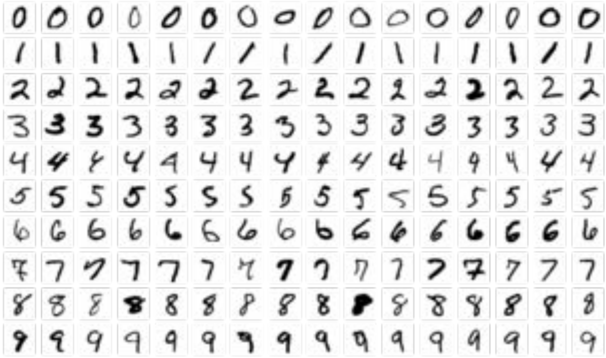
# Perceptrons, Backprop, and Gradient Descent

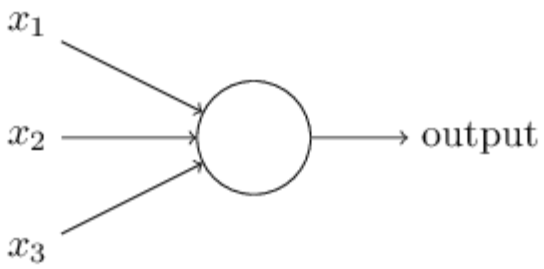
RRC Summer School ‘23



# Perceptron

Takes in several binary inputs

Gives out one binary output

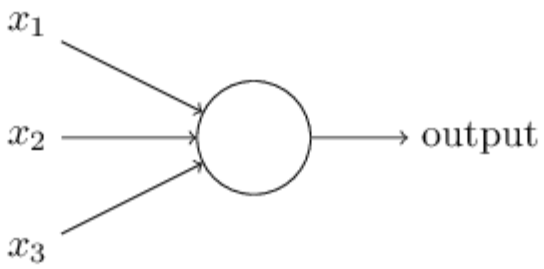


# Perceptron

Takes in several binary inputs

Gives out one binary output

- How does it know what to output?

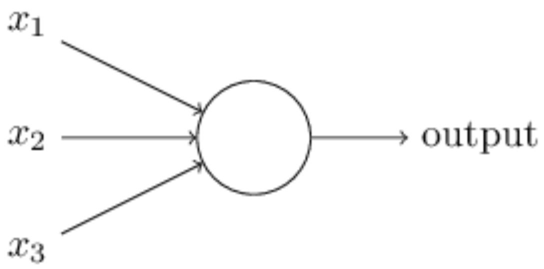


# Perceptron

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

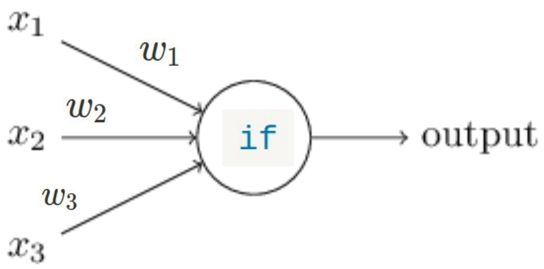


# Perceptron

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



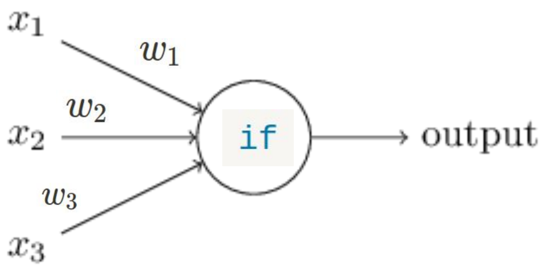

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

# Perceptron

Takes in several binary inputs

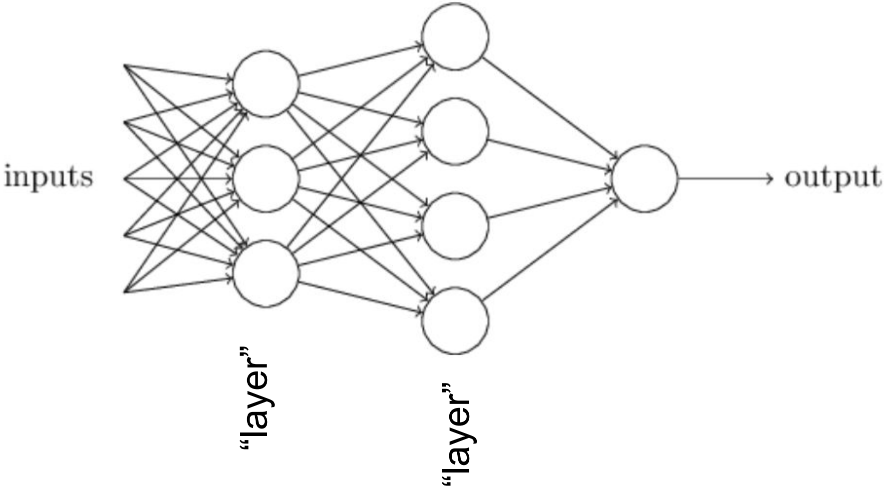
Gives out one binary output

- 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

Stay lull

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

Fired

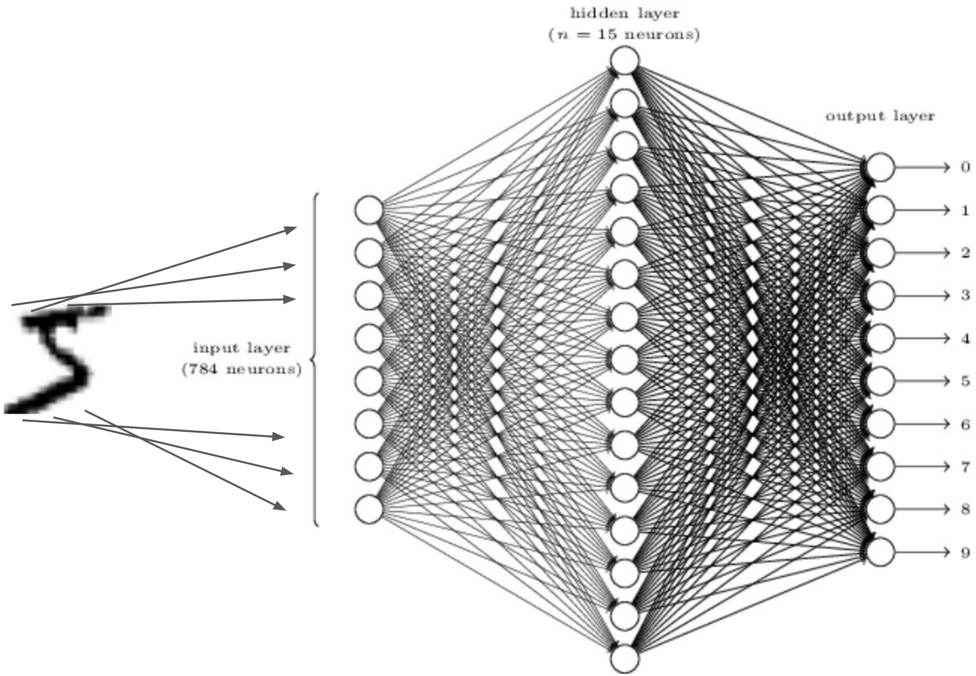
# Learning

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

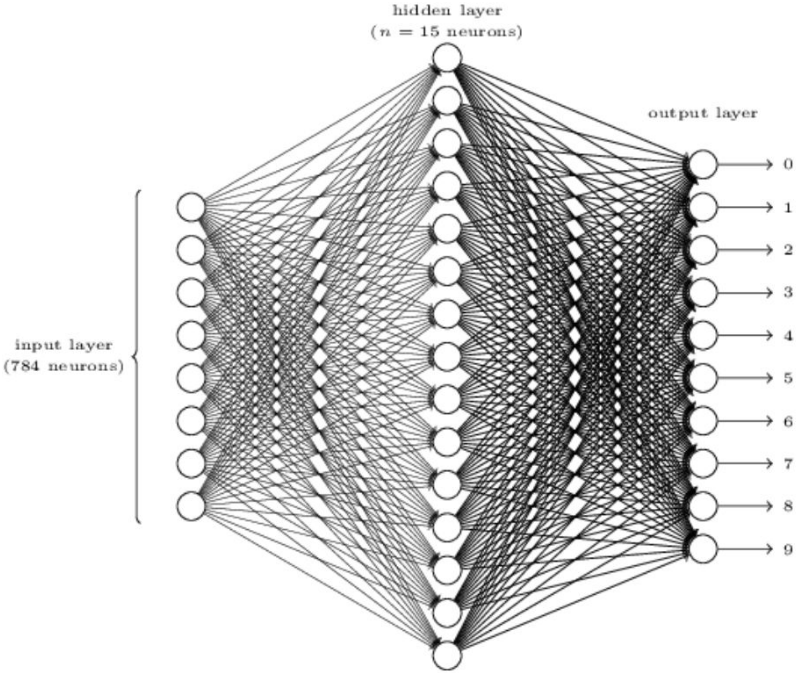
What exactly does a neural net do?

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

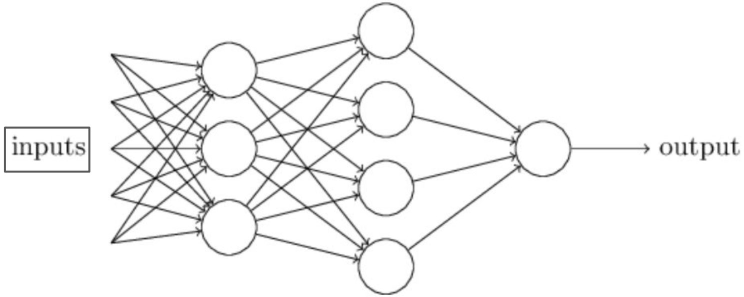
# Neural Networks in Action



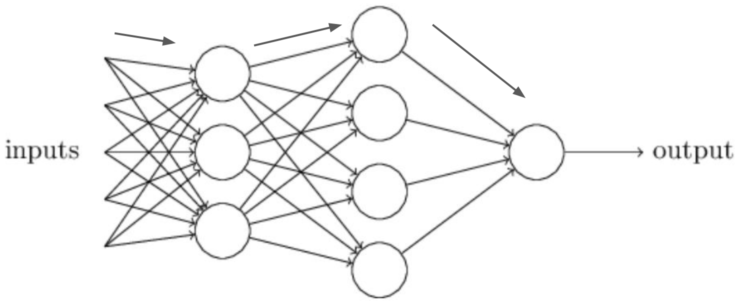
- What are those hidden neurons doing?



# Learning: The Big picture

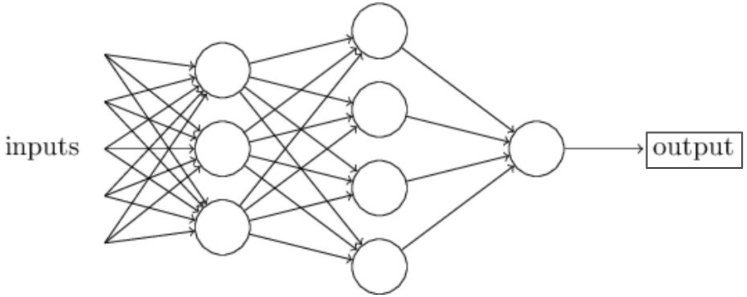


# Learning: The Big picture



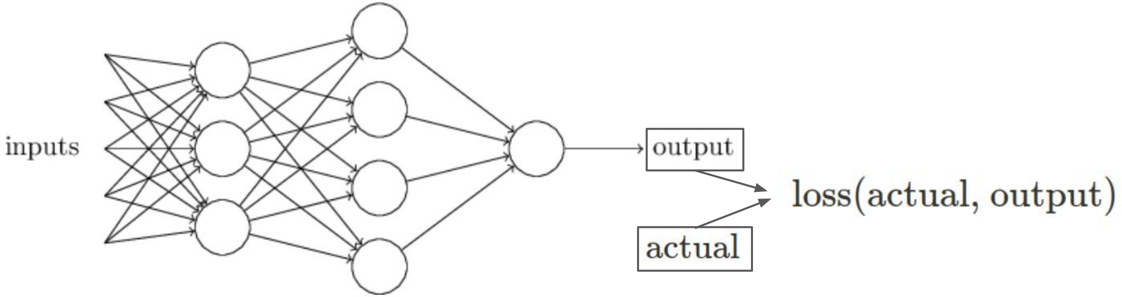

Forward Pass

# Learning: The Big picture




Get the output

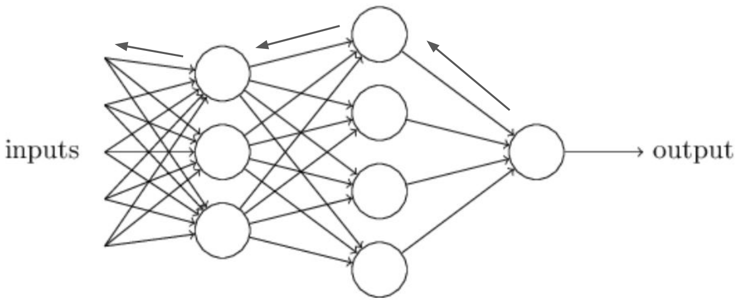
# Learning: The Big picture




## Loss wrt Label

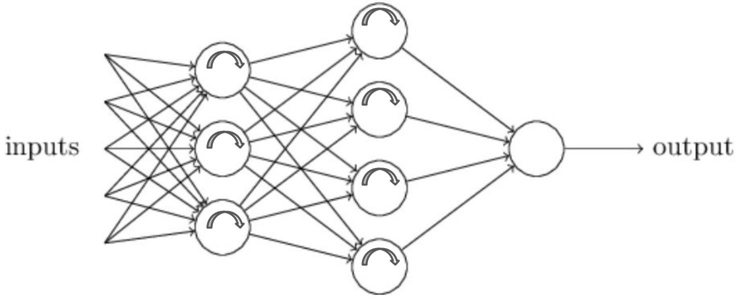
$$\text{cost}(w,b) = \frac{1}{2n} \sum_{\textit{all training samples}} ||\text{model}(\textit{training sample}) - \textit{label}||^2$$

# Learning: The Big picture



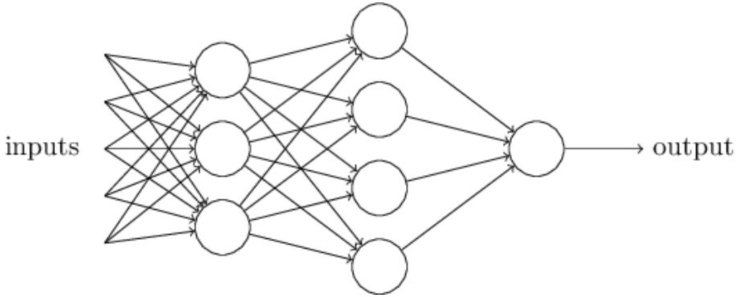

Gradient Calculation: Backpropagation

# Learning: The Big picture



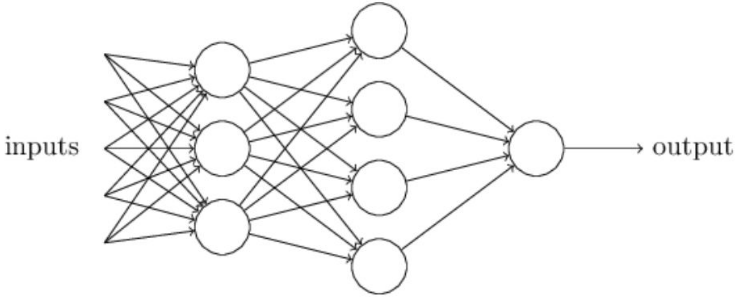

Update “optimise” weights

# Learning: The Big picture




Repeat

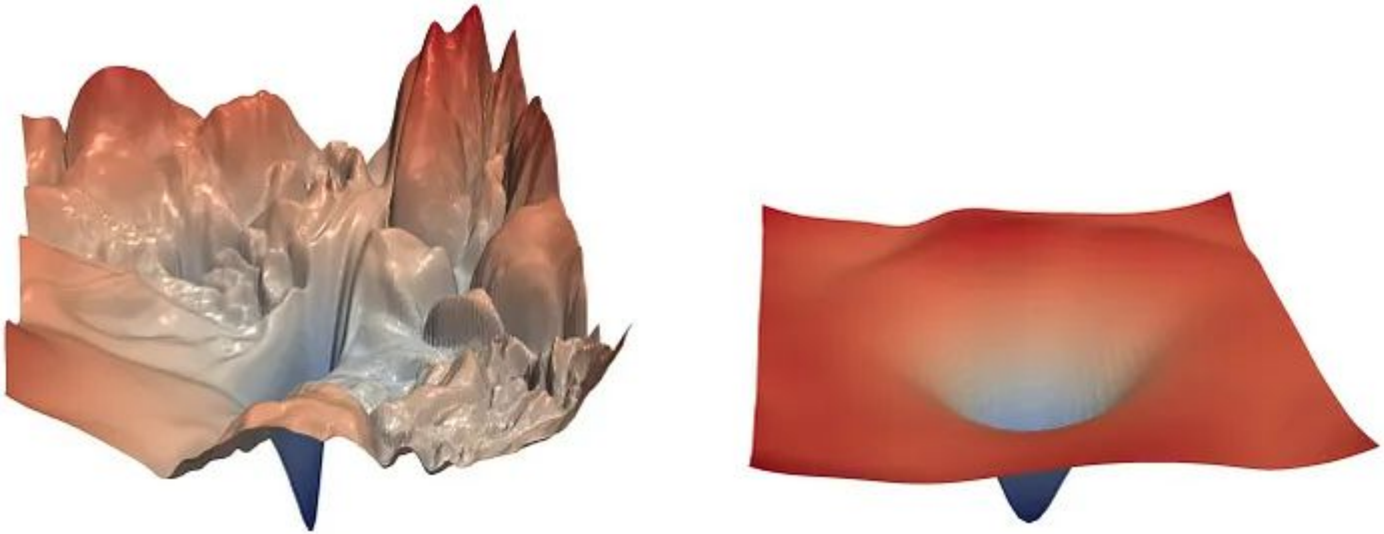
# Learning: The Big picture




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 v1 and v2 are the only two learnable parameters in your model

With a small change in v1 and v2, 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

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

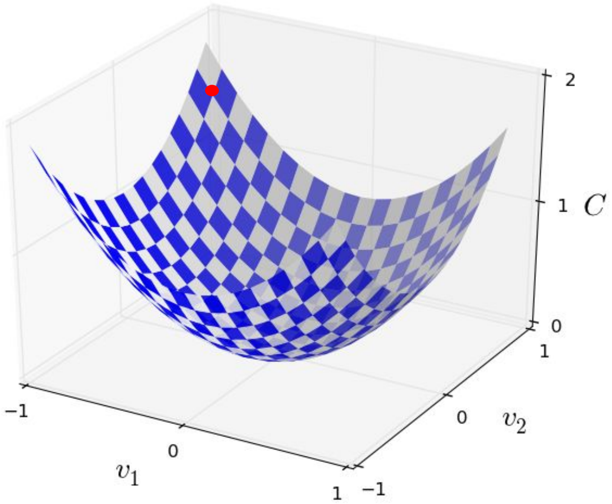
We can figure out what change in the params we should do

$$\Delta v = -\eta \nabla C$$

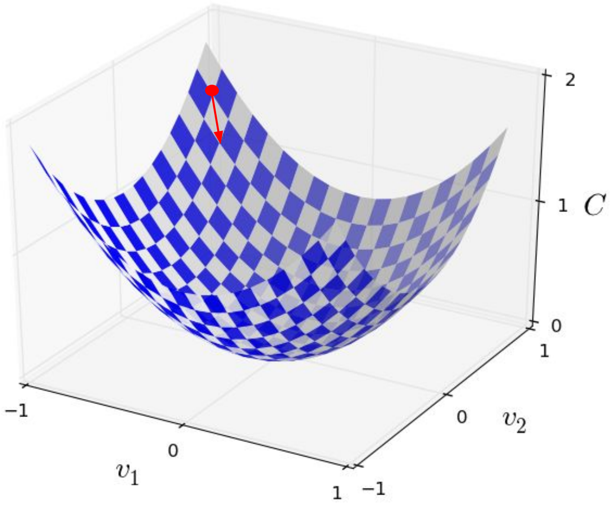
$$v \rightarrow v - \eta \nabla C$$

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

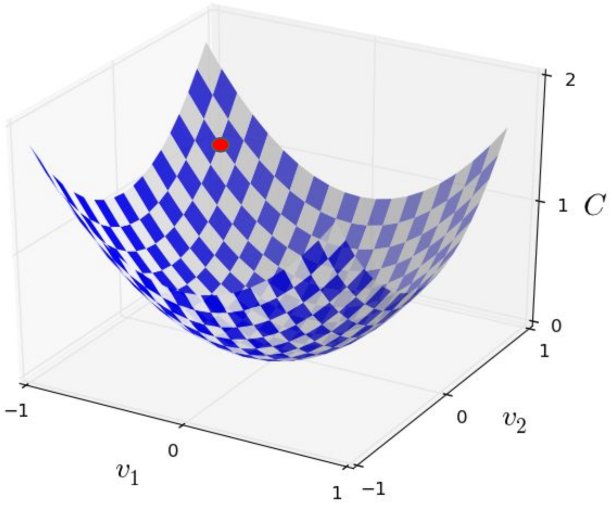
# Gradient Descent



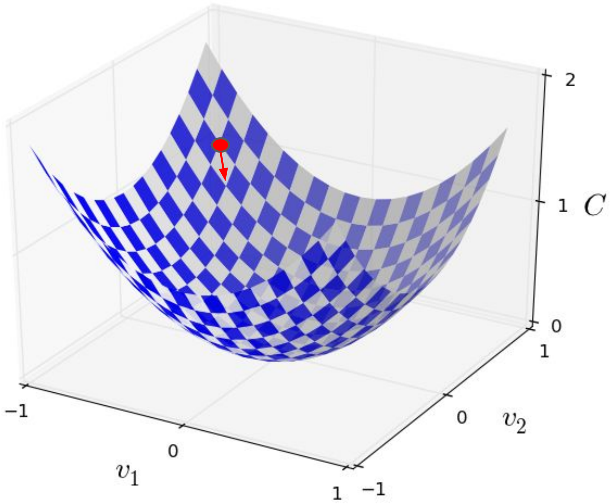
# Gradient Descent



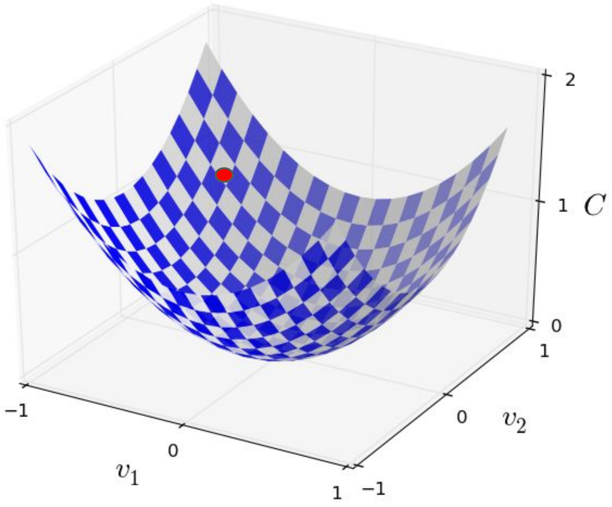
# Gradient Descent



# Gradient Descent

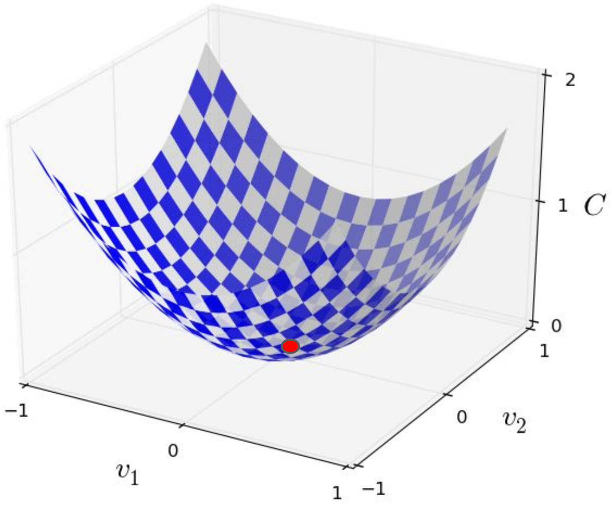


# Gradient Descent




TWO HOURS LATER

# Gradient Descent



# Gradient Descent

Doing away with the brevity,

Learning rate

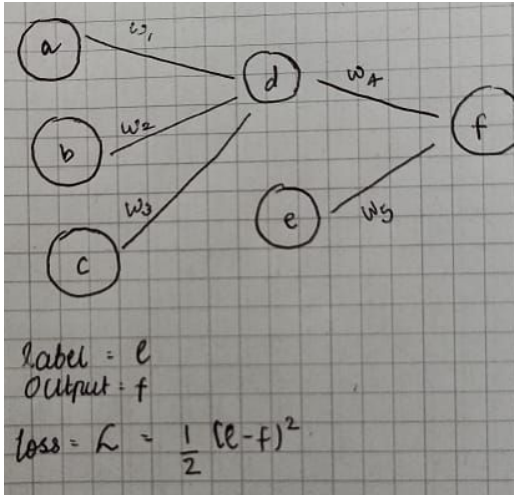
$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

“Effect” of that weight on the loss

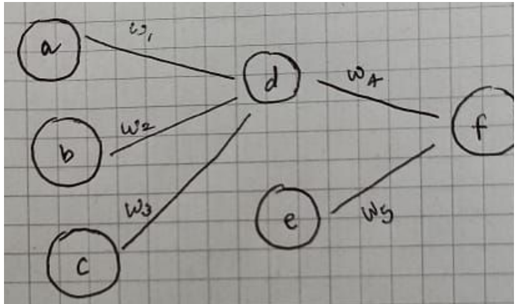
$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$

Old weight

# Okay but how is the gradient Calculated?



# Okay but how is the gradient Calculated?



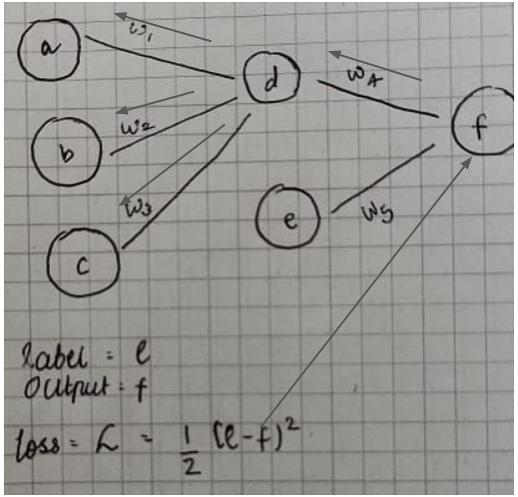

label = e
output = f

loss = $L = \frac{1}{2}(e-f)^2$

$d = aw_1 + bw_2 + cw_3$

$f = dw_4 + ew_5$

# Okay but how is the gradient Calculated?

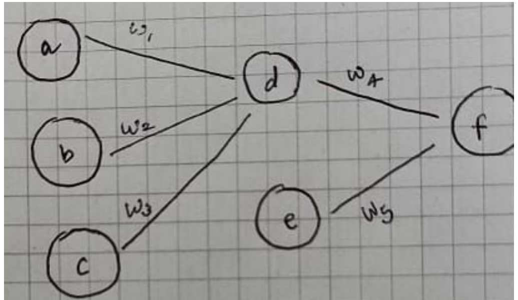



$d = aw_1 + bw_2 + cw_3$

$f = dw_4 + ew_5$

Chain Rule!

# Okay but how is the gradient Calculated?




label = $l$
Output = $f$

loss = $L = \frac{1}{2}(l-f)^2$

$d = aw_1 + bw_2 + cw_3$

$f = dw_4 + ew_5$

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

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

# Okay but how is the gradient Calculated?

AutoGrad Demo