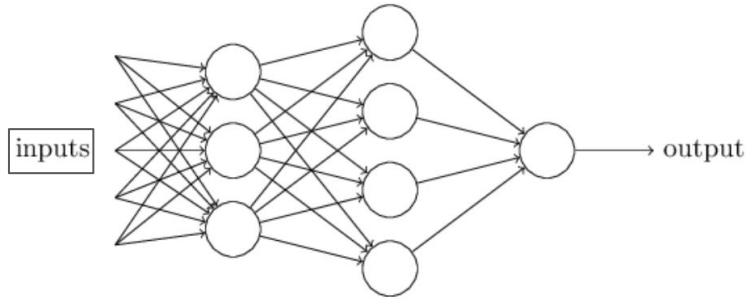
Neural Networks in Action



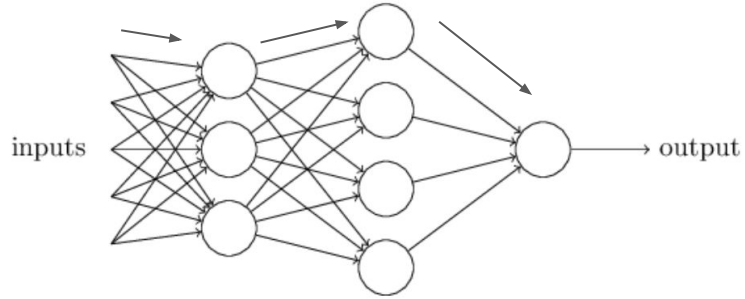
- What are those hidden neurons doing?



Learning: The Big picture

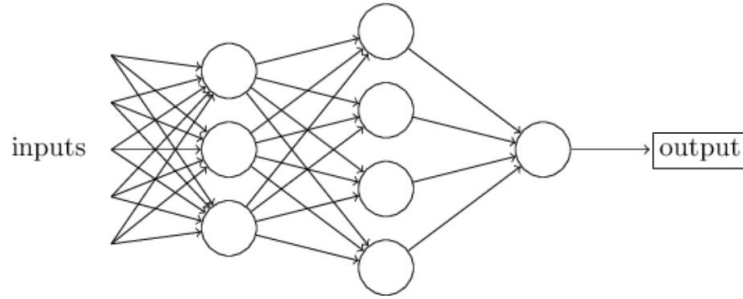


Learning: The Big picture



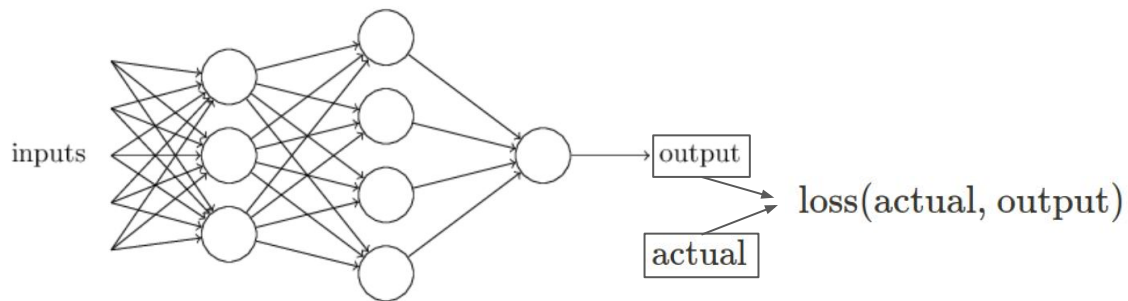
Forward Pass

Learning: The Big picture



Get the output

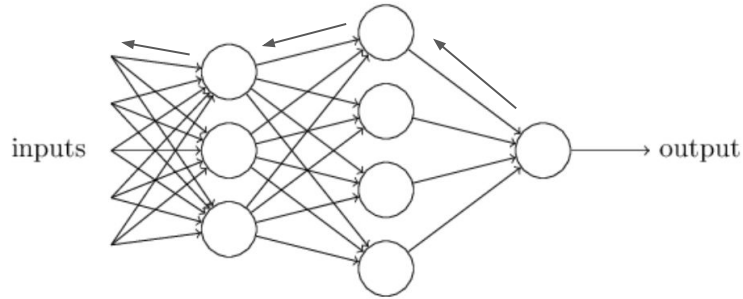
Learning: The Big picture



Loss wrt Label

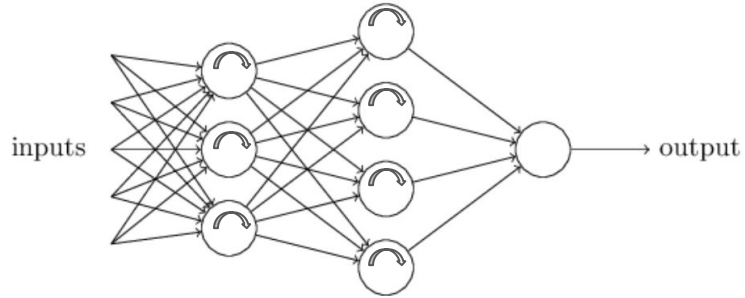
$$\text{cost}(w, b) = \frac{1}{2n} \sum_{\text{all training samples}} ||\text{model}(\text{training sample}) - \text{label}||^2$$

Learning: The Big picture



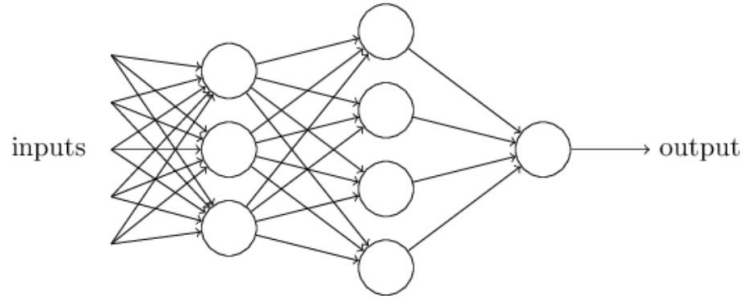
Gradient Calculation: Backpropagation

Learning: The Big picture



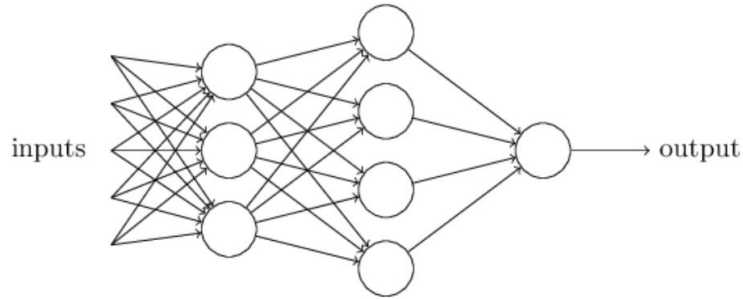
Update “optimise” weights

Learning: The Big picture



Repeat

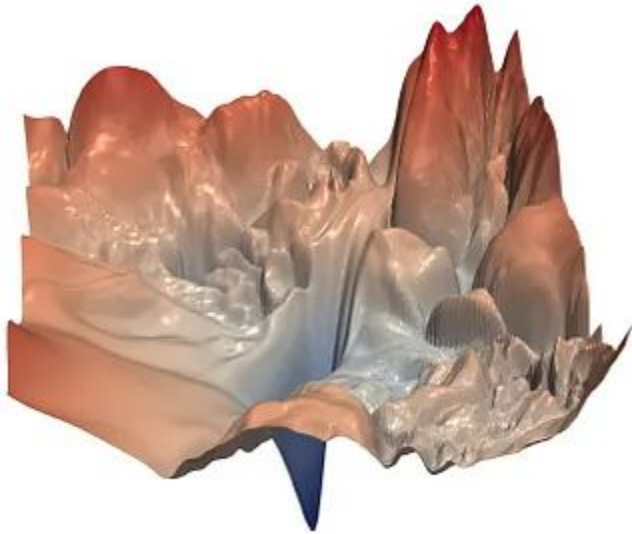
Learning: The Big picture



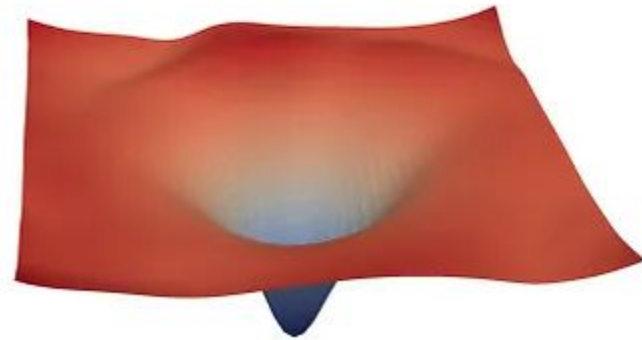
Repeat

One traversal over the entire training dataset = 1 Epoch

Gradient Descent and Loss Landscapes



(a) without skip connections



(b) with skip connections

Gradient Descent

$$\Delta C = \frac{\partial C}{\partial v_1} v_1 + \frac{\partial C}{\partial v_2} v_2$$

Say v_1 and v_2 are the only two learnable parameters in your model

With a small change in v_1 and v_2 , this is the change in Loss we measure

$$\Delta v \equiv (\Delta v_1, \Delta v_2)^T$$

If we are able to measure the gradient of C at a point

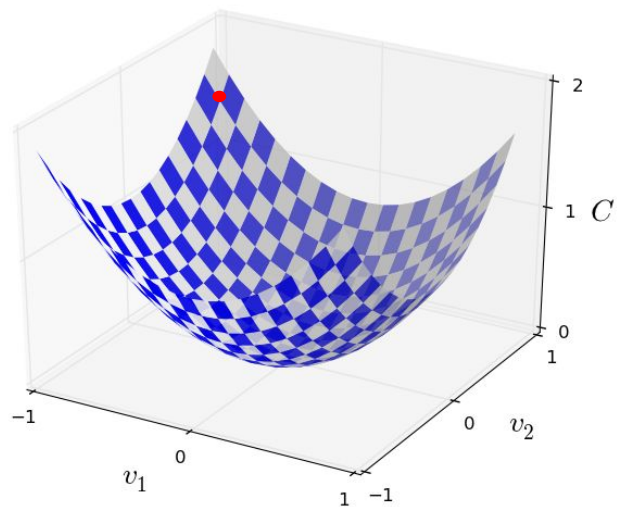
We can figure out what change in the params we should do

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2} \right)^T$$

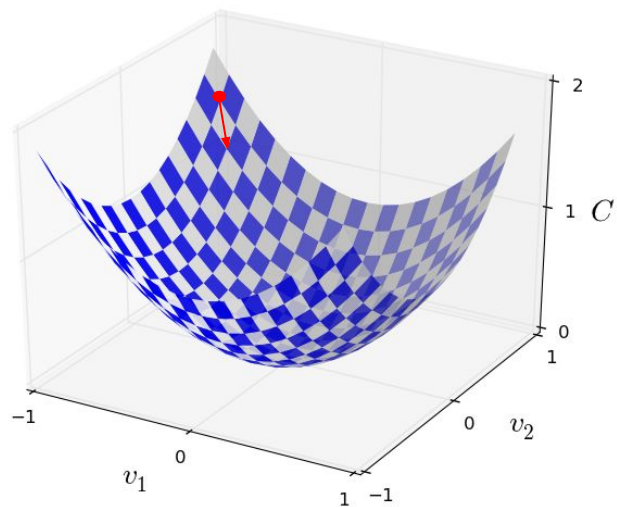
$$\begin{aligned}\Delta v &= -\eta \nabla C \\ v &\rightarrow v - \eta \nabla C\end{aligned}$$

$$\Delta C \approx \nabla C \cdot \Delta v.$$

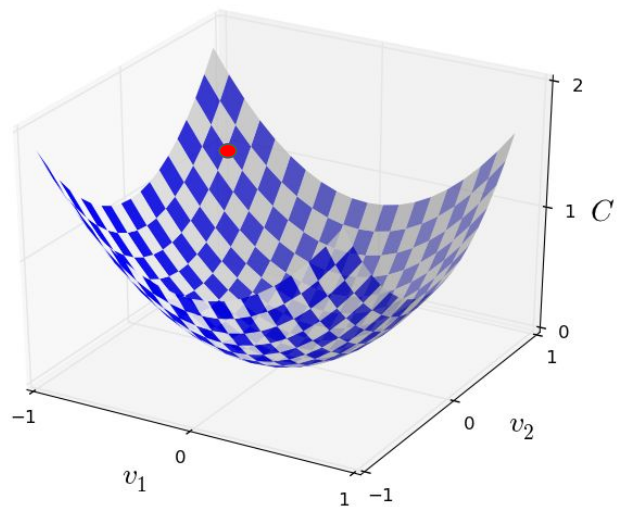
Gradient Descent



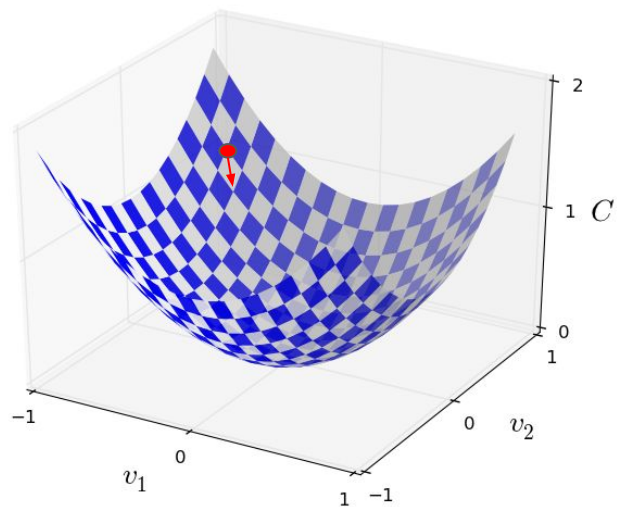
Gradient Descent



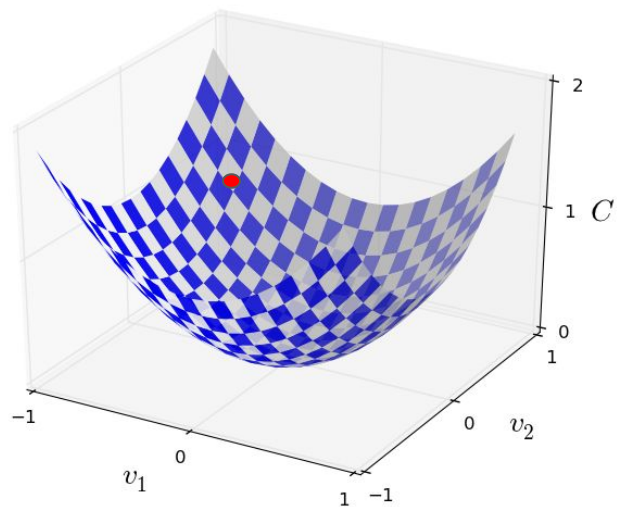
Gradient Descent



Gradient Descent

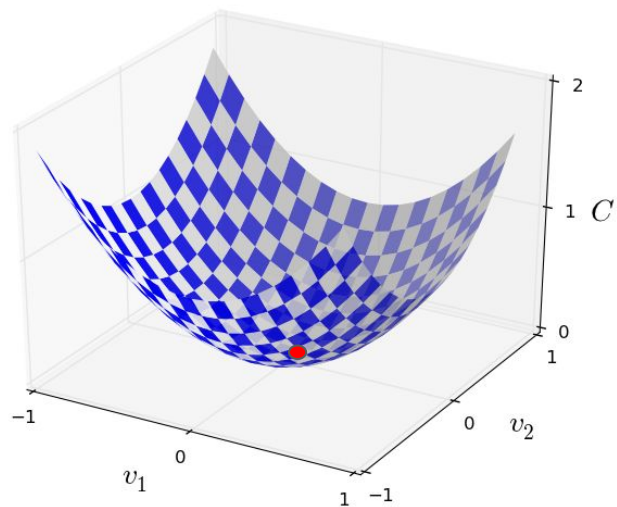


Gradient Descent





Gradient Descent



Gradient Descent

Doing away with the brevity,

The diagram illustrates the gradient descent update equations for weights and biases. It features two equations: $w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$ and $b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$. Annotations with arrows point to specific parts of these equations: 'Learning rate' points to the η in the first equation; '“Effect” of that weight on the loss' points to the $\frac{\partial C}{\partial w_k}$ term in the first equation; 'Old weight' points to the w_k term in the first equation; and an unlabeled arrow points to the $\frac{\partial C}{\partial b_l}$ term in the second equation.

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

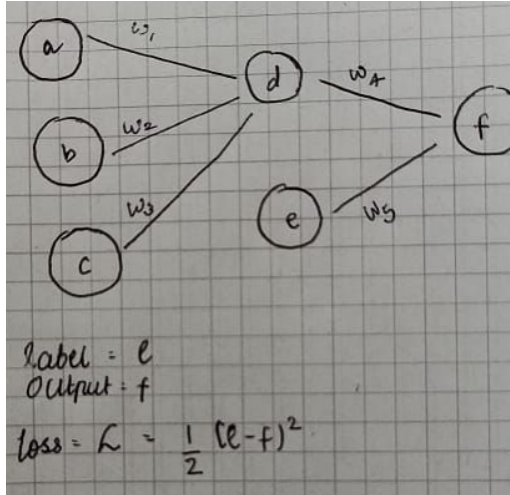
Learning rate

“Effect” of that weight on the loss

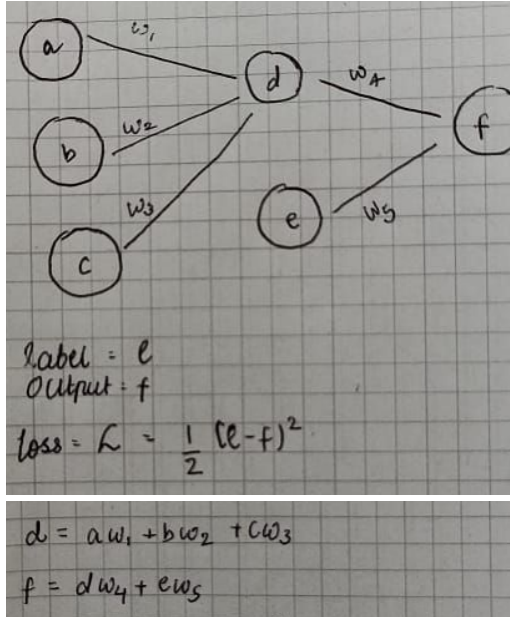
$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

Old weight

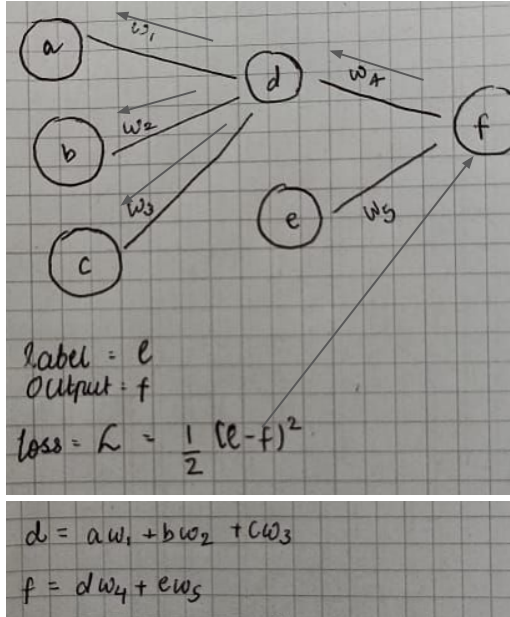
Okay but how is the gradient Calculated?



Okay but how is the gradient Calculated?

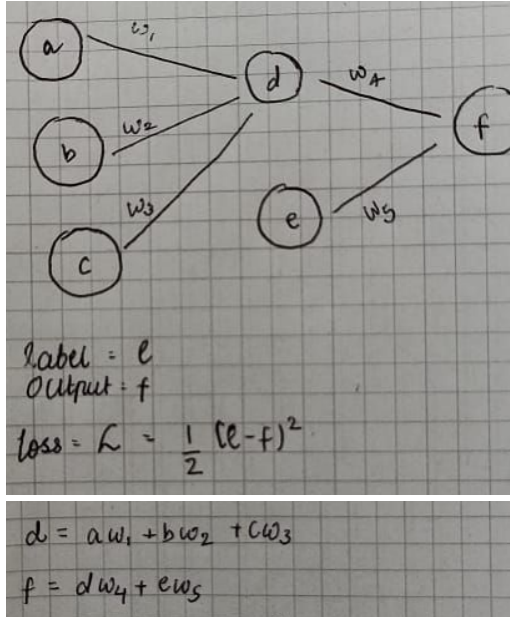


Okay but how is the gradient Calculated?



Chain Rule!

Okay but how is the gradient Calculated?



$$\rightarrow \frac{\partial \mathcal{L}}{\partial w_4} = \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial w_4} = (l - f)(d)$$
$$\rightarrow \frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial w_1} = (l - f)(w_4)(a)$$

Okay but how is the gradient Calculated?

AutoGrad Demo

