بسم الله الرحمن الرحيم

In the Name of Allah, the Most Gracious, the Most Merciful

قال الم ۱۶ — طٰہٰ ۲۰

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ﴿۲۵﴾

۲۵۔ (موسیٰ علیہ السلام نے) عرض کیا: اے میرے رب! میرے لئے میرا سینہ کشادہ فرما دے۔

وَيَسِّرْ لِيْٓ اَمْرِيْ ﴿۲۶﴾

۲۶۔ اور میرا کار (رسالت) میرے لئے آسان فرما دے۔

وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ﴿۲۷﴾

۲۷۔ اور میری زبان کی گرہ کھول دے۔

يَفْقَهُوْا قَوْلِيْ ﴿۲۸﴾

۲۸۔ کہ لوگ میری بات (آسانی سے) سمجھ سکیں۔

Surah Taha with Urdu Translation

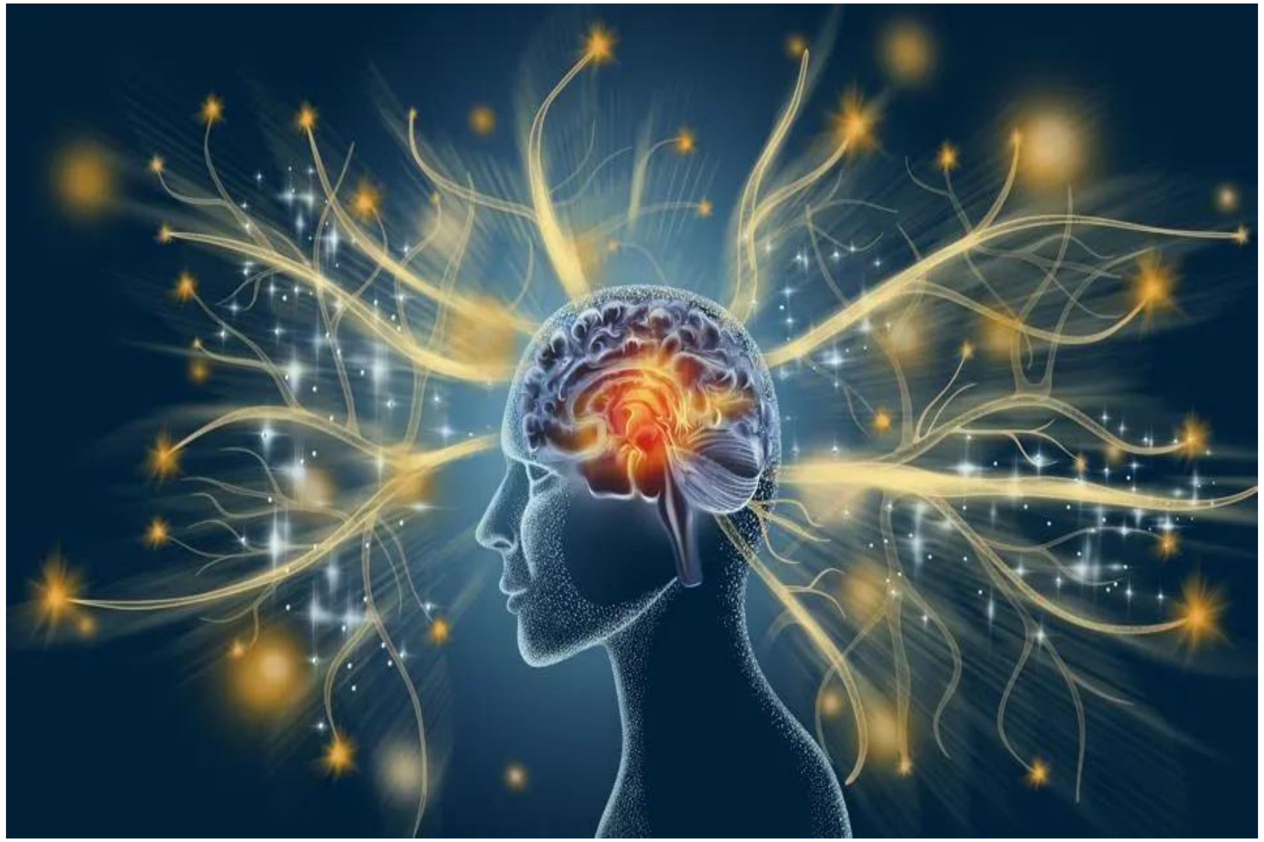
رَبِّ إِنِّي لِمَا أَنْزَلْتَ إِلَيَّ مِنْ خَيْرٍ فَقِيرٌ

اے میرے رب! بے شک میں، جو بھلائی
بھی تو میری طرف نازل فرمائے، اس کا محتاج ہوں۔

﴿سورۃ القصص: آیت نمبر 24﴾

CS4152: Deep Learning and Neural Networks

Lecture 4 (Neural Networks Training and Optimization)



Dr. Jameel Ahmad

Associate Professor

Jameel.ahmad@umt.edu.pk

Department of Computer Science

School of Systems and Technology

University of Management and Technology, Lahore

Fall 2025

Neural Networks

A Deeper Look

-Computational Graphs

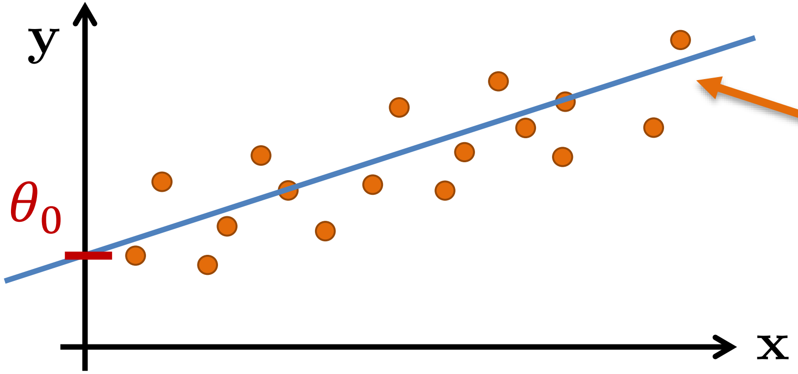
-Loss Functions

-Gradient Descent Optimization

Linear Regression

= a **supervised learning** method to find a **linear model** of the form

$$\hat{y}_i = \theta_0 + \sum_{j=1}^{d} x_{ij}\theta_j = \theta_0 + x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$



Goal: find a model that explains a target y given the input x

Logistic Regression

- Loss function

$$\mathcal{L}(\hat{y}_i, y_i) = -[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

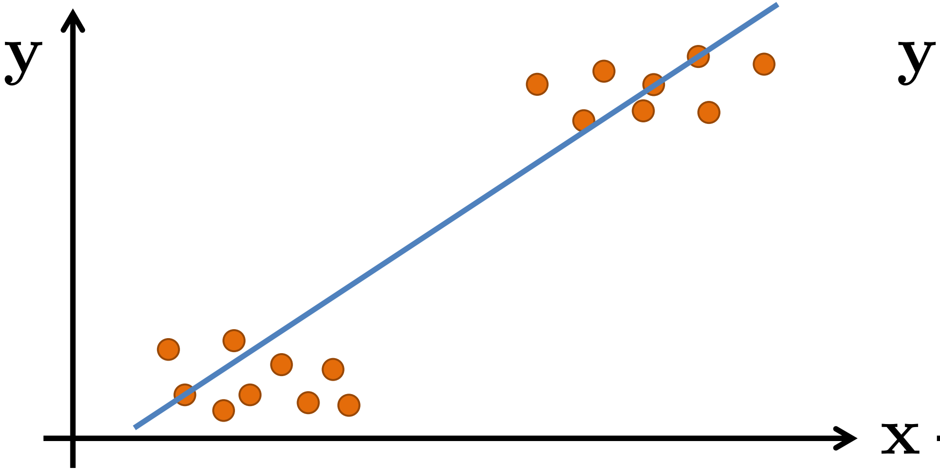
- Cost function

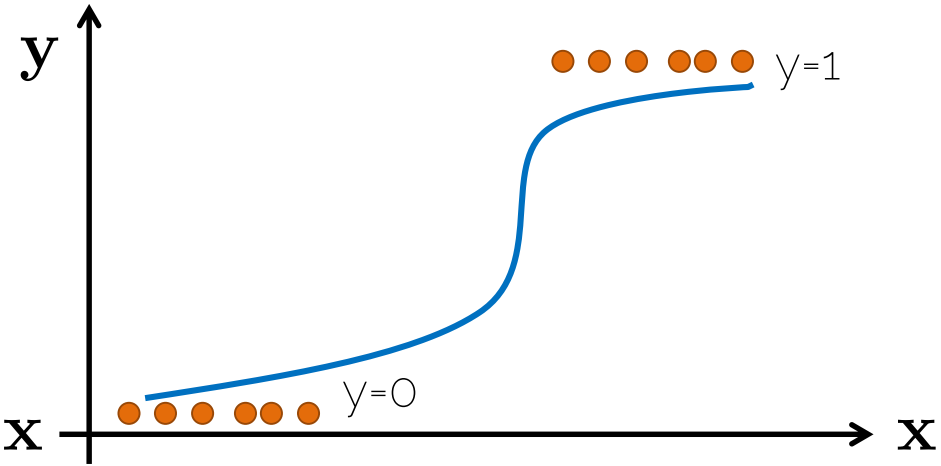
$$\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log \widehat{y}_i + (1 - y_i) \cdot \log[1 - \widehat{y}_i])$$

Minimization

$$\widehat{y}_i = \sigma(x_i\boldsymbol{\theta})$$

Linear vs Logistic Regression

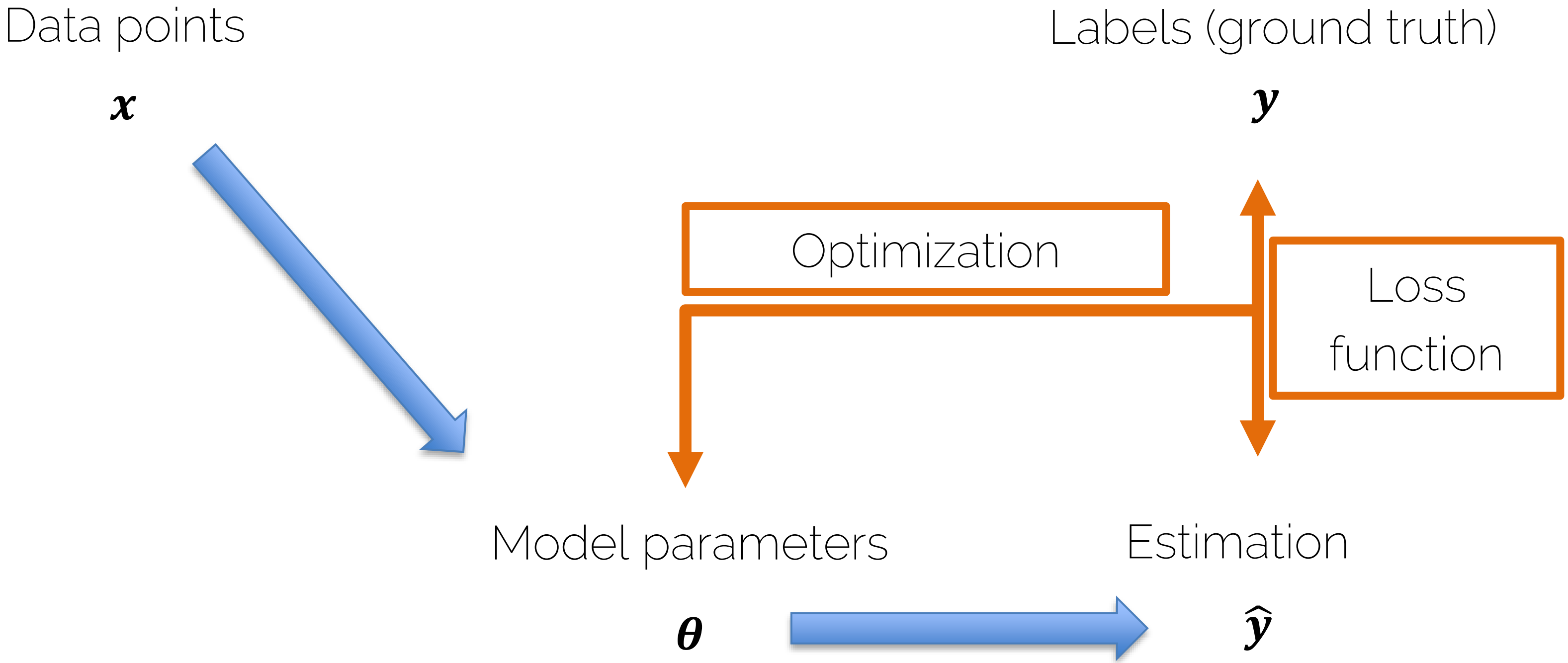




Predictions can exceed the range of the training samples
→ in the case of classification [0;1] this becomes a real issue

Predictions are guaranteed to be within [0;1]

How to obtain the Model?



Linear Score Functions

- Linear score function as seen in linear regression

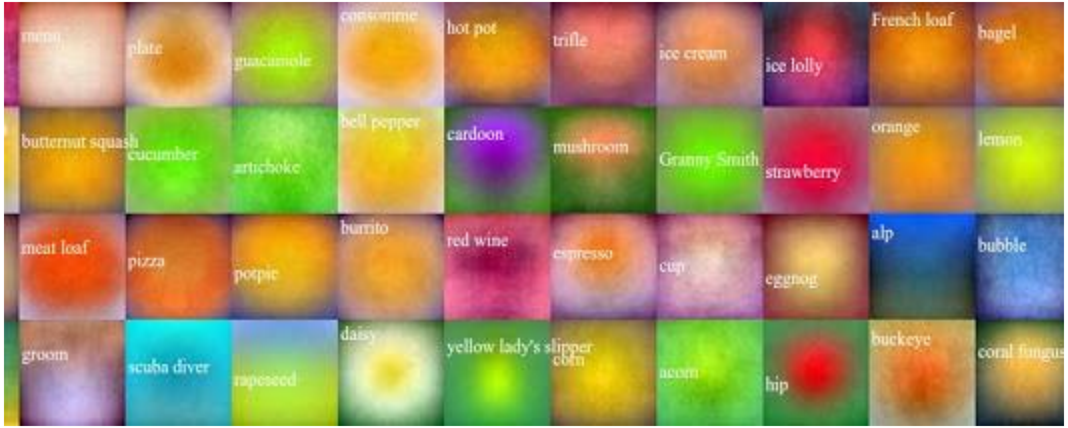
$$\boldsymbol{f_i} = \sum_{\boldsymbol{j}} w_{\boldsymbol{i,j}}\, x_{\boldsymbol{j}}$$

$$\boldsymbol{f} = \boldsymbol{W}\,\boldsymbol{x} \qquad \text{(Matrix Notation)}$$

Linear Score Functions on Images

- Linear score function $\boldsymbol{f} = \boldsymbol{W}\boldsymbol{x}$

On CIFAR-10

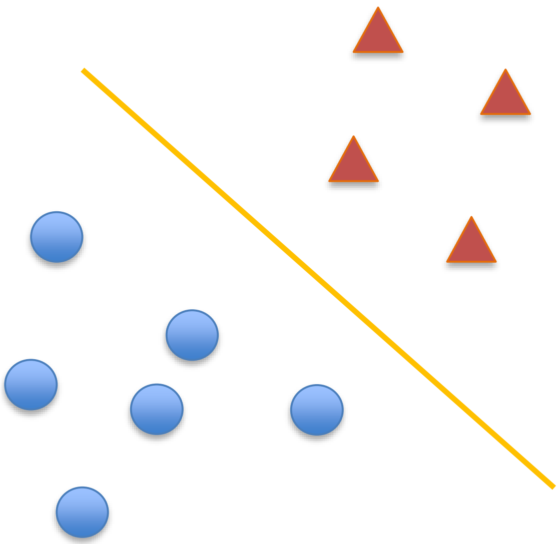


On ImageNet

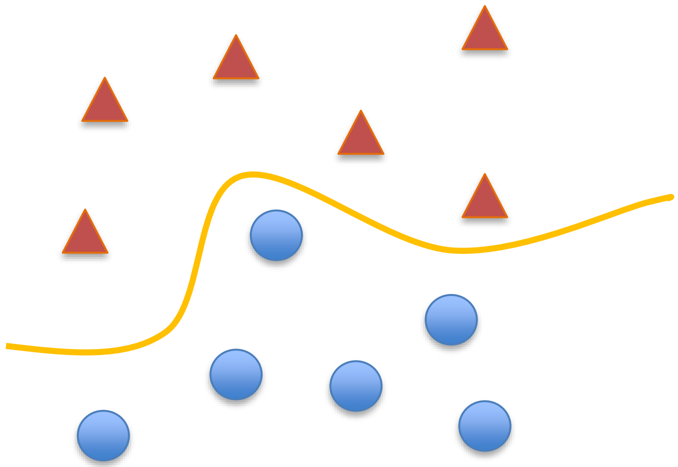
Source:: Li/Karpathy/Johnson

Linear Score Functions?

Logistic Regression



Linear Separation Impossible!



Linear Score Functions?

- Can we make linear regression better?
  - Multiply with another weight matrix $\boldsymbol{W_2}$

$$\hat{\boldsymbol{f}} = \boldsymbol{W_2} \cdot \boldsymbol{f}$$
$$\hat{\boldsymbol{f}} = \boldsymbol{W_2} \cdot \boldsymbol{W} \cdot \boldsymbol{x}$$

- Operation is still linear.

$$\widehat{\boldsymbol{W}} = \boldsymbol{W_2} \cdot \boldsymbol{W}$$
$$\hat{\boldsymbol{f}} = \widehat{\boldsymbol{W}}\, \boldsymbol{x}$$

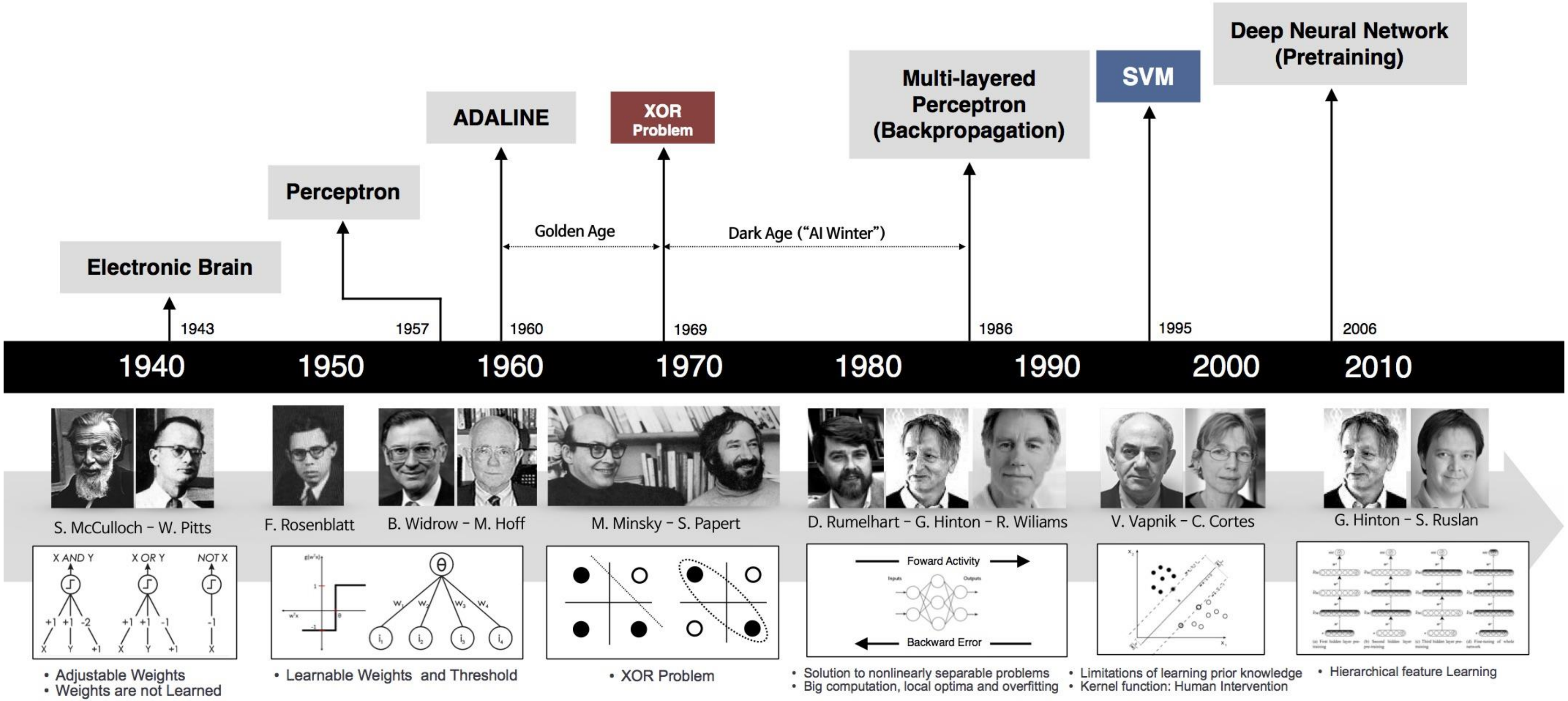
- Solution → add non-linearity!!

Neural Network

- Linear score function $\boldsymbol{f} = \boldsymbol{W}\boldsymbol{x}$

- Neural network is a nesting of 'functions'
  - 2-layers: $\boldsymbol{f} = \boldsymbol{W_2} \max(\mathbf{0}, \boldsymbol{W_1}\boldsymbol{x})$
  - 3-layers: $\boldsymbol{f} = \boldsymbol{W_3} \max(\mathbf{0}, \boldsymbol{W_2} \max(\mathbf{0}, \boldsymbol{W_1}\boldsymbol{x}))$
  - 4-layers: $\boldsymbol{f} = \boldsymbol{W_4} \tanh\,(\boldsymbol{W_3}, \max(\mathbf{0}, \boldsymbol{W_2} \max(\mathbf{0}, \boldsymbol{W_1}\boldsymbol{x})))$
  - 5-layers: $\boldsymbol{f} = \boldsymbol{W_5}\sigma(\boldsymbol{W_4} \tanh(\boldsymbol{W_3}, \max(\mathbf{0}, \boldsymbol{W_2} \max(\mathbf{0}, \boldsymbol{W_1}\boldsymbol{x}))))$
  - … up to hundreds of layers

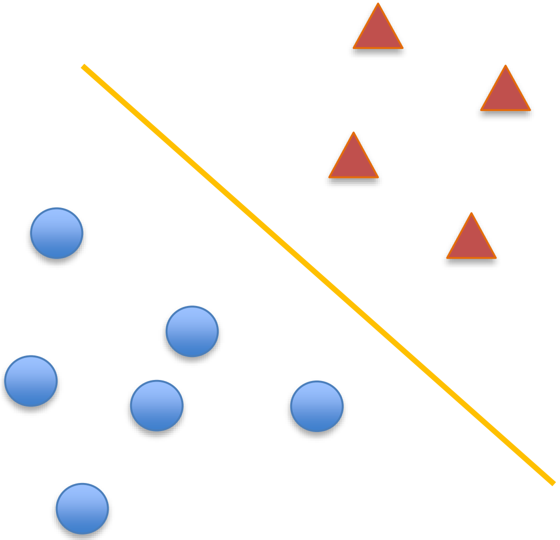
History of Neural Networks



Source: http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html

Neural Network

Logistic Regression

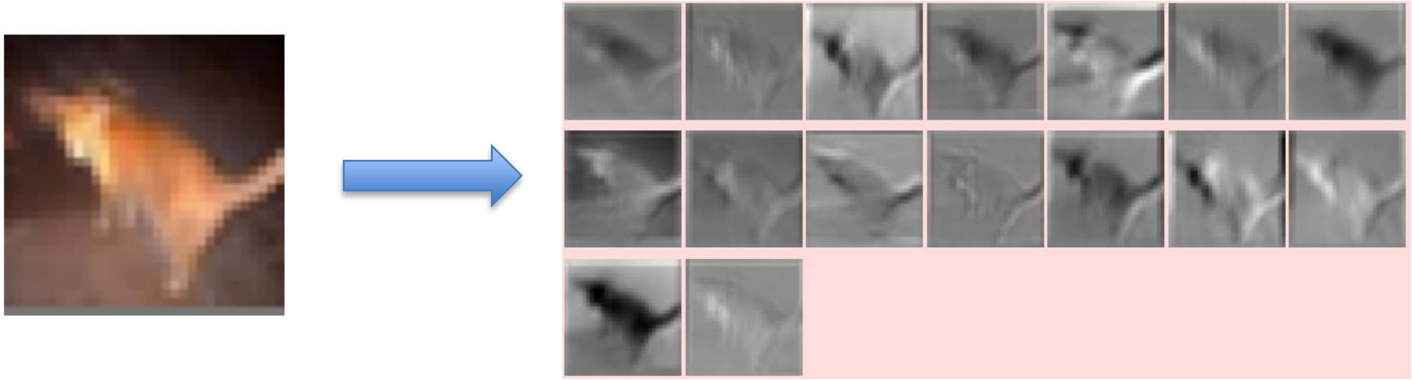


Neural Networks

Neural Network

- Non-linear score function $\boldsymbol{f} = \dots(\max(\mathbf{0}, \boldsymbol{W}_1\boldsymbol{x}))$

On CIFAR-10



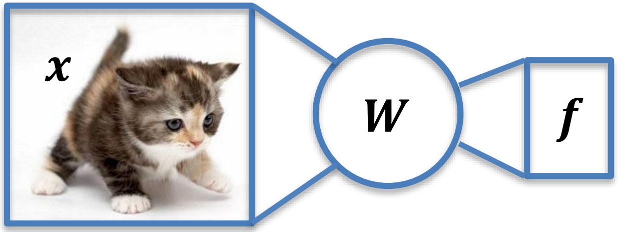
Visualizing activations of the first layer.

Source: ConvNetJS

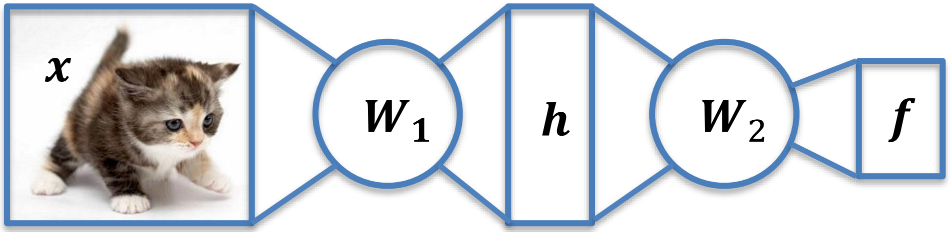
Neural Network

1-layer network: $\boldsymbol{f} = \boldsymbol{W}\boldsymbol{x}$

2-layer network: $\boldsymbol{f} = \boldsymbol{W}_{\mathbf{2}} \max(\mathbf{0}, \boldsymbol{W}_{\mathbf{1}}\boldsymbol{x})$



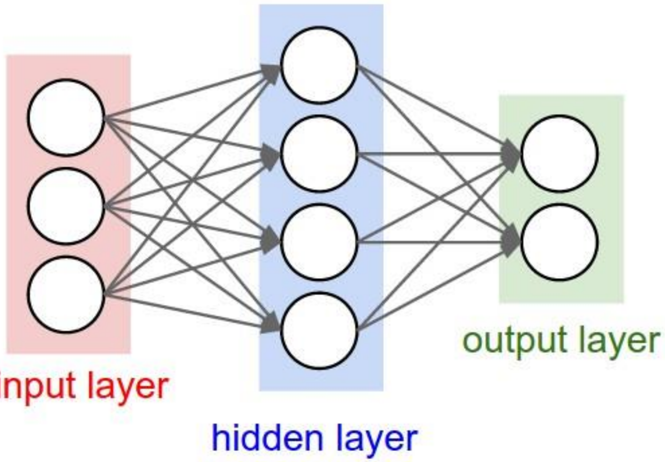
$128 \times 128 = 16384$ $\quad$ 10



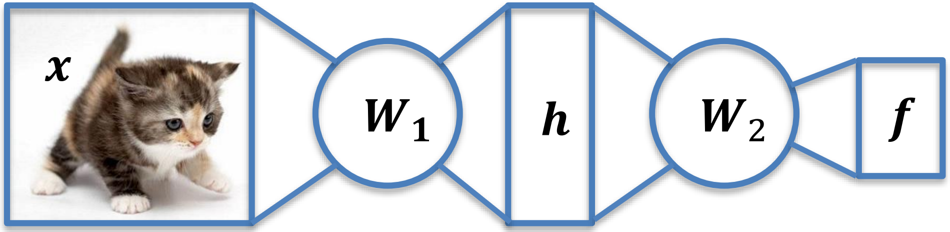
$128 \times 128 = 16384$ $\quad$ 1000 $\quad$ 10

Why is this structure useful?

Neural Network



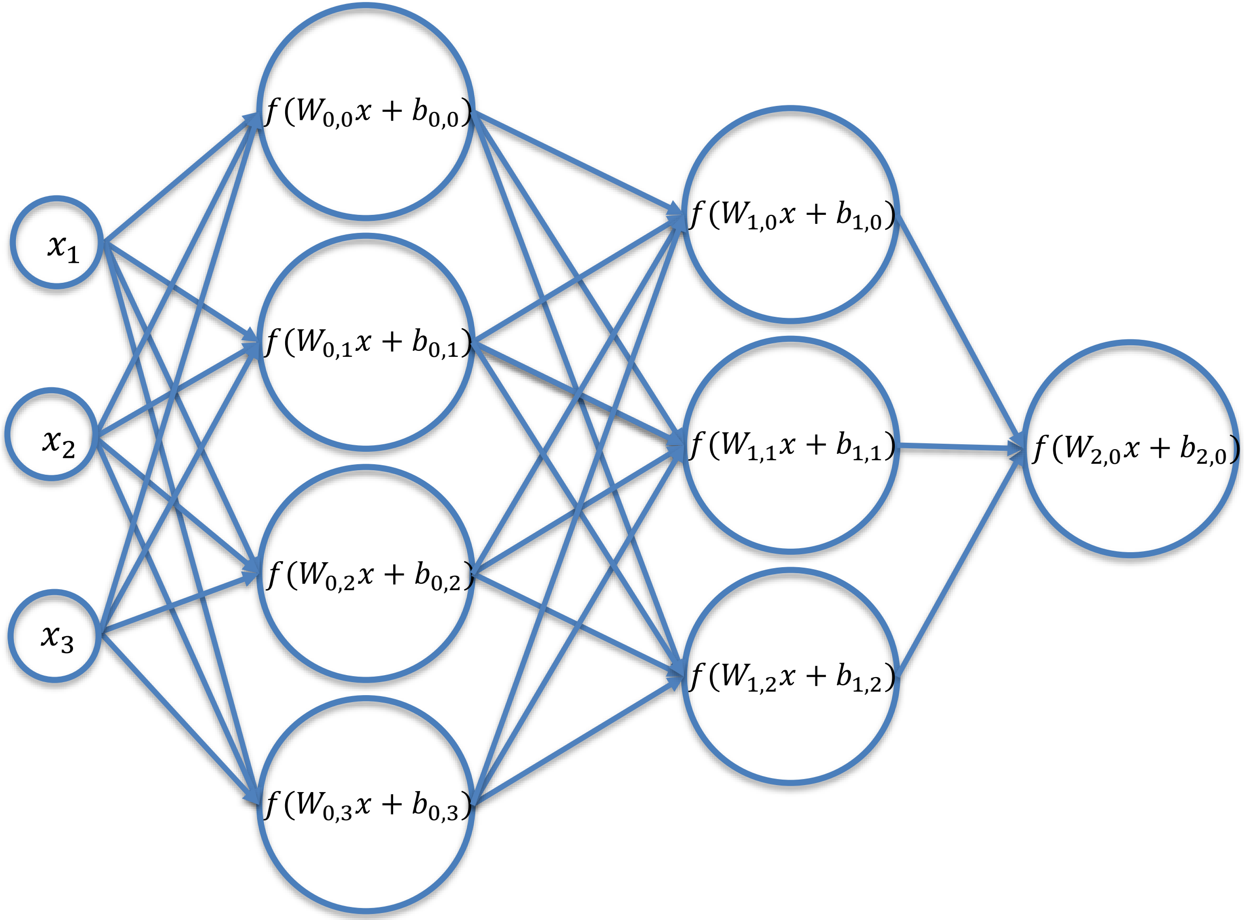
2-layer network: $\boldsymbol{f} = \boldsymbol{W_2} \max(\boldsymbol{0}, \boldsymbol{W_1 x})$



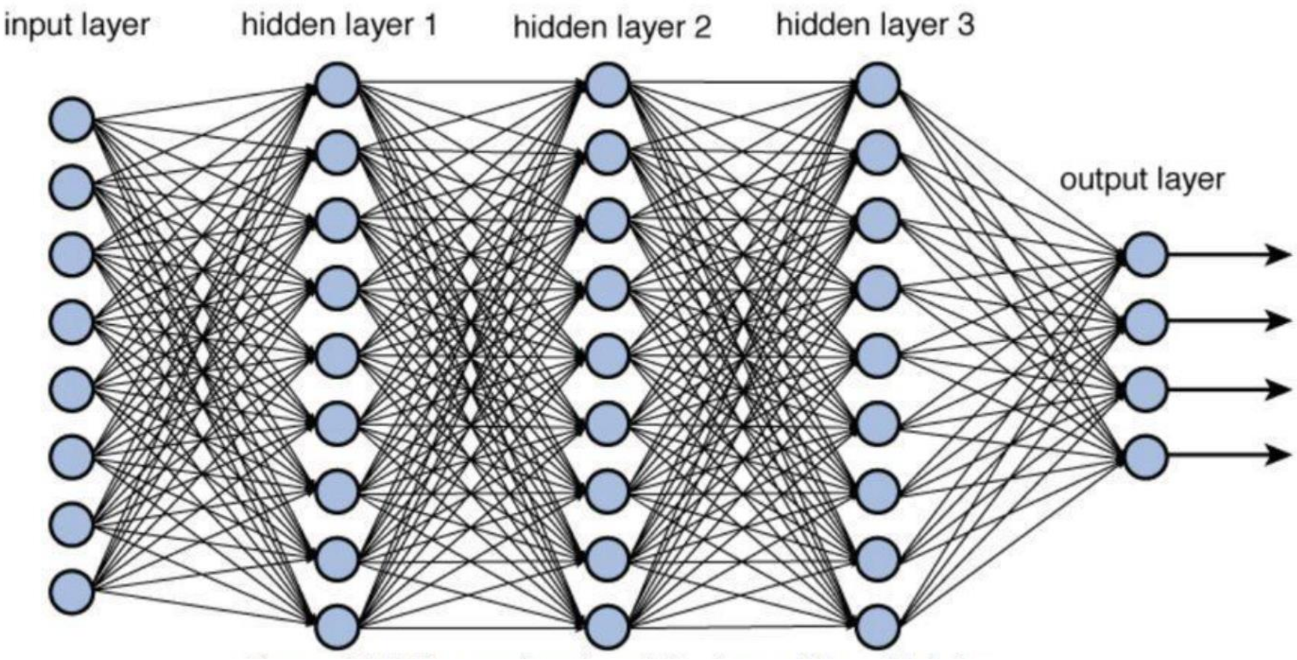
$128 \times 128 = 16384$ | 1000 | 10

Input Layer | Hidden Layer | Output Layer

Net of Artificial Neurons



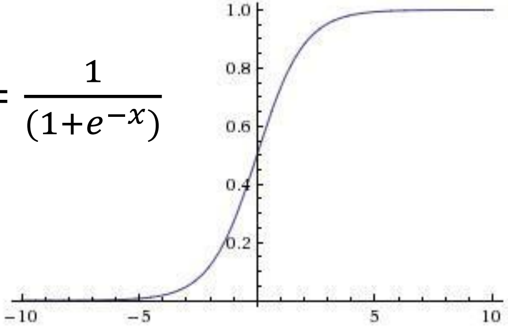
Neural Network



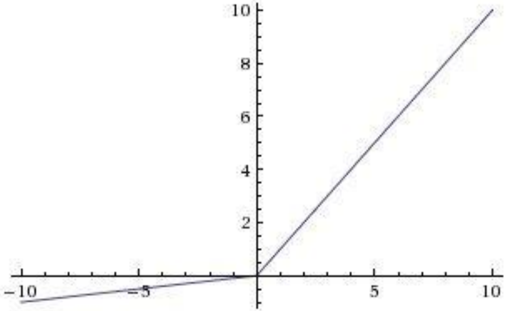
Source: https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

Activation Functions

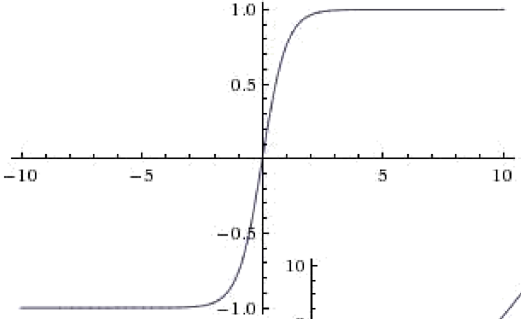
Sigmoid: $\boldsymbol{\sigma(x) = \frac{1}{(1+e^{-x})}}$



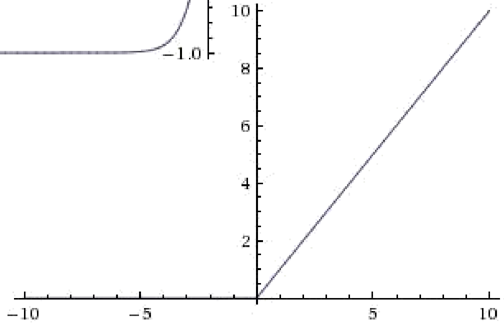
Leaky ReLU: $\mathbf{max}(0.1x, x)$



tanh: $\mathbf{tanh}(x)$



ReLU: $\mathbf{max}(0, x)$



Parametric ReLU: $\mathbf{max}(\alpha x, x)$

Maxout $\mathbf{max}(w_1^T x + b_1, w_2^T x + b_2)$

ELU $\mathrm{f(x)} = \begin{cases} x & \text{if } x > 0 \\ \alpha(\mathrm{e}^x - 1) & \text{if } x \le 0 \end{cases}$

Neural Network

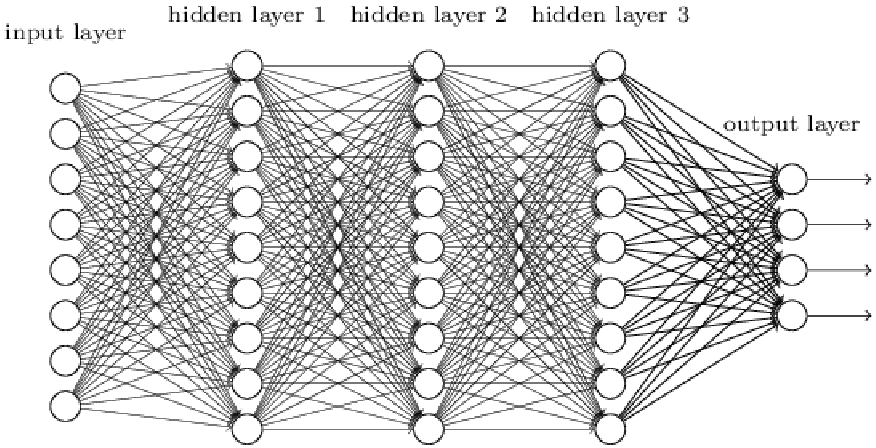
$$\boldsymbol{f} = \boldsymbol{W_3} \cdot (\boldsymbol{W_2} \cdot (\boldsymbol{W_1} \cdot \boldsymbol{x}))$$

Why activation functions?

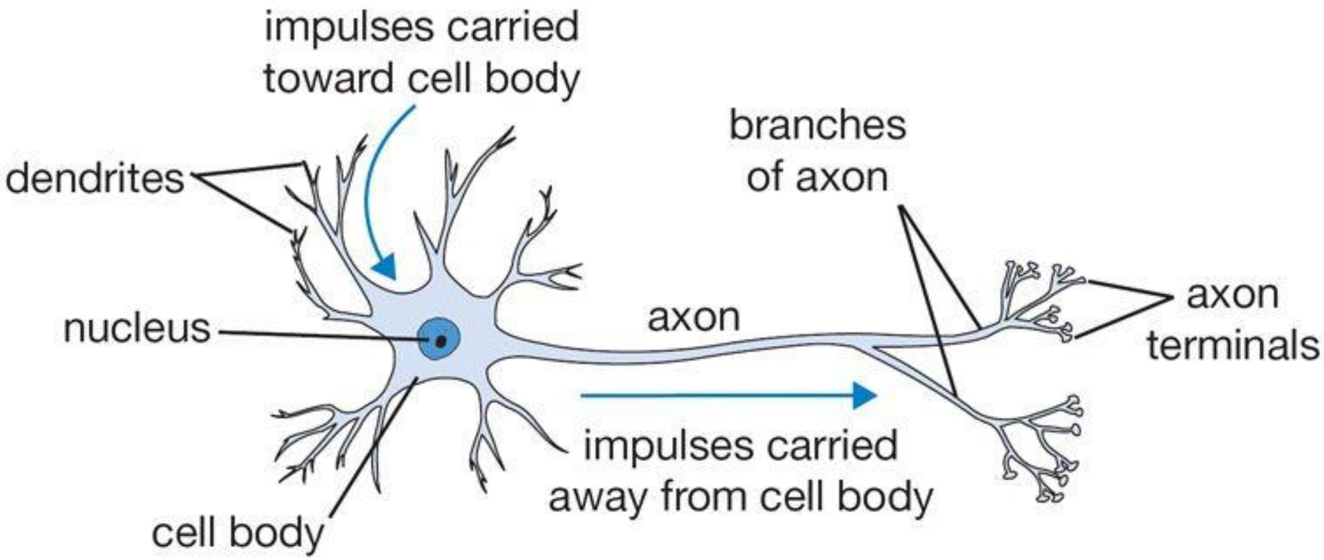
Simply concatenating linear layers would be so much cheaper...

Neural Network

Why organize a neural network into layers?

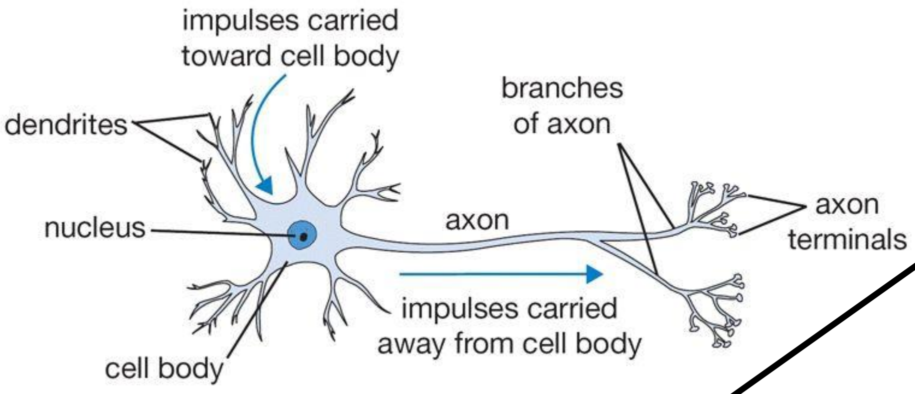


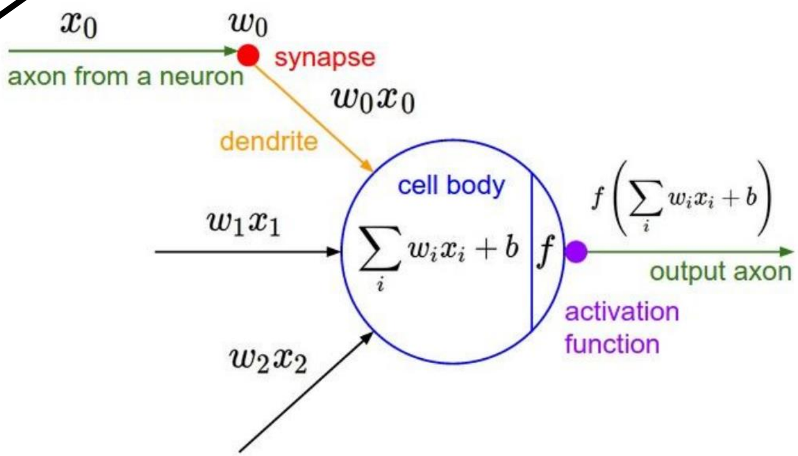
Biological Neurons



Credit: Stanford CS 231n

Biological Neurons



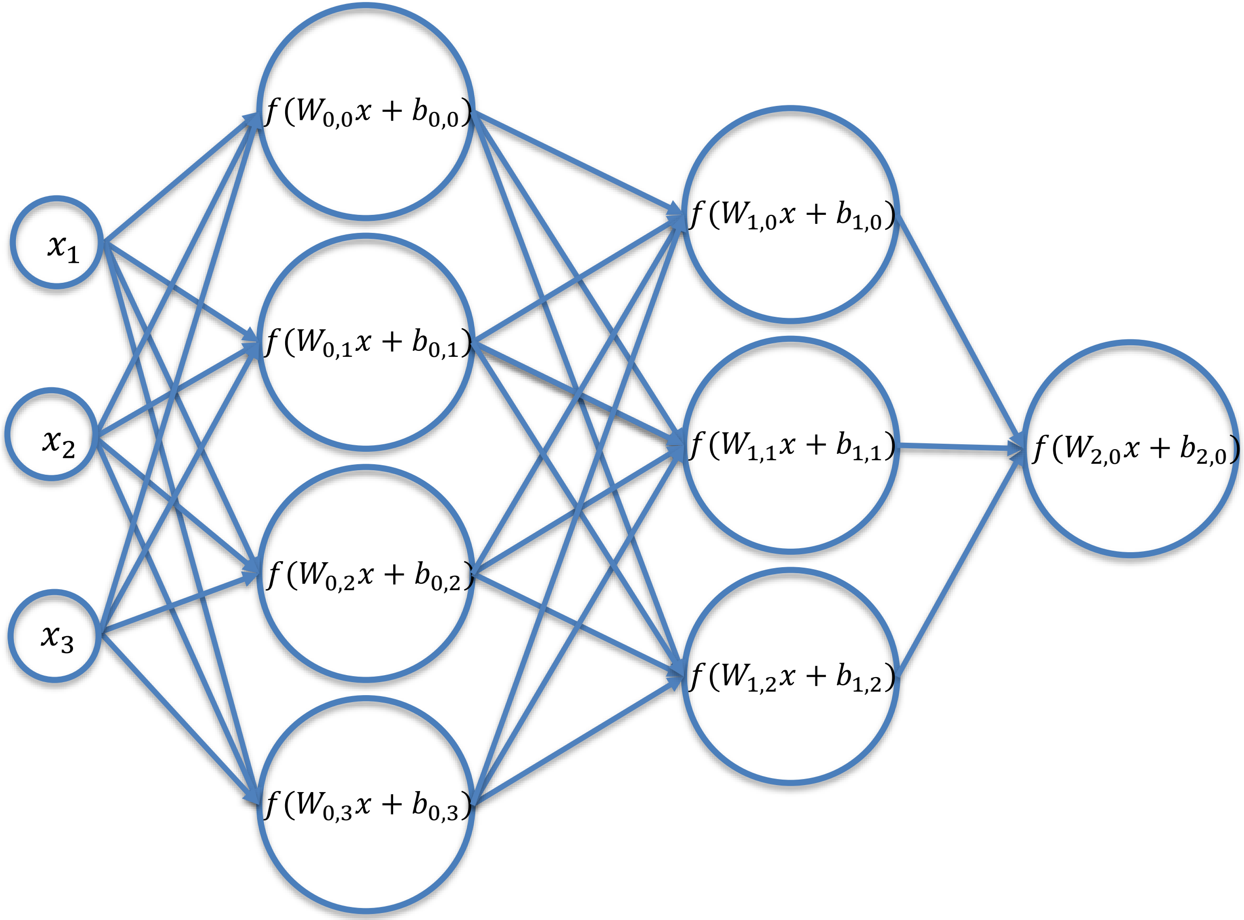


Credit: Stanford CS 231n

Artificial Neural Networks vs Brain

Artificial neural networks are **inspired** by the brain,
but not even close in terms of complexity!
The comparison is great for the media and news articles though... ☺

Artificial Neural Network



Neural Network

- Summary
  - Given a dataset with ground truth training pairs $[x_i; y_i]$,
  - Find optimal weights and biases $\boldsymbol{W}$ using stochastic gradient descent, such that the loss function is minimized
    - Compute gradients with backpropagation (use batch-mode; more later)
    - Iterate many times over training set (SGD; more later)

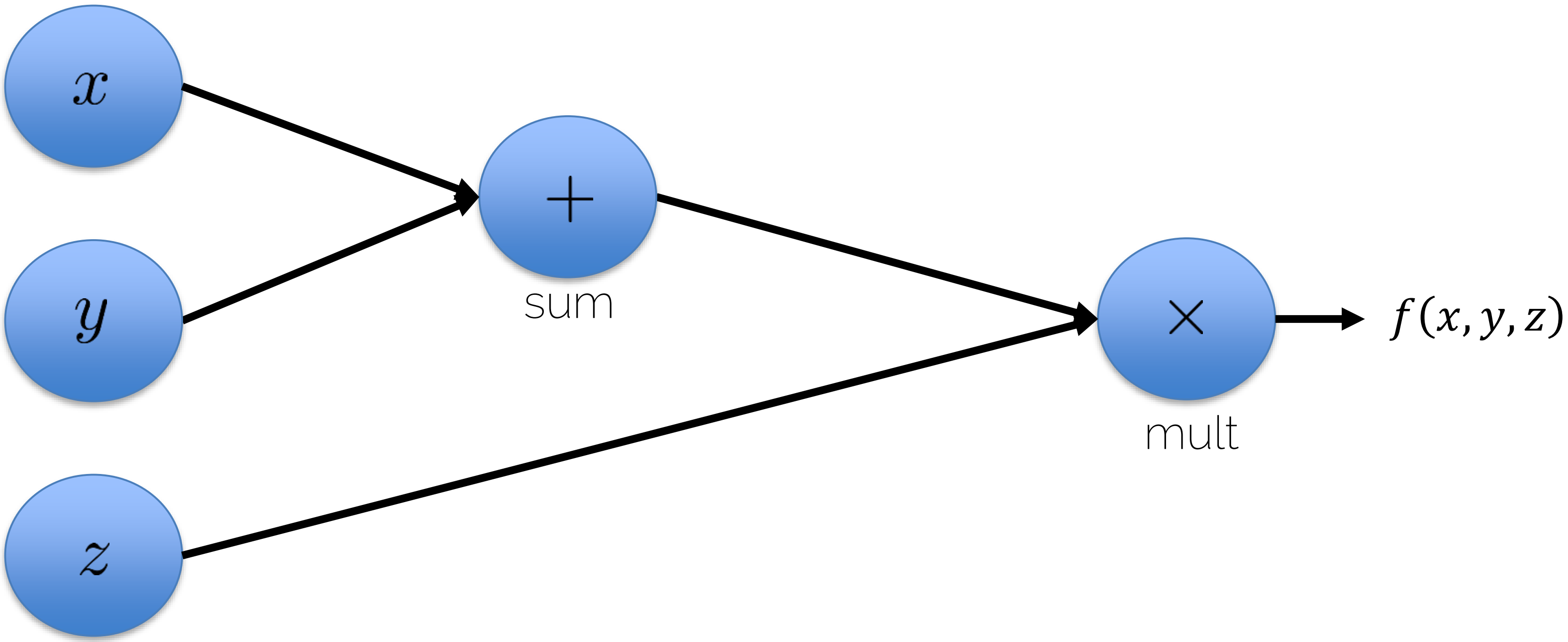
Computational Graphs

Computational Graphs

- Directional graph
- Matrix operations are represented as compute nodes.
- Vertex nodes are variables or operators like +, -, *, /, log(), exp() …
- Directional edges show flow of inputs to vertices

Computational Graphs

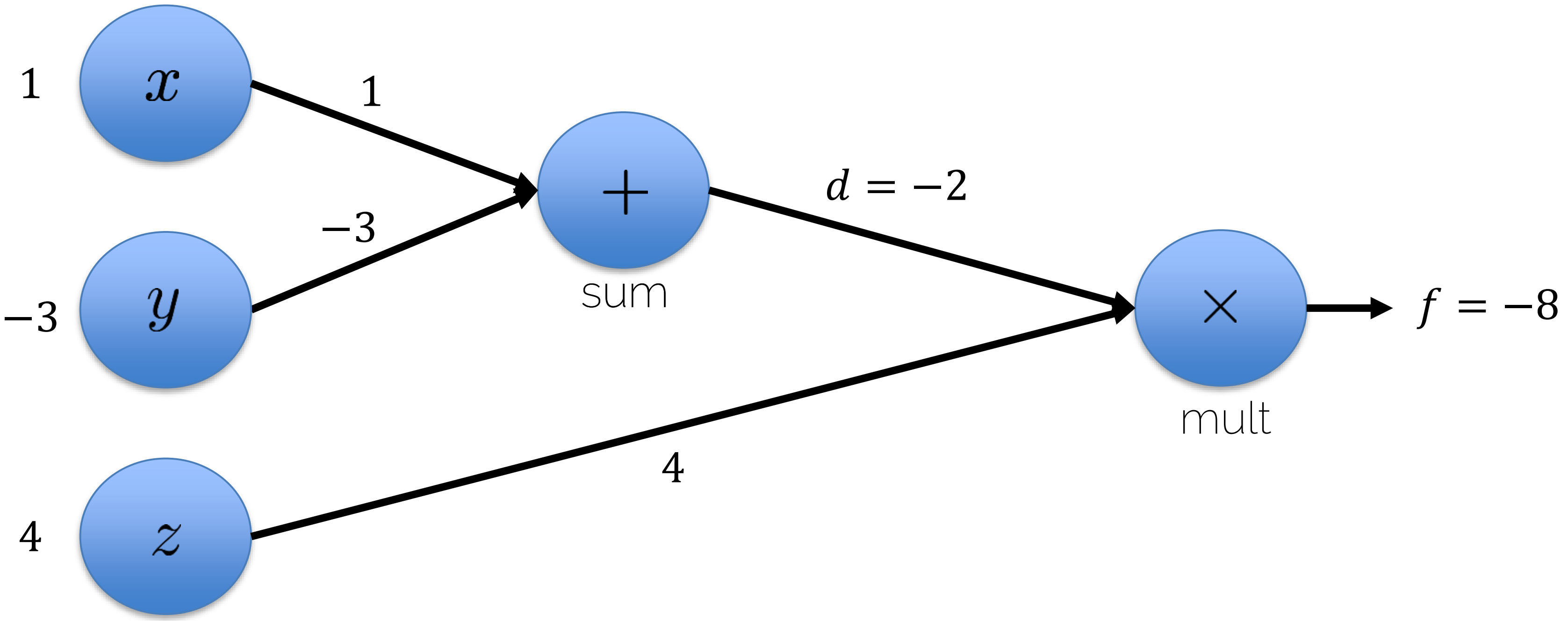
- $f(x,y,z) = (x+y)\cdot z$



Evaluation: Forward Pass

- $f(x, y, z) = (x + y) \cdot z$

Initialization $x = 1, y = -3, z = 4$



Computational Graphs

- Why discuss compute graphs?

- Neural networks have complicated architectures
  $\boldsymbol{f} = \boldsymbol{W}_5 \sigma(\boldsymbol{W}_4 \tanh(\boldsymbol{W}_3, \max(\mathbf{0}, \boldsymbol{W}_2 \max(\mathbf{0}, \boldsymbol{W}_1 \boldsymbol{x}))))$

- Lot of matrix operations!

- Represent NN as computational graphs!

Computational Graphs

A neural network can be represented as a computational graph...

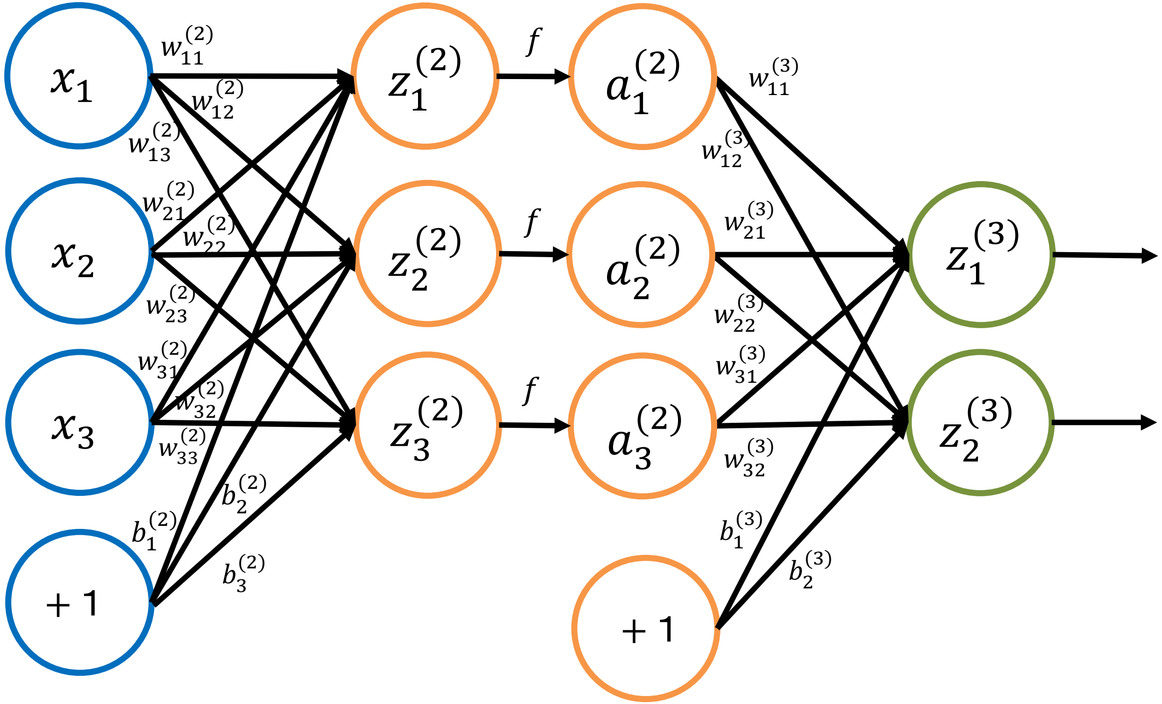
– it has compute nodes (operations)

– it has edges that connect nodes (data flow)

– it is directional

– it can be organized into 'layers'

Computational Graphs



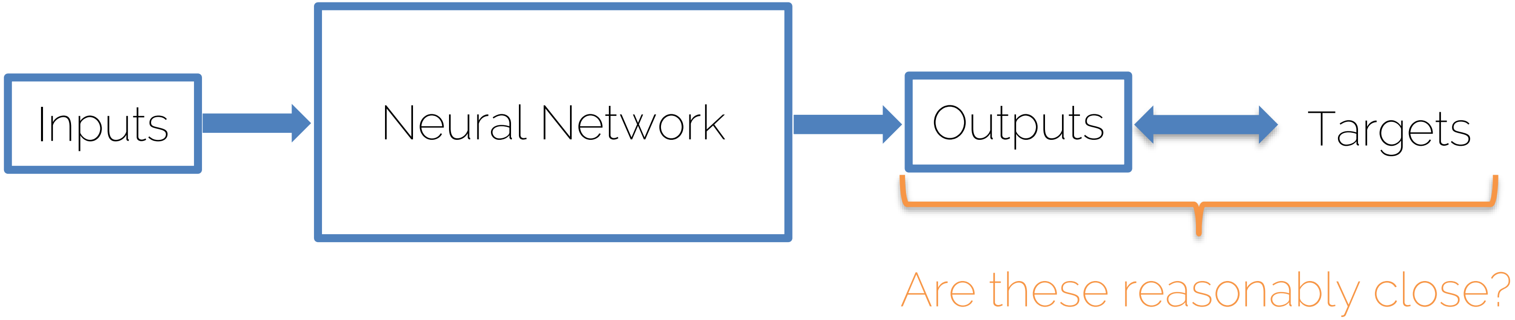
$$z_k^{(2)} = \sum_i x_i w_{ik}^{(2)} + b_k^{(2)}$$

$$a_k^{(2)} = f(z_k^{(2)})$$

$$z_k^{(3)} = \sum_i a_i^{(2)} w_{ik}^{(3)} + b_k^{(3)}$$

Loss Functions

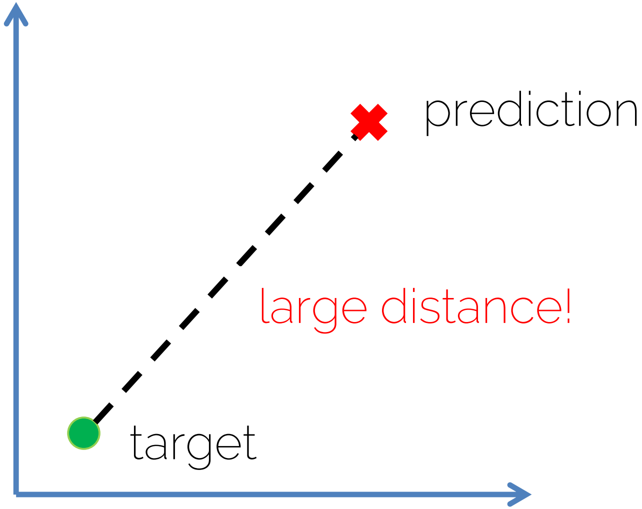
What's Next?



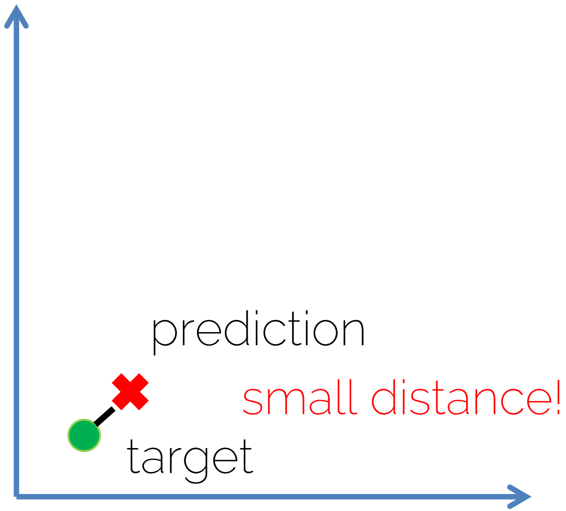
We need a way to describe how close the network's outputs (= predictions) are to the targets!

What's Next?

Idea: calculate a **'distance'** between prediction and target!



bad prediction



good prediction

Loss Functions

- A function to **measure the goodness of the predictions** (or equivalently, the network's performance)

Intuitively, ...

- a **large loss indicates bad predictions**/performance (→ performance needs to be improved by training the model)
- the choice of the loss function depends on the concrete problem or the distribution of the target variable

Regression Loss

- L1 Loss:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{n} ||y_i\ - \widehat{y_i}||_1$$

- MSE Loss:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_{i}^{n} ||y_i\ - \widehat{y_i}||_2^2$$

Binary Cross Entropy

- Loss function for binary (yes/no) classification

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}};\ \boldsymbol{\theta}) = -\frac{1}{n}\sum_{i}^{n}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

Yes! (0.8)

No! (0.2)

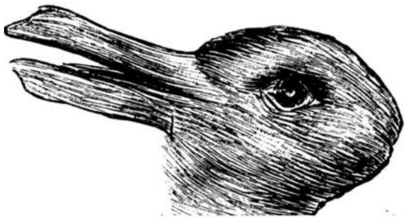
The network predicts the probability of the input belonging to the "yes" class!

Cross Entropy

- loss function for multi-class classification

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = -\sum_{i=1}^{n}\sum_{k=1}^{k}(y_{ik} \cdot \log \hat{y}_{ik})$$

This generalizes the binary case from the slide before!



dog (0.1)

rabbit (0.2)

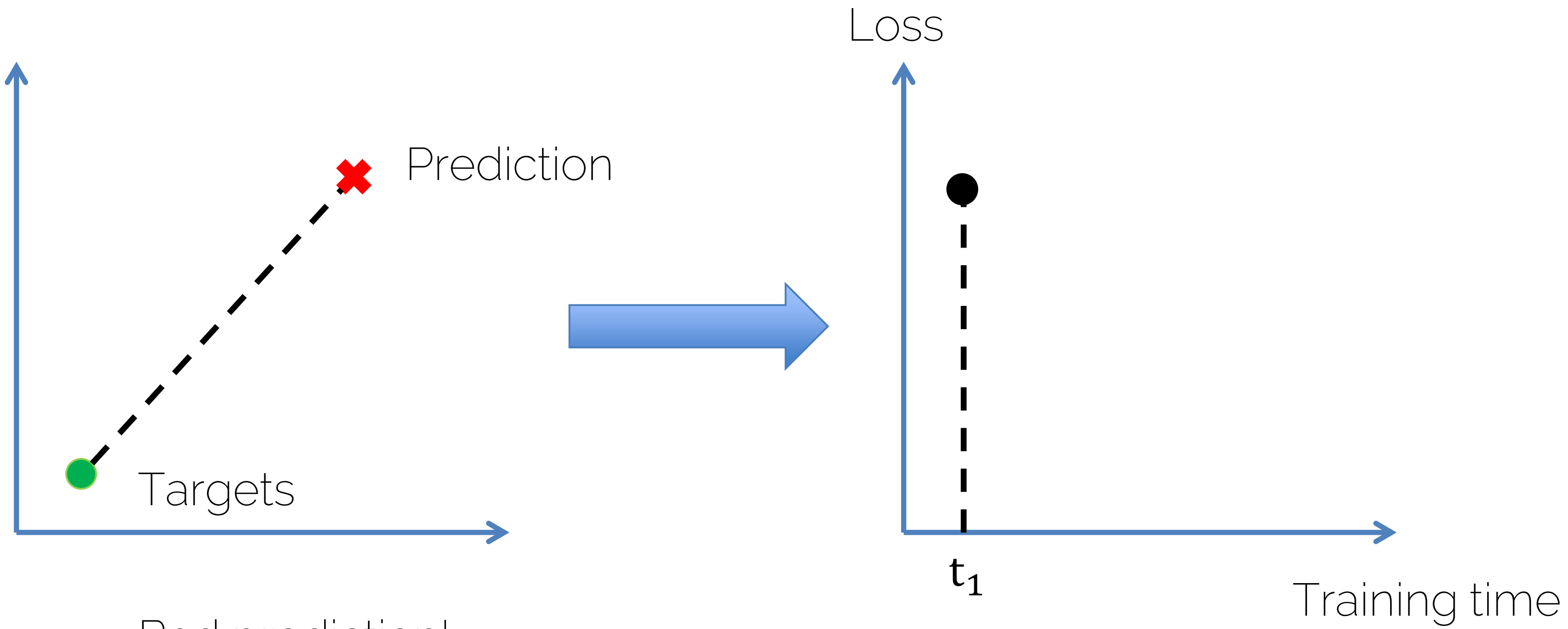
duck (0.7)

...

More General Case

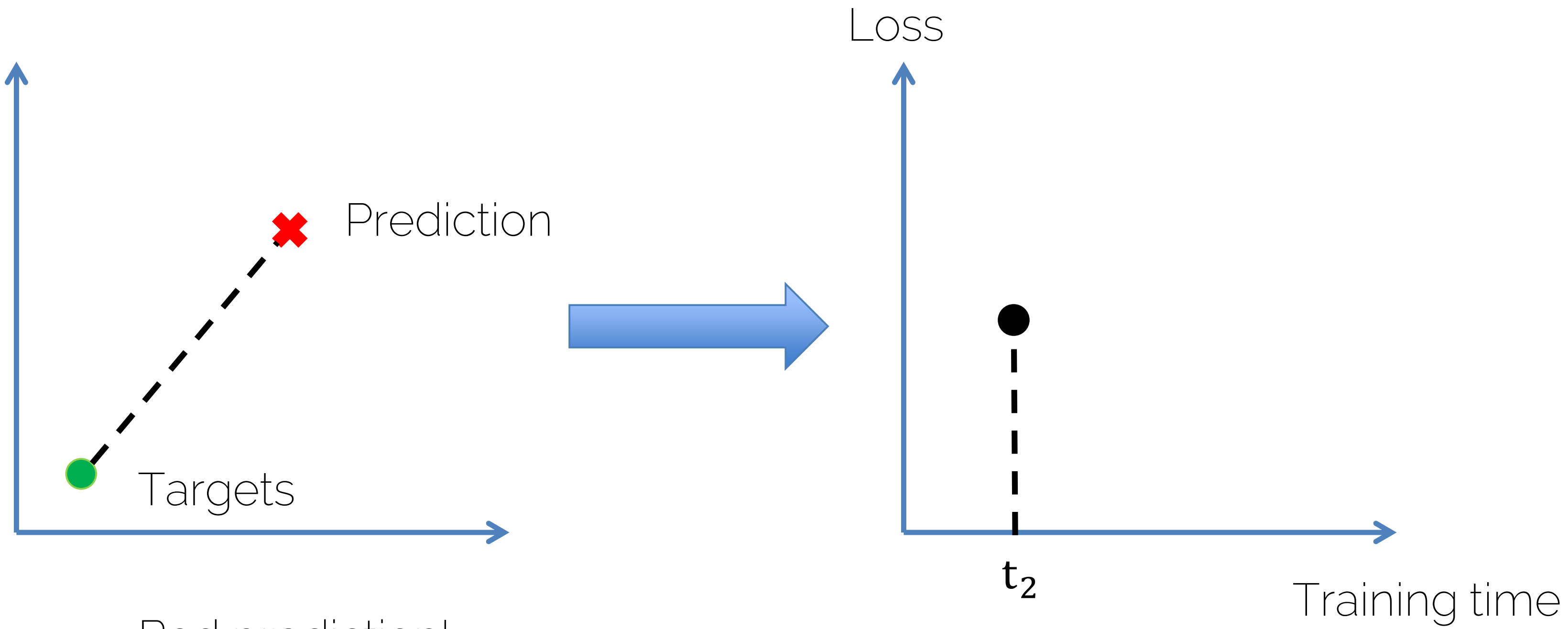
- Ground truth: $\boldsymbol{y}$
- Prediction: $\hat{\boldsymbol{y}}$
- Loss function: $\boldsymbol{L(y, \hat{y})}$
- Motivation:
  - minimize the loss <=> find better predictions
  - predictions are generated by the NN
  - find better predictions <=> find better NN

Initially



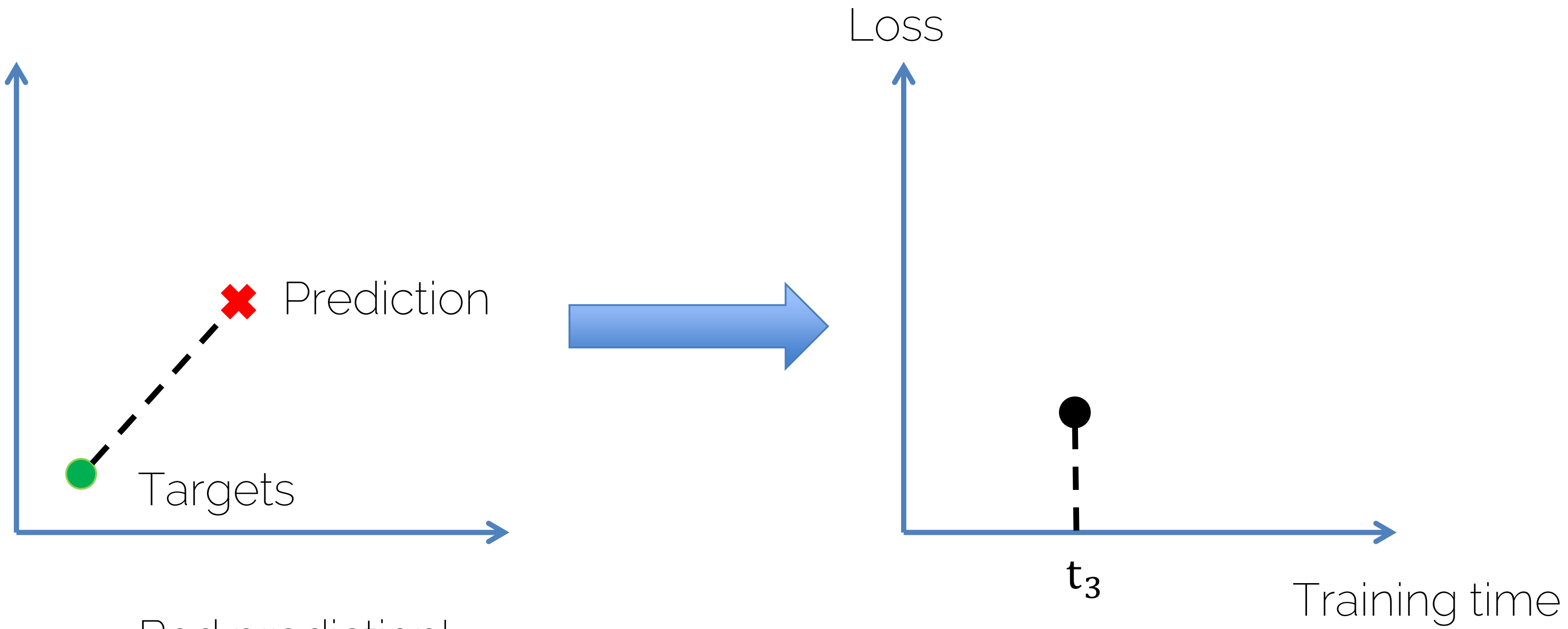
Bad prediction!

During Training…



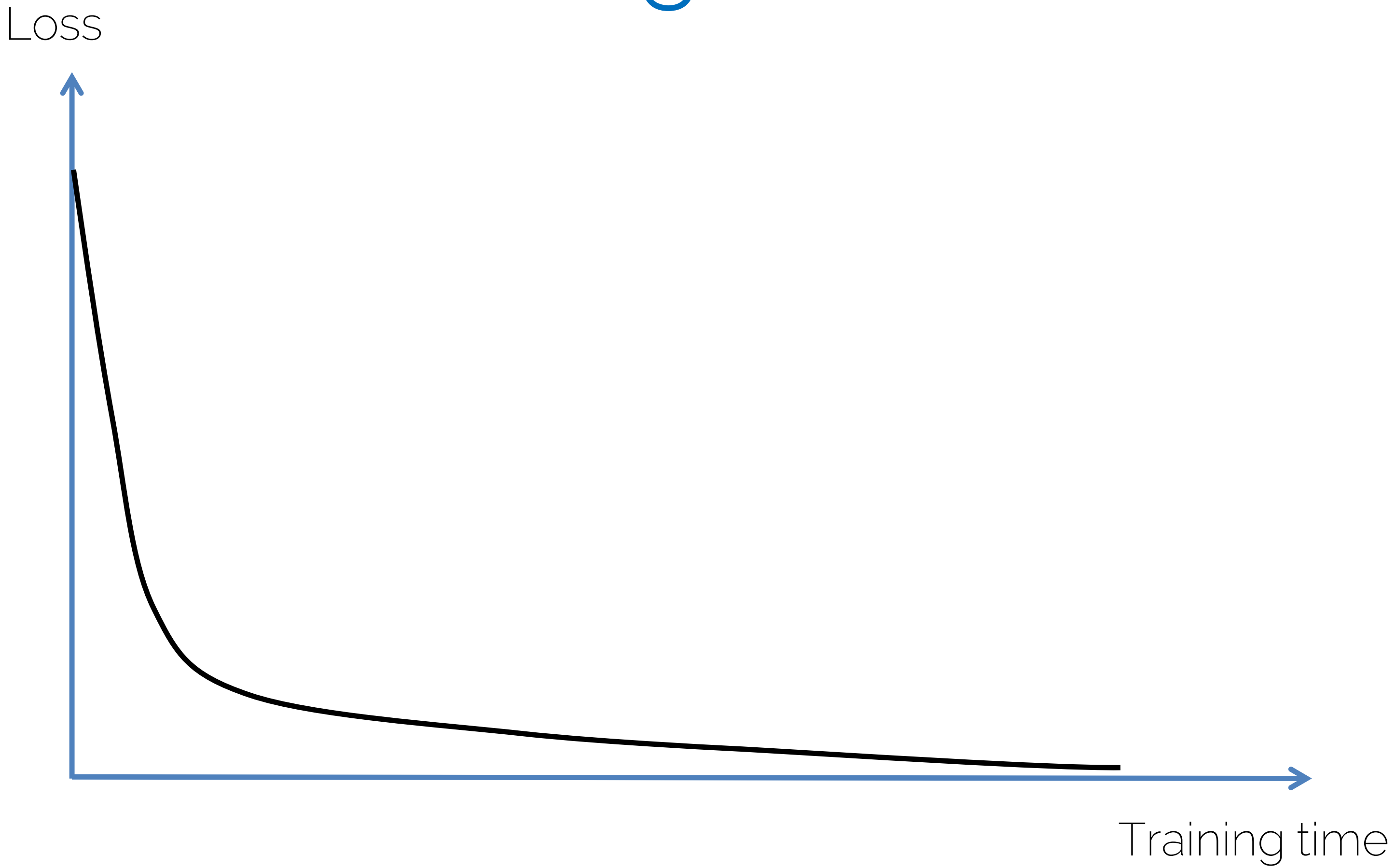
Bad prediction!

During Training…



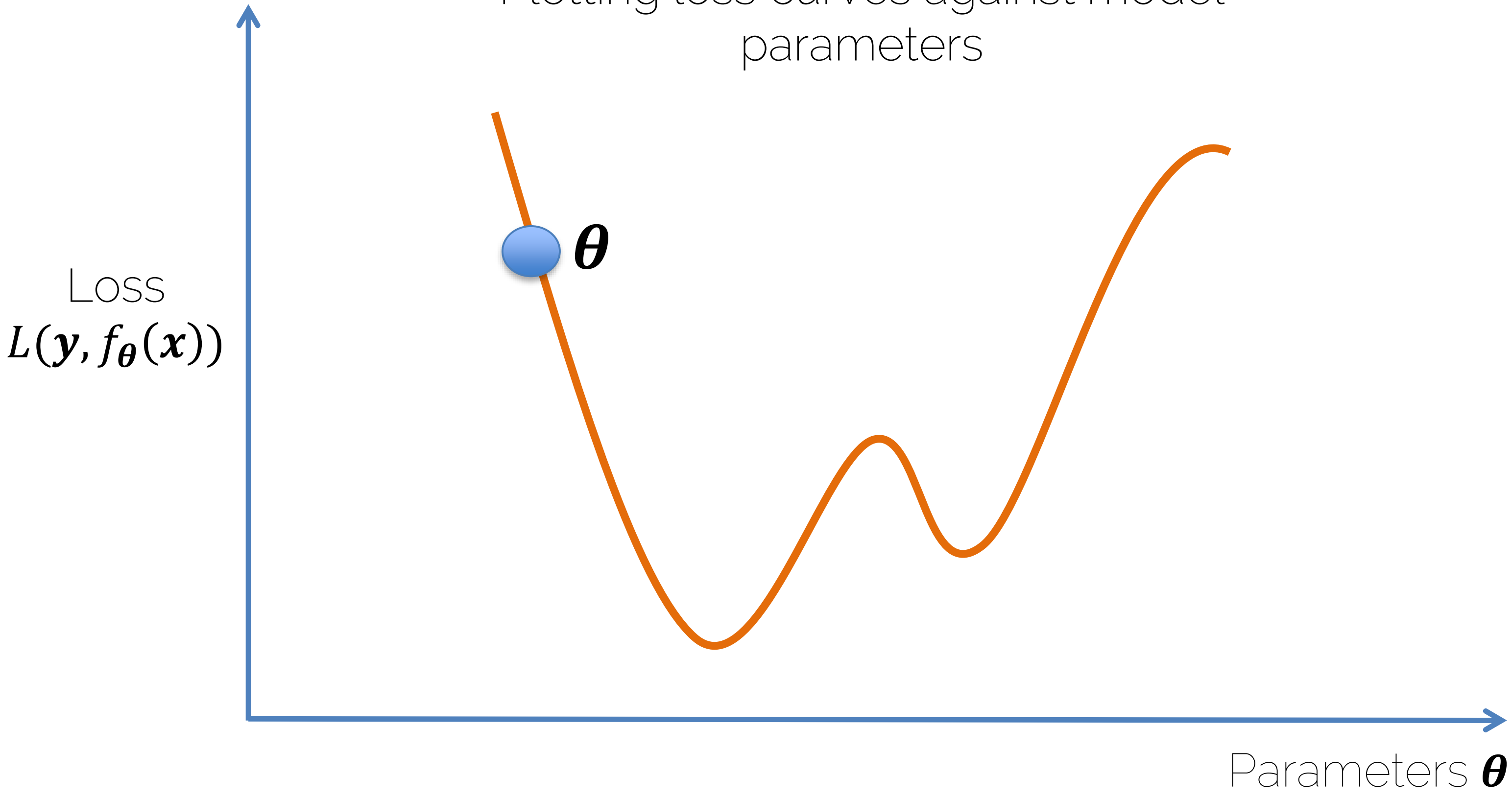
Bad prediction!

Training Curve



How to Find a Better NN?

Plotting loss curves against model parameters



How to Find a Better NN?

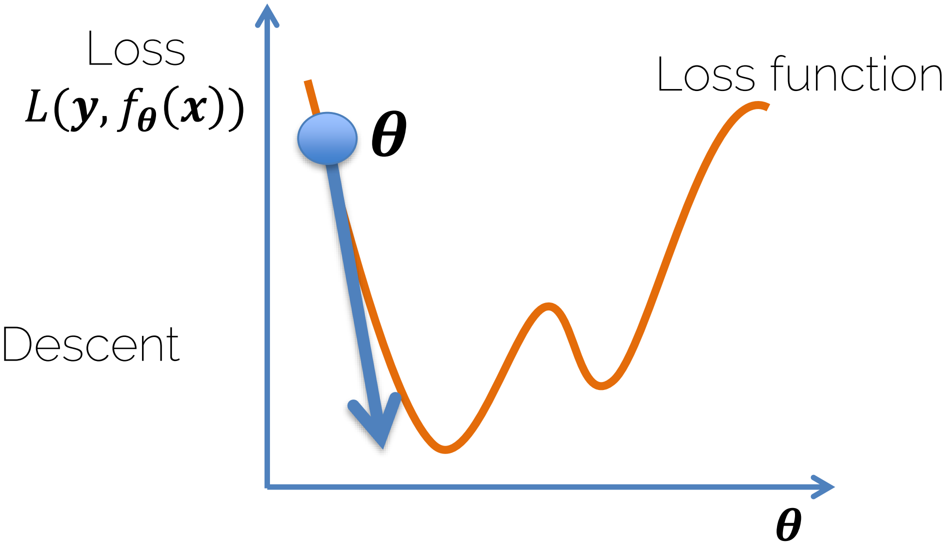
- Loss function: $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$
- Neural Network: $f_{\boldsymbol{\theta}}(\boldsymbol{x})$
- Goal:
  - minimize the loss w. r. t. $\boldsymbol{\theta}$

→ Optimization! We train compute graphs with some optimization techniques!

How to Find a Better NN?

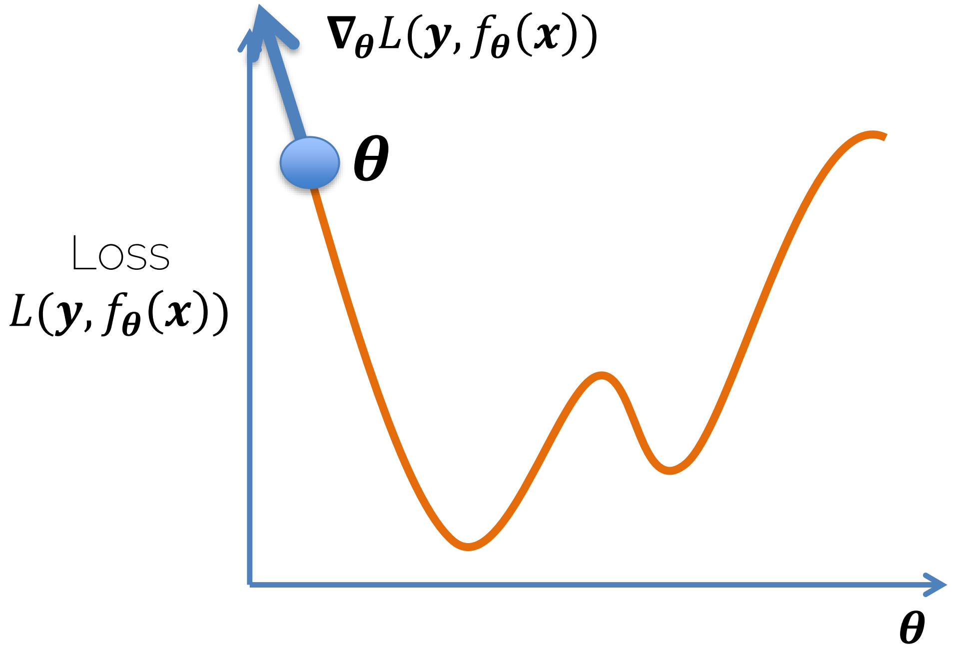
- Minimize: $L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$ w.r.t. $\boldsymbol{\theta}$
- In the context of NN, we use gradient-based optimization

Gradient Descent



How to Find a Better NN?

- Minimize: $\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{f_\theta}(\boldsymbol{x}))$ w.r.t. $\boldsymbol{\theta}$



Learning rate

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$$

$$\boldsymbol{\theta}^* = \arg\min L(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$$

How to Find a Better NN?

- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given one layer NN with no activation function

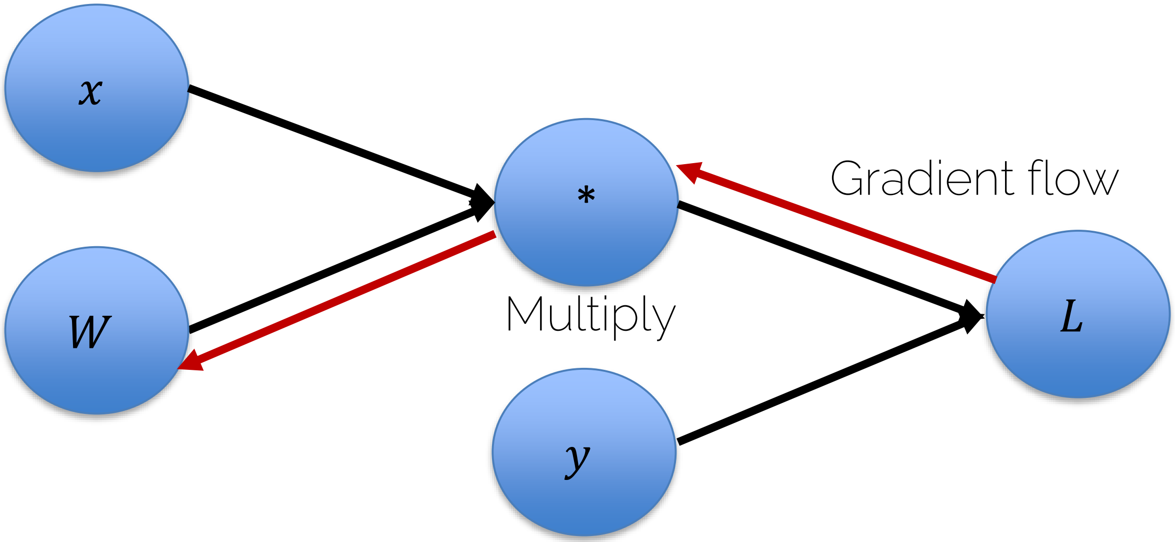
$$\boldsymbol{f_\theta}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}\,, \qquad \boldsymbol{\theta} = \boldsymbol{W}$$

Later $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$

- Given MSE Loss: $\boldsymbol{L}(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i - \widehat{y}_i||_2^2$

How to Find a Better NN?

- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given one layer NN with no activation function
- Given MSE Loss: $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i - \boldsymbol{W} \cdot x_i||_2^2$



How to Find a Better NN?

- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given one layer NN with no activation function

$$f_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}, \qquad \boldsymbol{\theta} = \boldsymbol{W}$$

- Given MSE Loss: $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||\boldsymbol{W} \cdot x_i - y_i||_2^2$
- $\nabla_\theta L(\boldsymbol{y}, f_\theta(\boldsymbol{x})) = \frac{2}{n}\sum_i^n (\boldsymbol{W} \cdot x_i - y_i) \cdot x_i^T$

How to Find a Better NN?

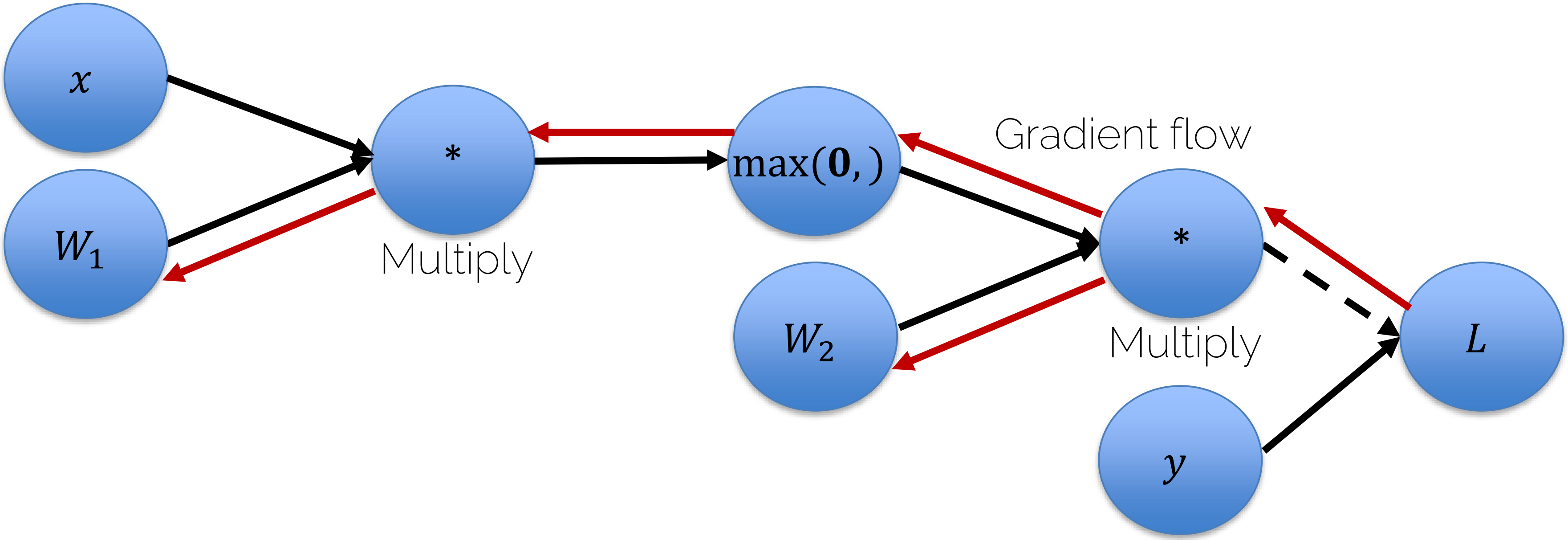
- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given a multi-layer NN with many activations

$$\boldsymbol{f} = \boldsymbol{W}_5 \sigma(\boldsymbol{W}_4 \tanh(\boldsymbol{W}_3, \max(\boldsymbol{0}, \boldsymbol{W}_2 \max(\boldsymbol{0}, \boldsymbol{W}_1 \boldsymbol{x}))))$$

- Gradient descent for $\boldsymbol{L}\big(\boldsymbol{y}, \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)$ w. r. t. $\boldsymbol{\theta}$
  - Need to propagate gradients from end to first layer ($\boldsymbol{W}_1$).

How to Find a Better NN?

- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given multi-layer NN with many activations



How to Find a Better NN?

- Given inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$
- Given multilayer layer NN with many activations
  $\boldsymbol{f} = \boldsymbol{W}_5 \sigma(\boldsymbol{W}_4 \tanh(\boldsymbol{W}_3, \max(\mathbf{0}, \boldsymbol{W}_2 \max(\mathbf{0}, \boldsymbol{W}_1 \boldsymbol{x}))))$
- Gradient descent solution for $L\big(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x})\big)$ w. r. t. $\boldsymbol{\theta}$
  - Need to propagate gradients from end to first layer ($\boldsymbol{W}_1$)
- Backpropagation: Use chain rule to compute gradients
  - Compute graphs come in handy!

How to Find a Better NN?

- Why gradient descent?
  - Easy to compute using compute graphs
- Other methods include
  - Newtons method
  - L-BFGS
  - Adaptive moments
  - Conjugate gradient

Summary

- Neural Networks are computational graphs
- Goal: for a given train set, find optimal weights

- Optimization is done using gradient-based solvers
  - Many options (more in the next lectures)

- Gradients are computed via backpropagation
  - Nice because can easily modularize complex functions