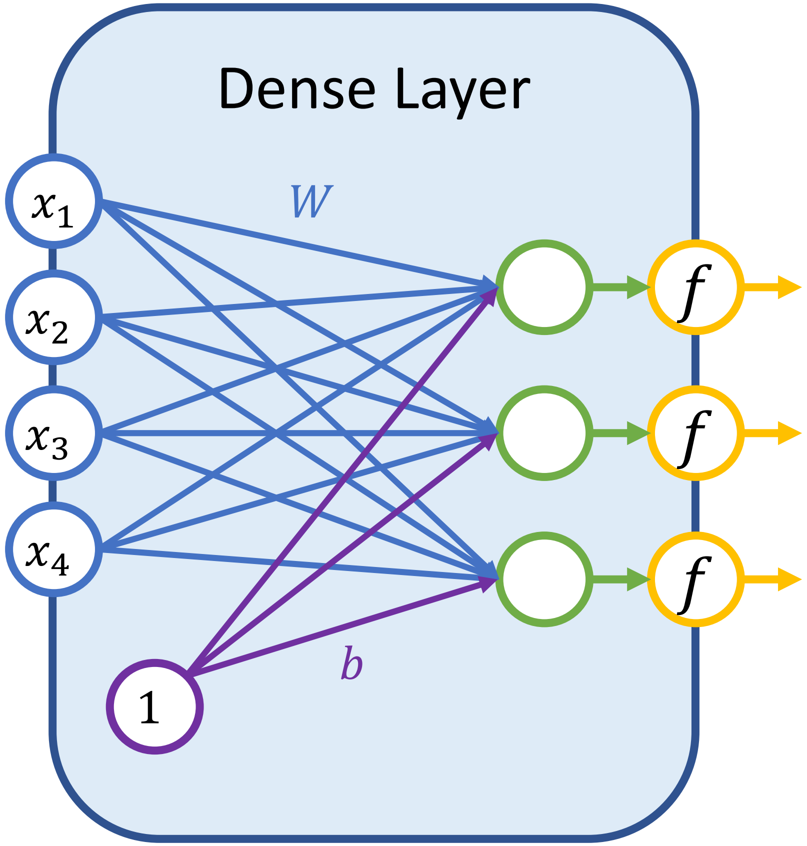
Deep Neural Networks
And Where to Find Them

Lecture 3

Artem Korenev, Nikita Gryaznov

Recap

Recap – Dense Layer



x_1 Layer input

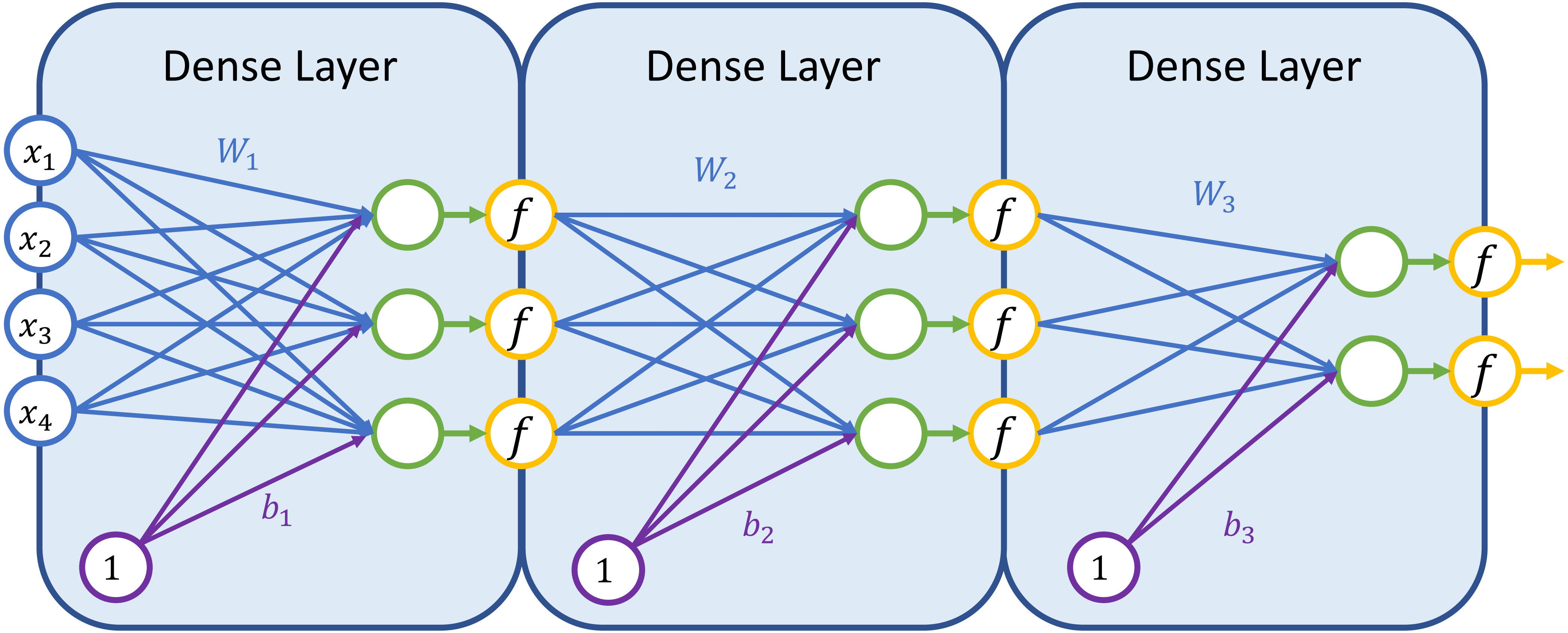
○ Neuron Output
$(W_{i1}x_1 + \cdots + W_{i2}x_2 + b_i)$

f Activation function
(Non-linearity)
(Neuron activation)

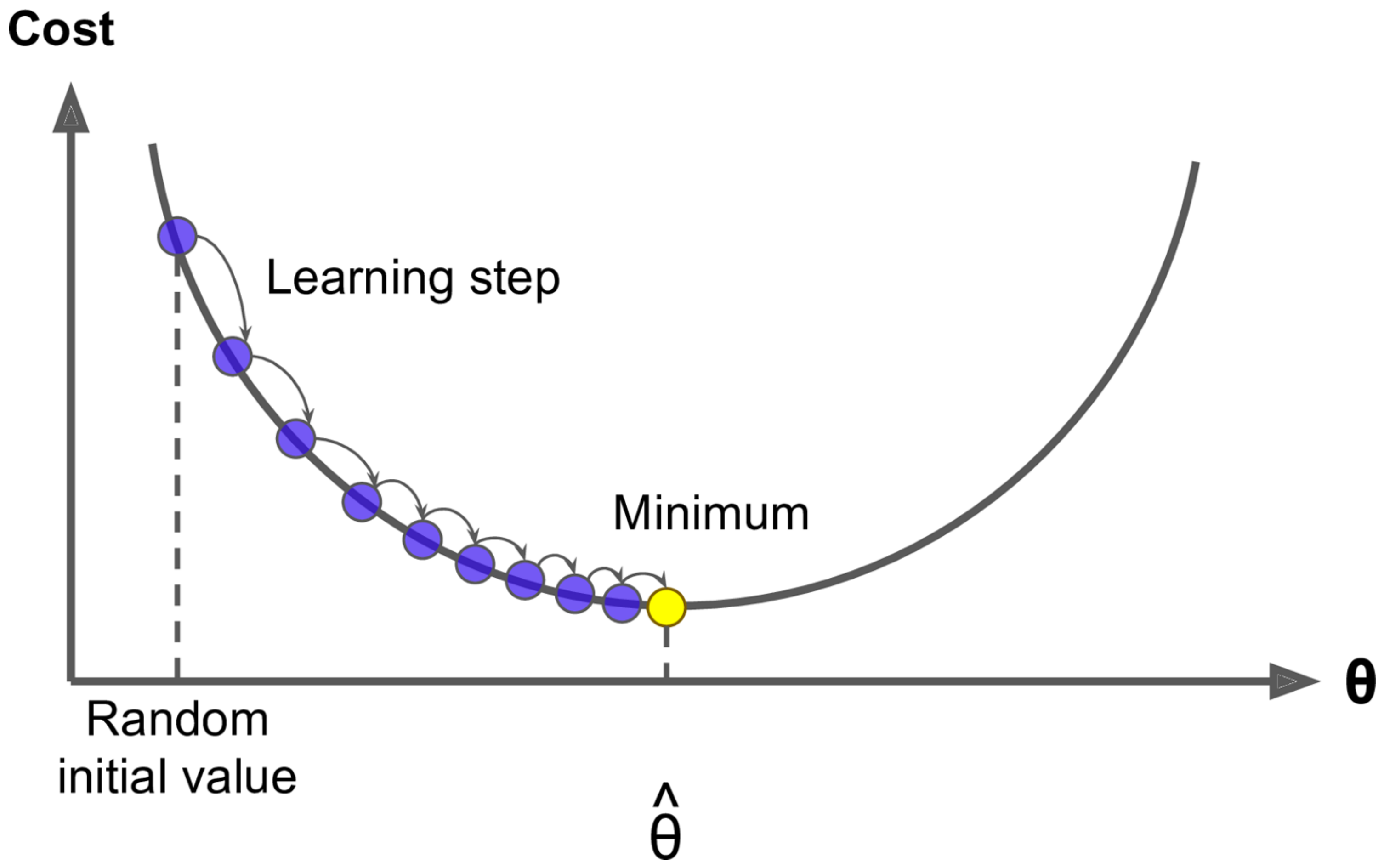
→ Layer weight
(trainable parameter)

→ Layer bias
(trainable parameter)

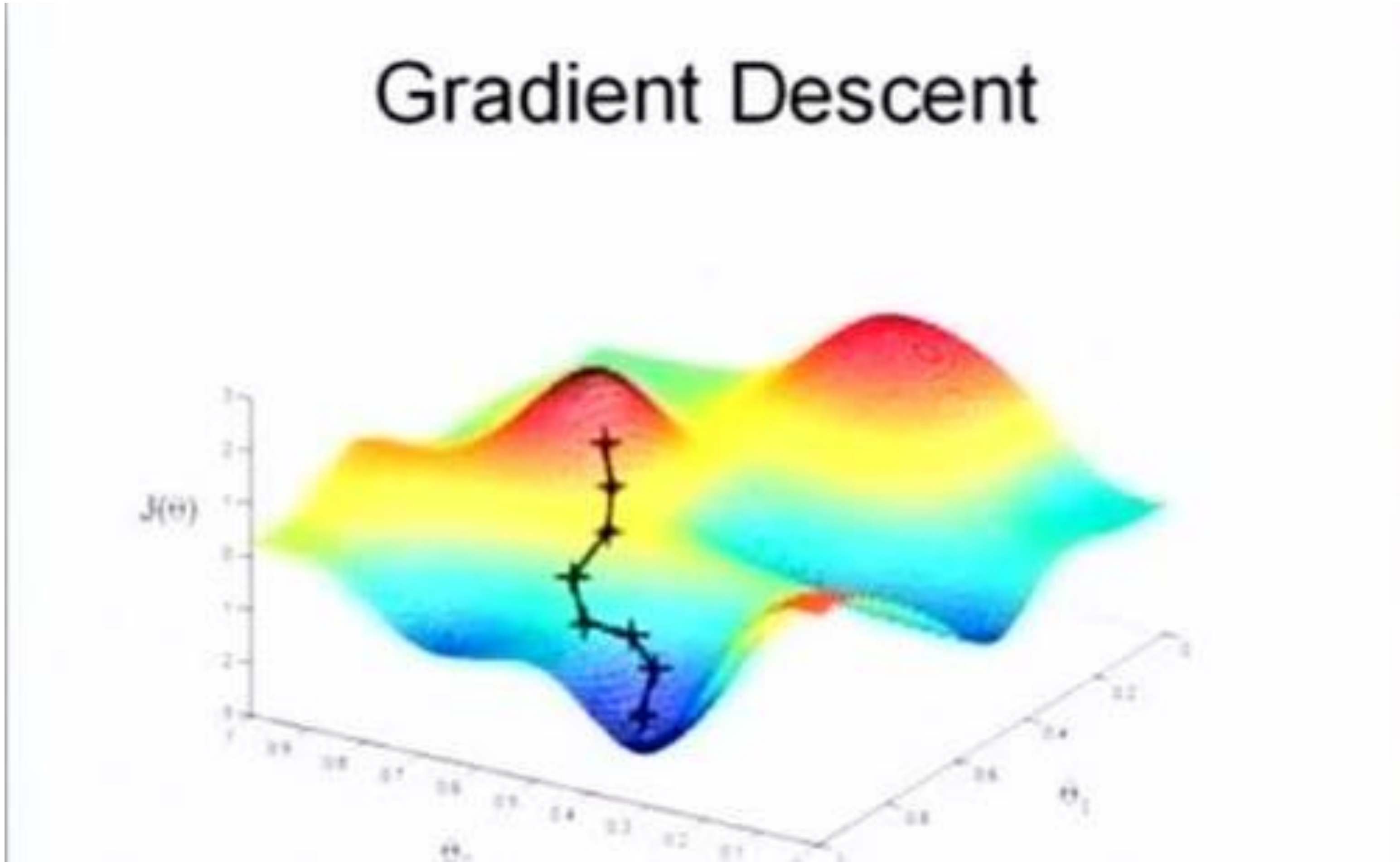
Recap – Multi Layer Neural Network



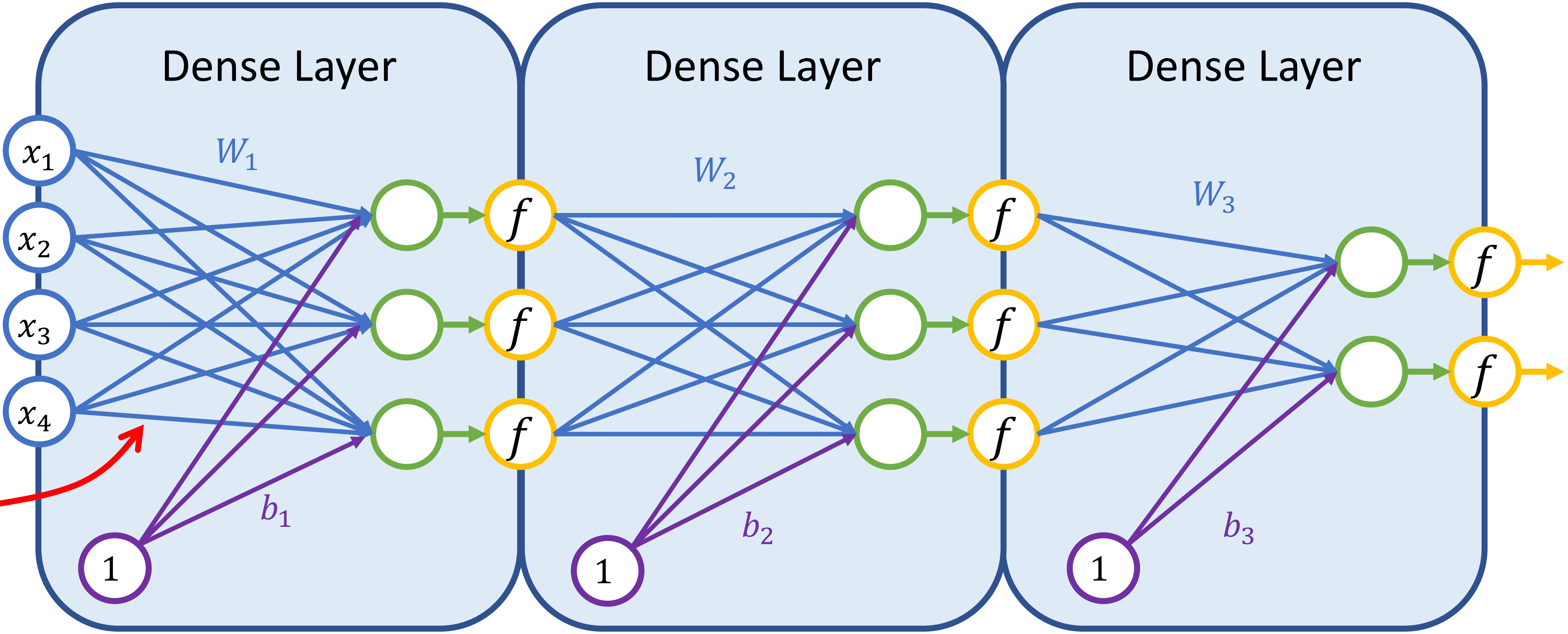
Recap – NN Optimization



Recap – NN Optimization



Recap – NN Optimization

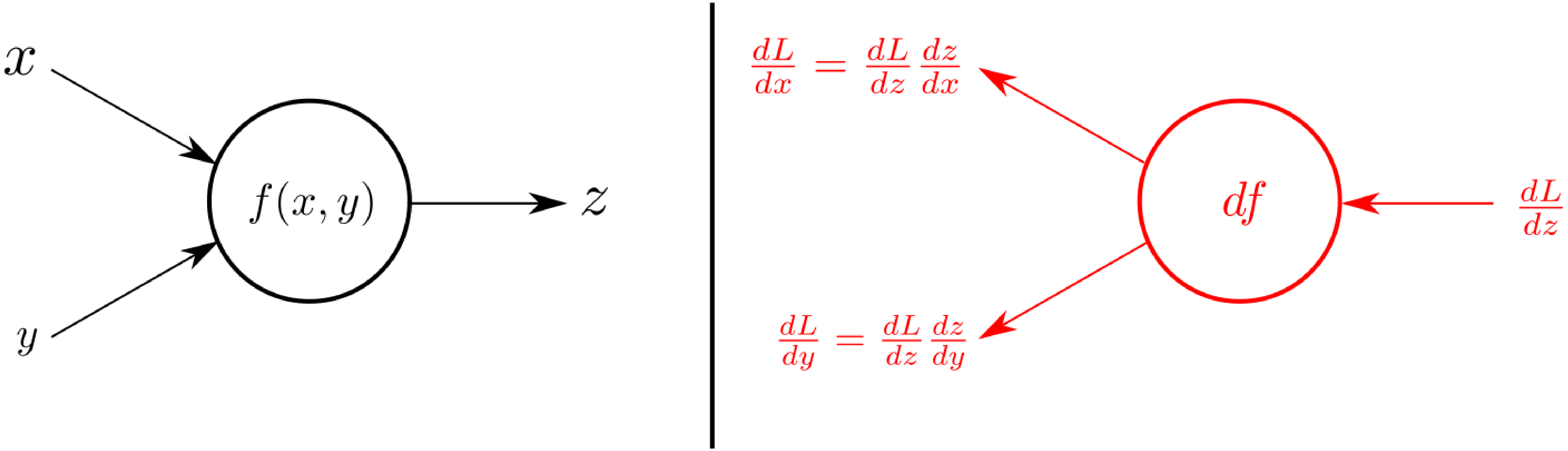


How do we compute gradients for this weight?

Recap – NN Optimization – Backpropagation

Forwardpass

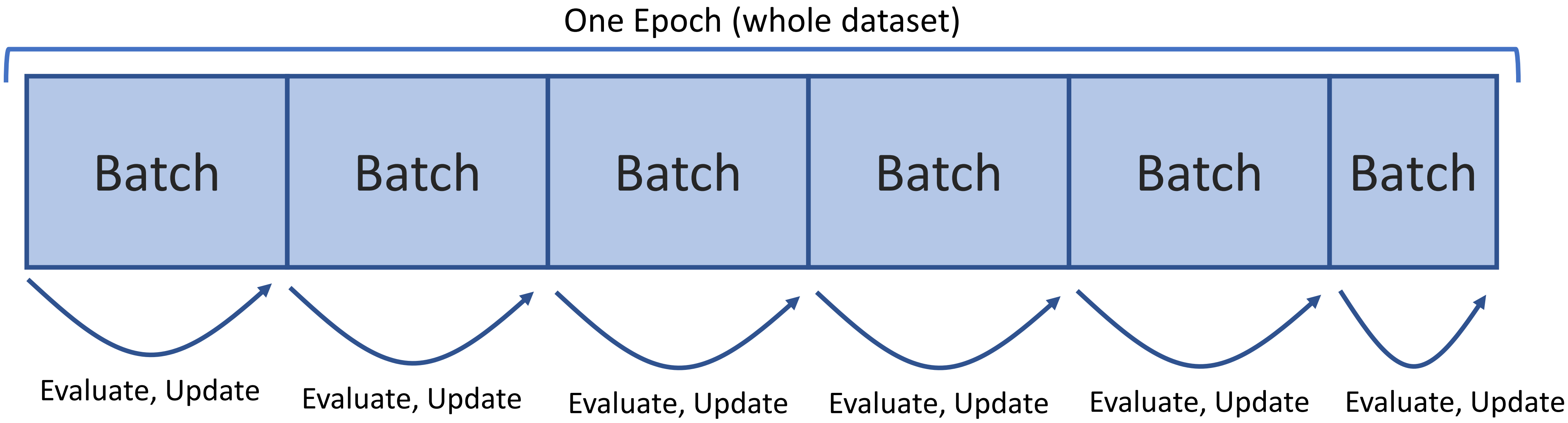
Backwardpass



$$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$$

$$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$$

Recap – NN Optimization - Minibatch



When you optimize in mini-batch setting this is called *stochastic gradient descent*

Activation Functions

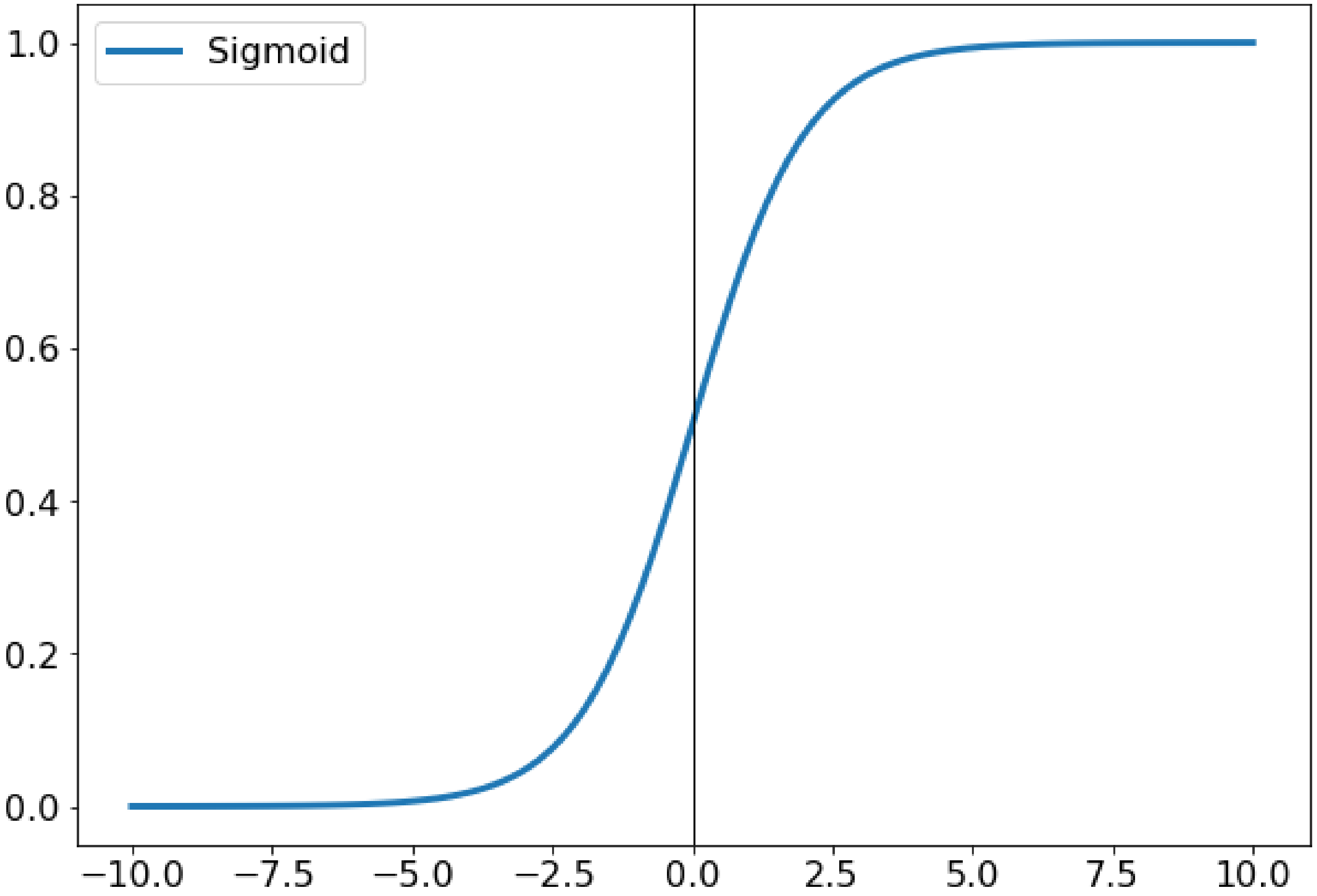
Activation Functions

- A non-linearity function between layers of Neural Networks
- The main reason why NNs are so powerful

Popular Activation Functions:

- Sigmoid
- Tanh
- ReLU (and many variations)

Sigmoid



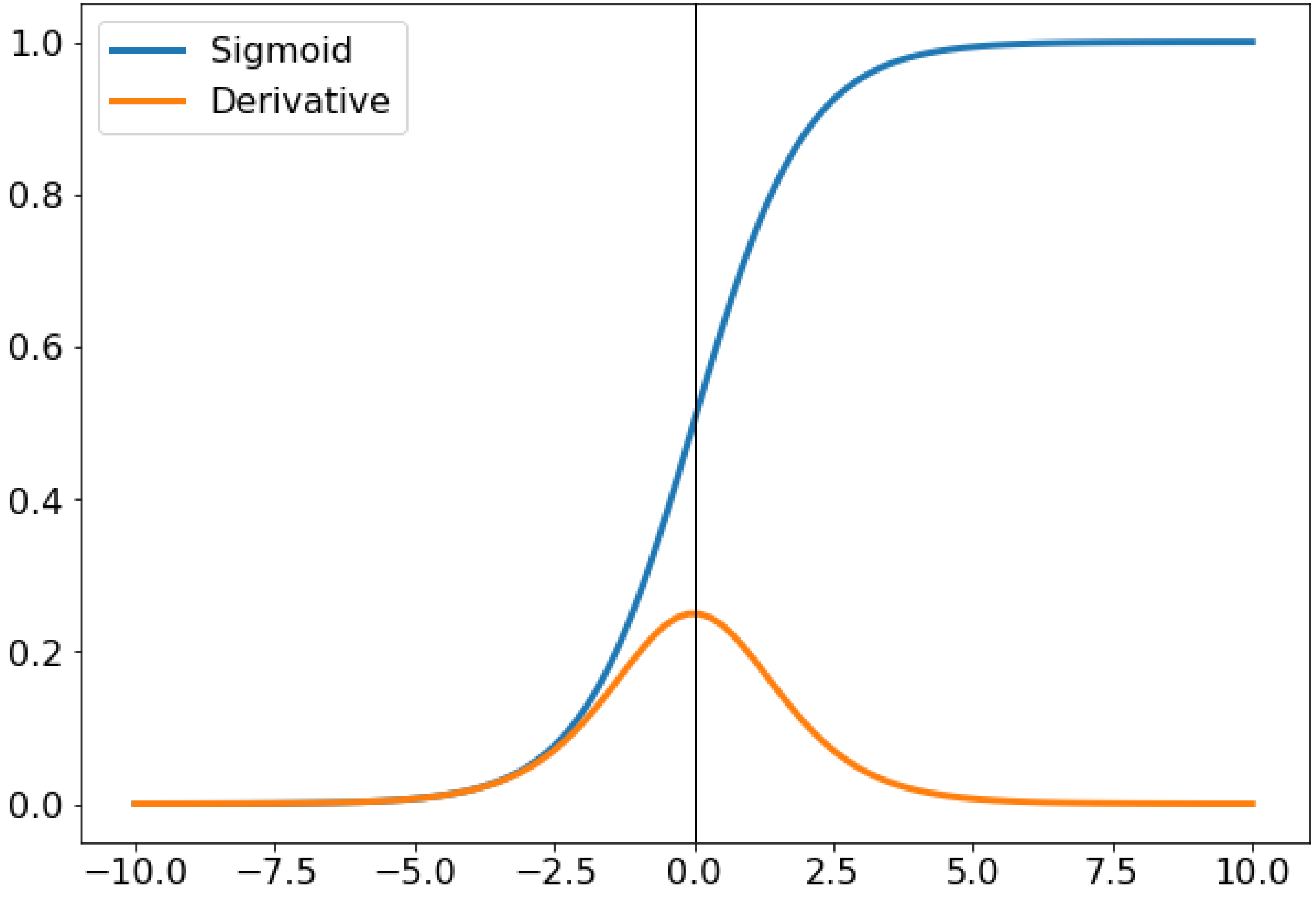
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- Maps real values to [0,1] range
- Historically popular

Cons:

- Saturated neurons kill gradient
- Sigmoid outputs are not zero-centered
- exp() is computationally expensive

Sigmoid (saturated neurons)



If x is small or large

derivative $\frac{d\sigma}{dx}$ is almost zero

Hence

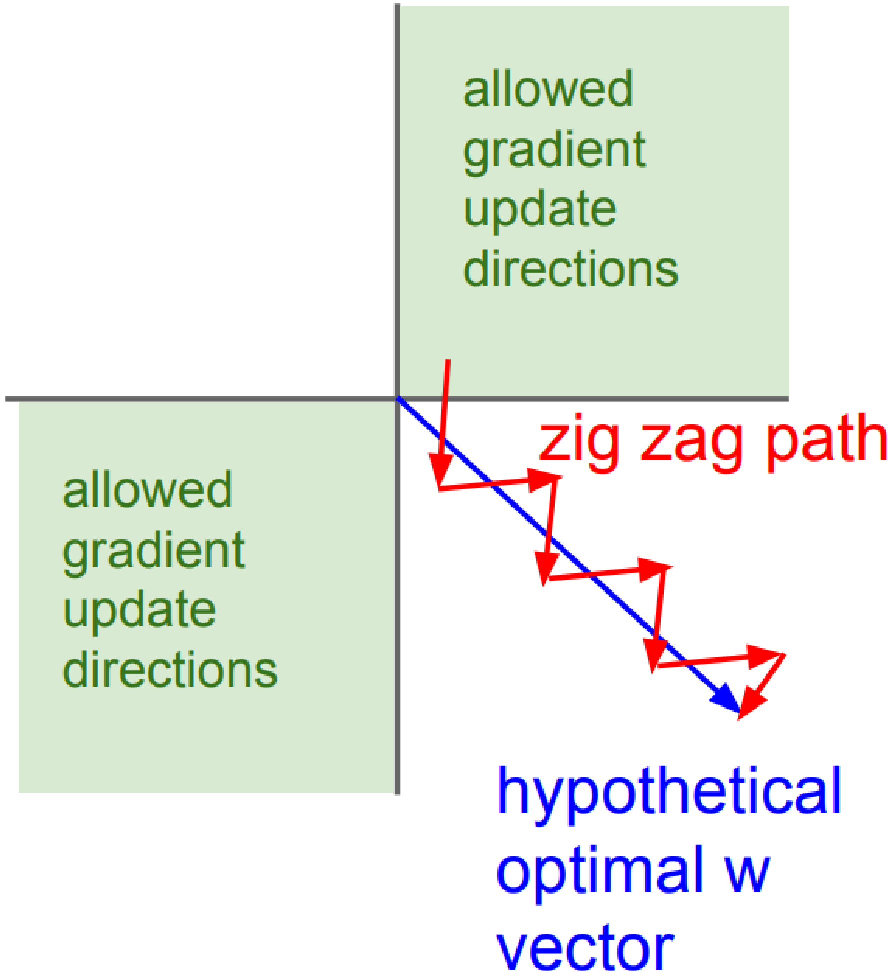
$$\frac{dL}{dx} = \frac{d\sigma}{dx}\frac{dL}{d\sigma}$$

is almost zero too

And all subsequent gradients are zero too

So backpropagation gets killed

Sigmoid (non-zero centered)


allowed gradient update directions
zig zag path
allowed gradient update directions
hypothetical optimal w vector

$$x_i = \sigma(\dots)$$

then all $x_i > 0$

Consider next layer

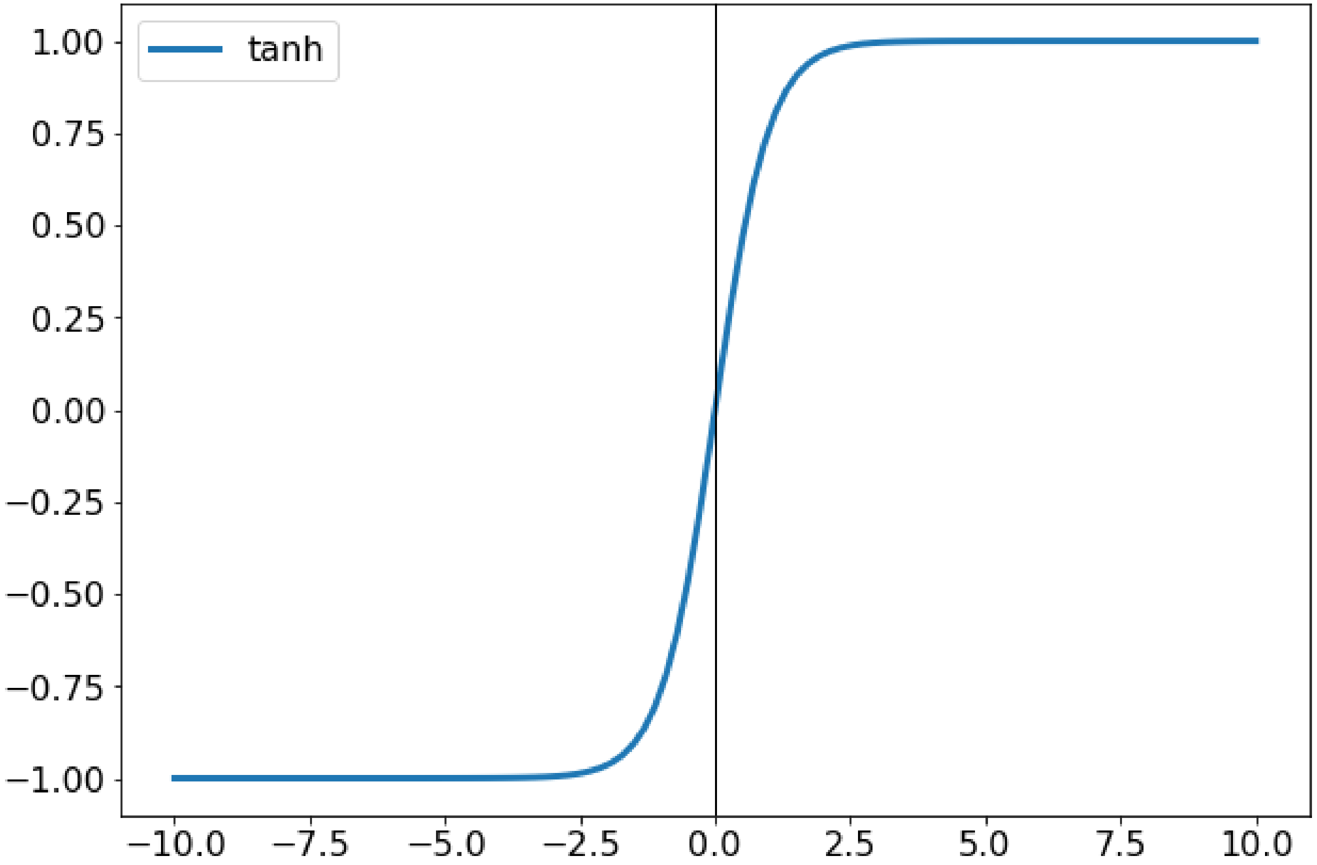
$$f = \sum_{i=1}^{n} x_i w_i + b$$

Then

$$\frac{dL}{dw_i} = \frac{dL}{df} x_i$$

is always positive or negative

Tanh



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Maps real values to [-1,1] range

Pros:

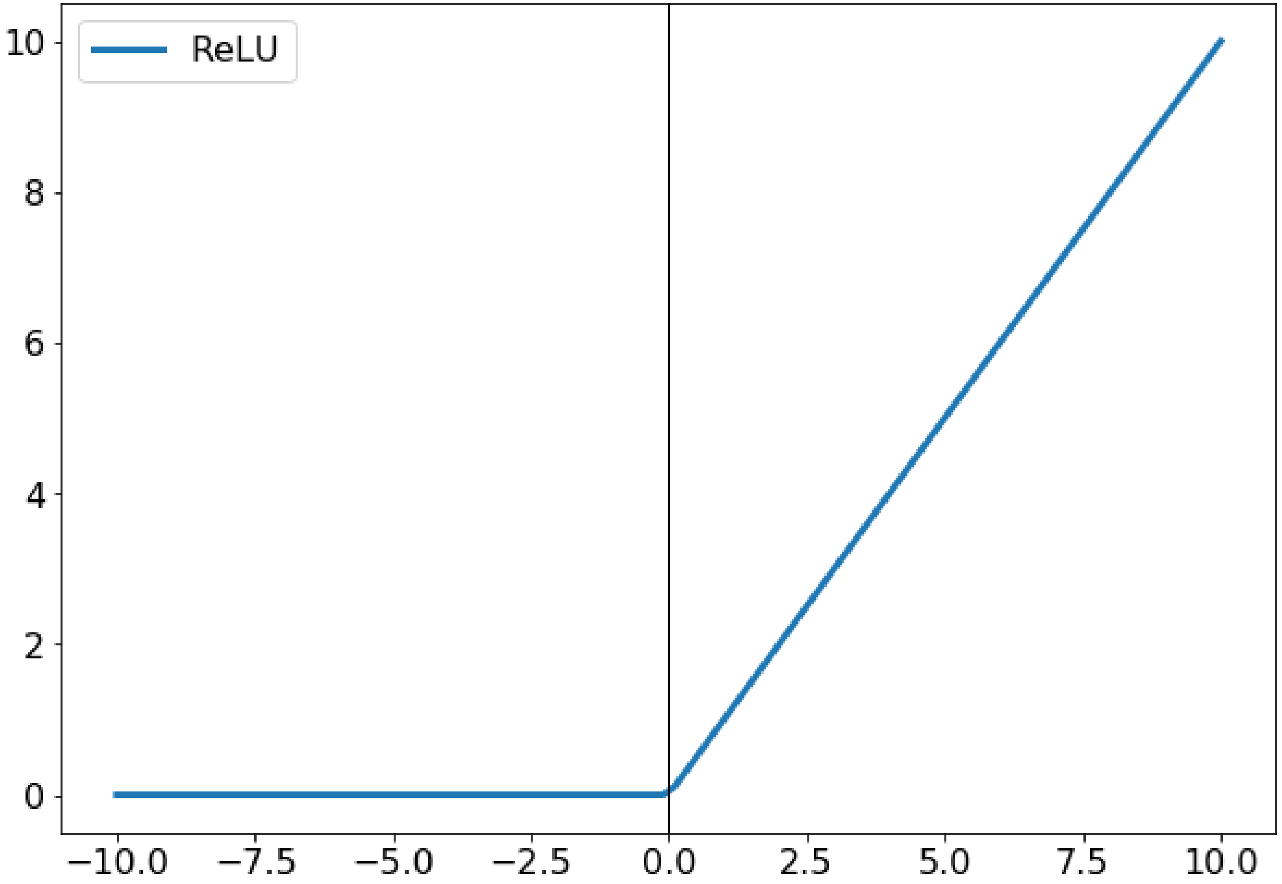
- Zero-centered

Cons:

- Saturated neurons kill gradient
- exp() is computationally expensive

ReLU

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



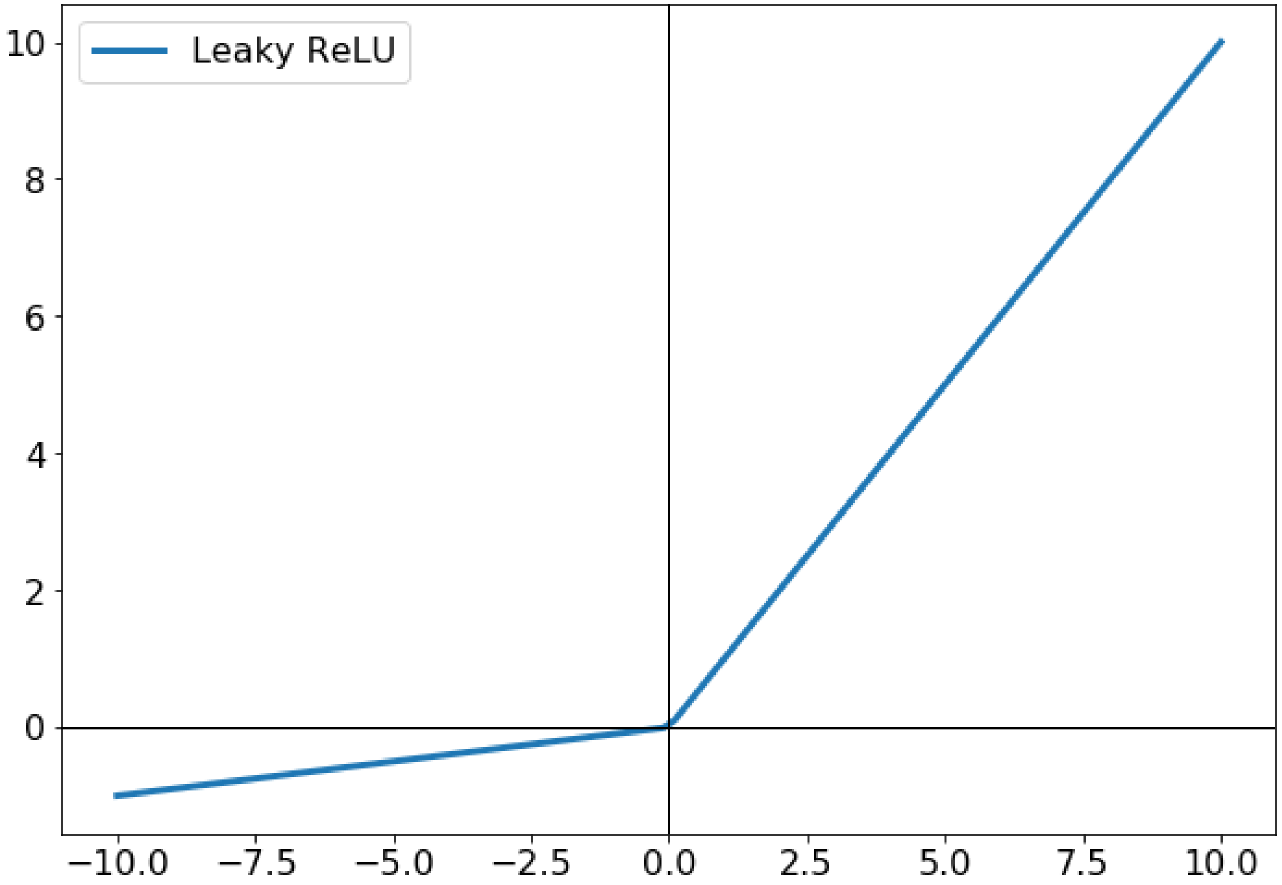
Pros:

- Does not saturate on $x > 0$
- Computationally efficient
- Converges about 6x times faster than sigmoid/tanh

Cons:

- Not zero centered
- Dies: gradient is 0 for $x < 0$, so weights can become unapdatable

Leaky ReLU



$$f(x) = \max(0.01x, x)$$

Pros:

- Does not saturate Computationally efficient
- Converges about 6x times faster than sigmoid/tanh
- Doesn’t die

Loss Functions

Recap of NN Outputs and Labels

- Binary classification label is presented as $y \in \{0, 1\}$
- Multiclass classification label is presented as a vector y, s.t. $y^T = (0, \dots, 0, 1, 0, \dots, 0)$, where $y_i = 1$ says that the object belongs to class i
- Sometimes an object can belong to multiple classes and also belong to classes softly, i.e. $y_i = 0.2$
- NN output in binary classification is presented as a *value* $f(x)$
- NN output in multiple classification is presented as a *vector* $f(x)$

Loss Functions

- It should describe your objective
- In order to use Backpropagation they must be *differentiable*
- Though Loss function must be differentiable there are a couple of exceptions

$$\max(0, 1 - y * f(x))$$

- Gradient is undefined at $1 - y * f(x) = 0$ but to handle this we can just say that it equals to 0

Popular losses

- Mean Squared Error (or MSE) (or L2 loss)
- Mean Absolute Error (or MAE) (or L1 loss)
- Huber Loss
- Hinge Loss
- Cross Entropy

Mean Square Error Loss

$$L\big(y, f(x)\big) = \big(y - f(x)\big)^2$$

- Often used for regression tasks
- Rarely used for classification tasks
- Not really robust to outliers

Mean Absolute Error Loss

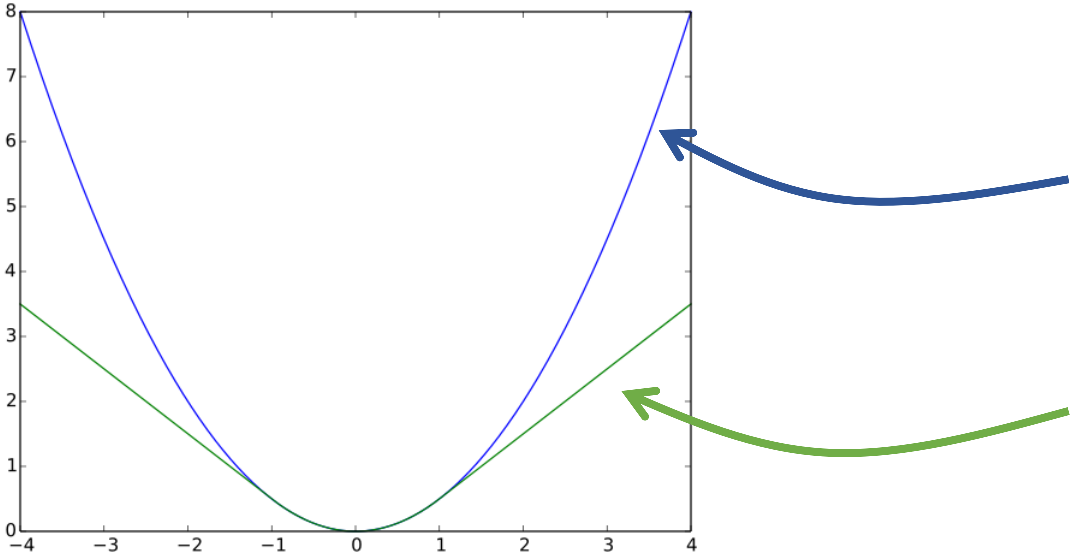
$$L\big(y, f(x)\big) = |y - f(x)|$$

- Often used for regression tasks
- Rarely used for classification tasks
- More robust to outliers

Huber Loss

- A combination of Mean Squared Error and Mean Absolute Error

$$L\big(y, f(x)\big) = \begin{cases} \dfrac{1}{2}(y - f(x))^2 & for\ |y - f(x)| \le \delta \\ \delta|y - f(x)| - \dfrac{1}{2}\delta^2, & otherwise \end{cases}$$



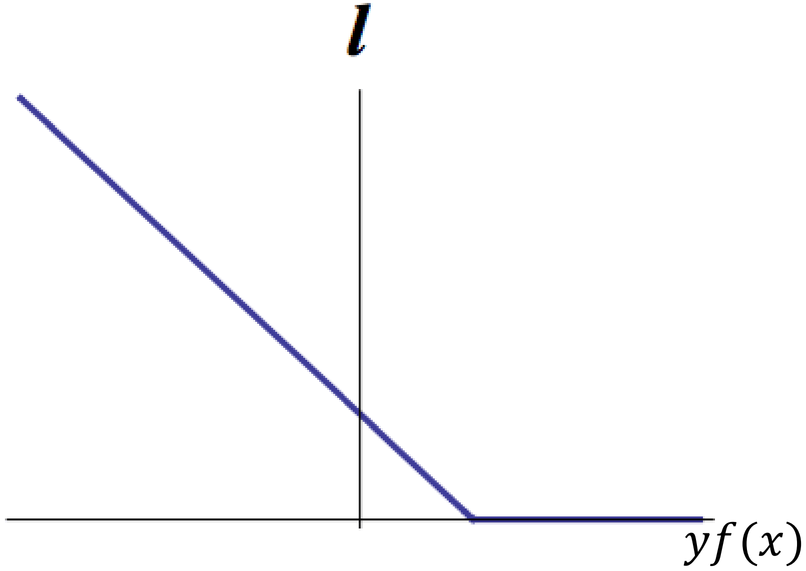
Mean Squared Error

Huber Loss

Hinge Loss

$$L\big(y, f(x)\big) = \max(0{,}1 - yf(x))$$

- Sometimes used for classification problems
- y values expected to be $\{-1, 1\}$ (instead of $\{0, 1\}$)
- $f(x)$ is expected to be unbounded (i.e. to return values $(-\infty, +\infty)$)
- Doesn’t penalize “correct” examples



Variations of Hinge Loss

- Squared Hinge Loss

$$L\big(y, f(x)\big) = \big(\max\big(0,1 - yf(x)\big)\big)^2$$

- Multiclass Variation

$$L\big(y, f(x)\big) = \max(0, 1 + \max_{t \neq argmax(y)} f(x)_t - f(x)_{argmax(y)})$$

- Sometimes used for classification problems
- Claimed to be useful in case of a lot of classes

Cross Entropy

- Binary Cross Entropy

$$L\big(y, f(x)\big) = -(y\log(f(x)) + (1-y)\log\big(1-f(x)\big))$$

- Categorical Cross Entropy (Multiclass)

$$L\big(y, f(x)\big) = \sum_{k=1}^{m} y_k \log(f(x)_k)$$

In case of only one class per point

$$L\big(y, f(x)\big) = \log(f(x)_{argmax(y)})$$

- A primal loss choice for classification problems

Recap

- For classification problem Cross Entropy Loss most of the times is a safe choice
- For regression problems consider Mean Squared Error or Mean Absolute Error depending on specifics of your data/problem
- Research thoroughly about the best choice thoroughly

Metrics

Metrics

- A numerical way to show how good your model is (besides Loss function)
- Most of the times is not optimizable directly
- Show some extra properties of your model that are not obvious from Loss function value

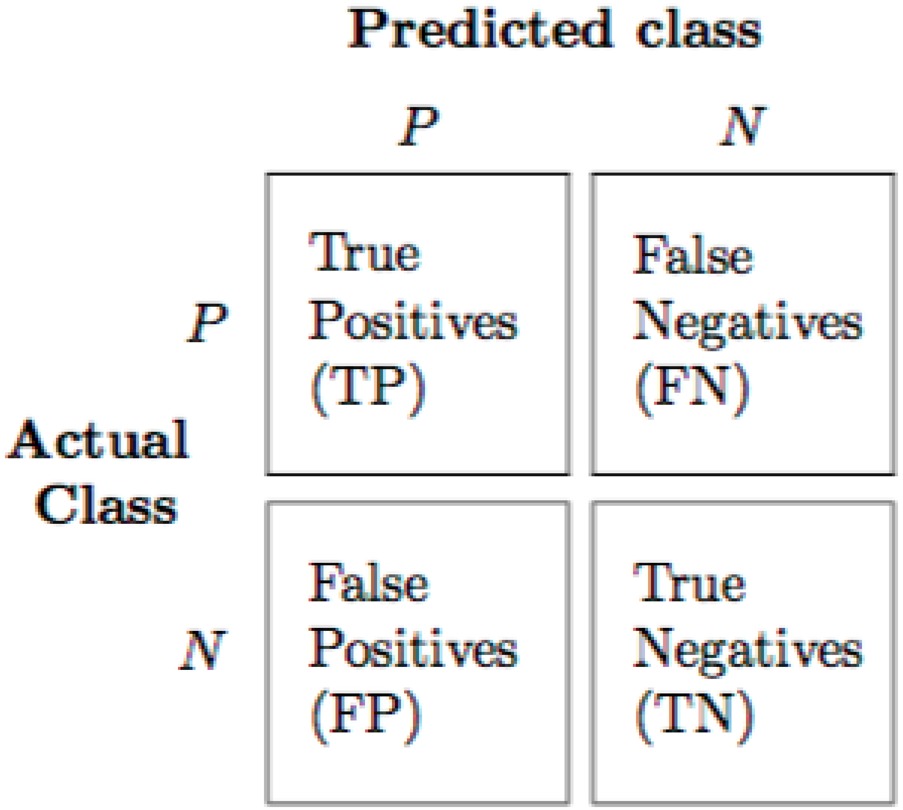
Accuracy

Easy:

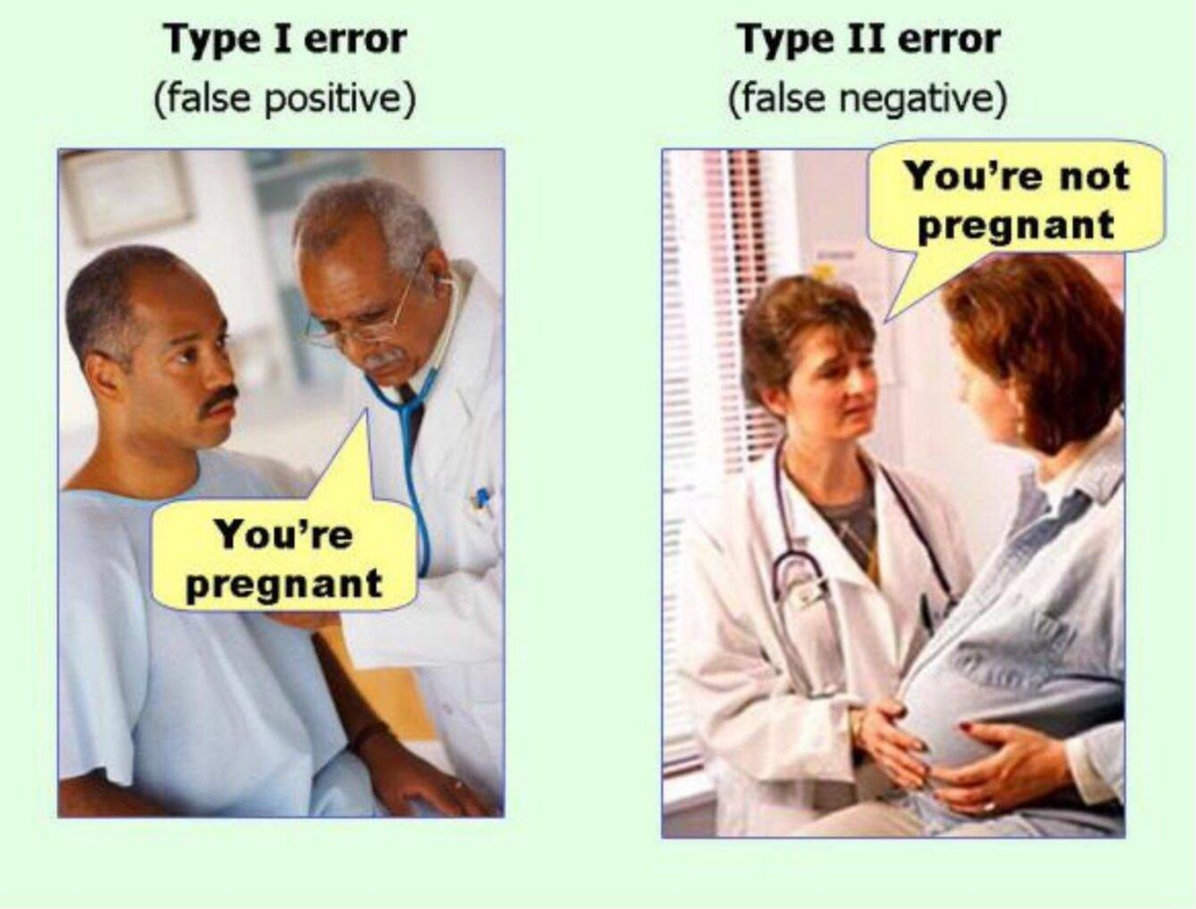
$$\frac{\#correct\ predictions}{\#samples}$$

Can be used for any classification task

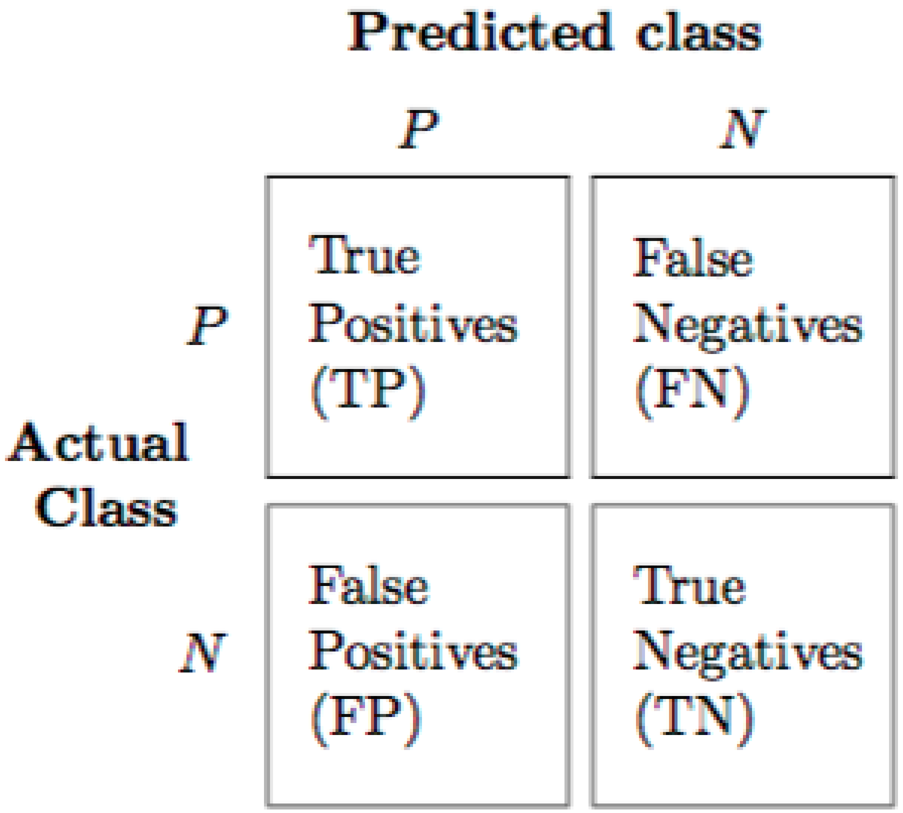
Errors in binary classification



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)



Precision, Recall

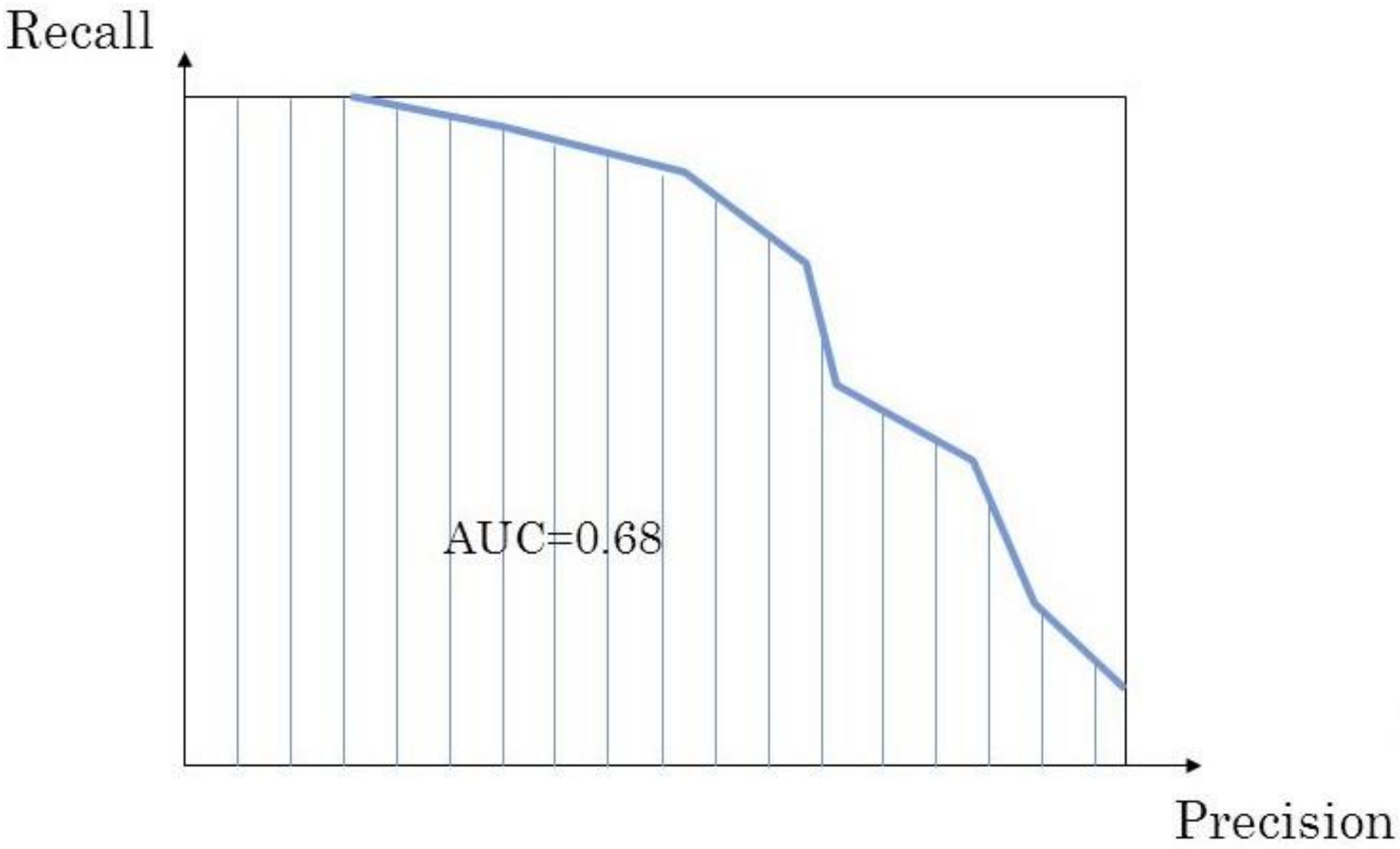


		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$precision = \frac{\mathrm{TP}}{TP + FP} = P(Y = 1|\hat{Y} = 1)$$

$$recall = \frac{TP}{TP + FN} = P(\hat{Y} = 1|Y = 1)$$

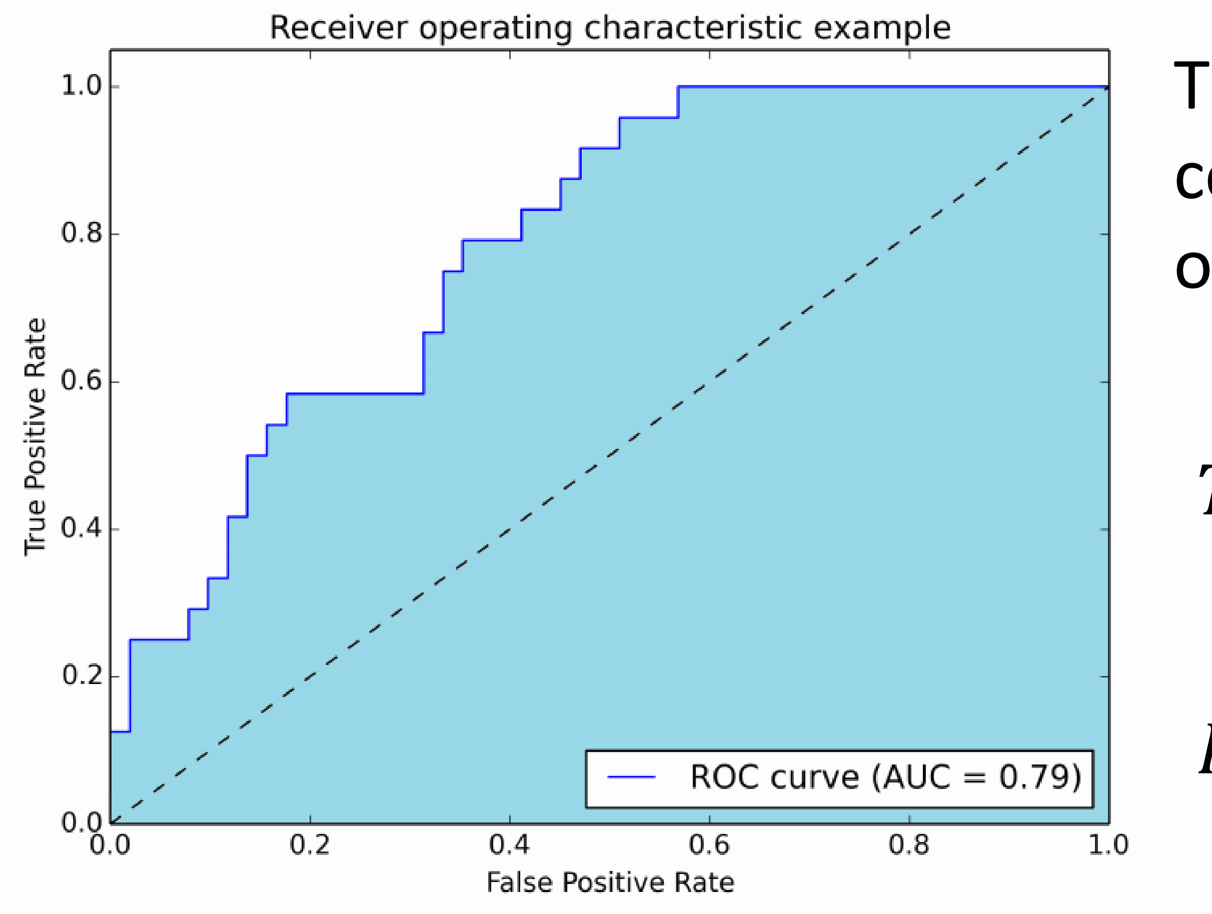
Precision Recall AUC (Area Under the Curve)



Changing threshold (default is 0.5) for decision making we chanage precision and recall

PR AUC shows how our model is sensitive to changes to the threshold in terms of precision and recall

ROC AUC



The idea is the same as in PR AUC – computing area under the curve obtained by changing threshold

$$TPR = \frac{TP}{TP + FN} = P(\hat{Y} = 1 | Y = 1)$$

$$FPR = \frac{FP}{FP + TN} = P(\hat{Y} = 0 | Y = 0)$$

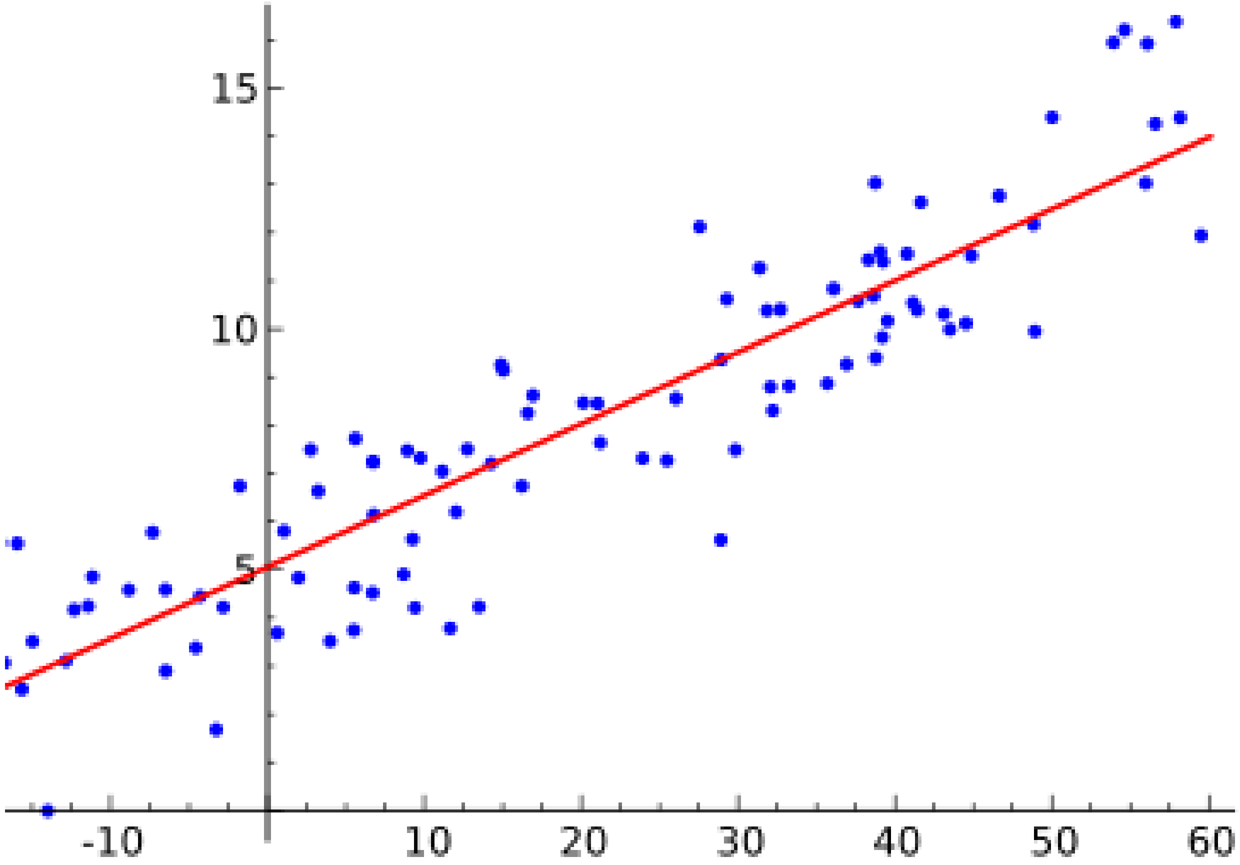
Regression Problem

Regression Problem

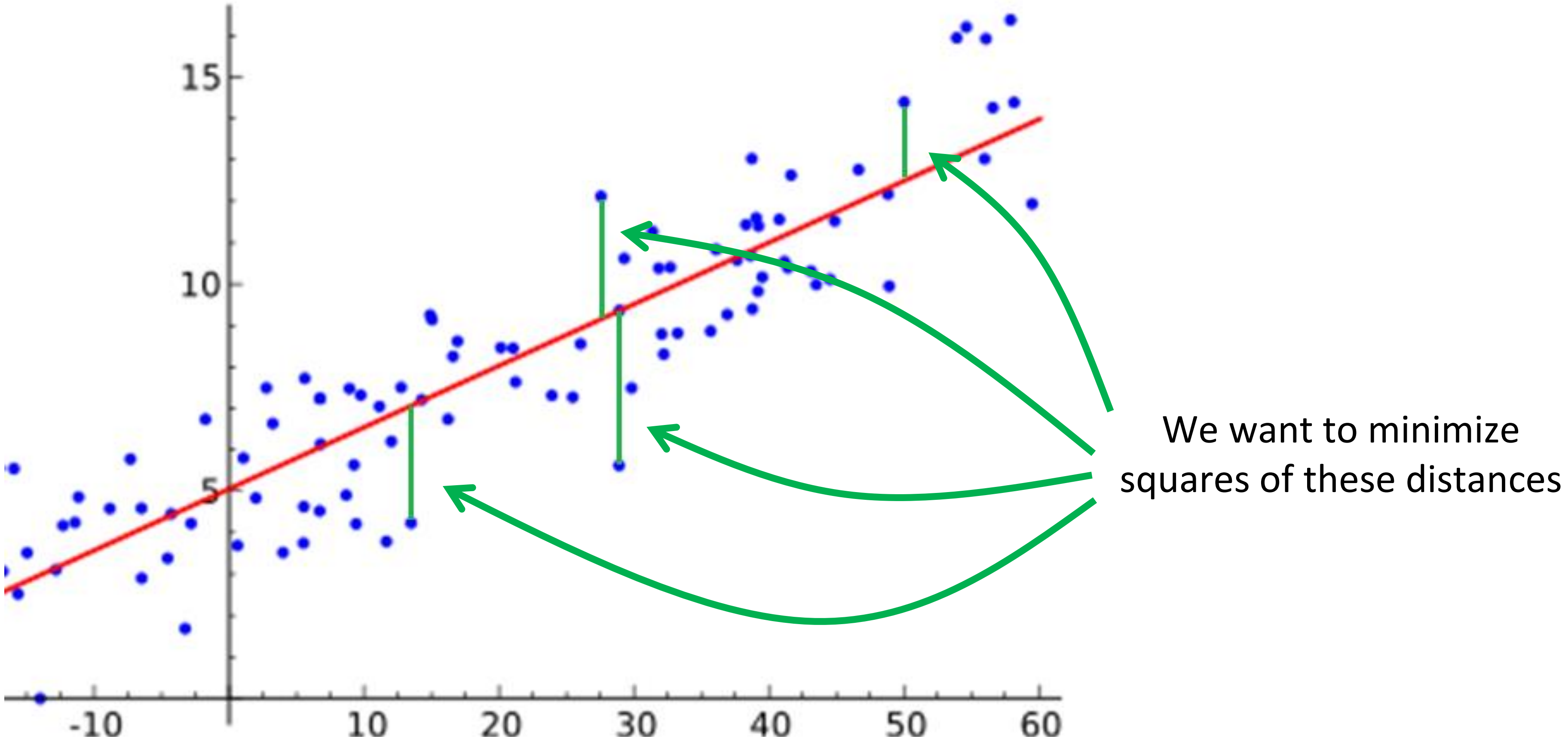
- Returning a real value given data
- Distinction from classification – not a finite set of possible results
- NN output can be unbounded

• Example 1:	predict tumor size
• Example 2:	predict age of a human
• Example 3:	predict number of cats on the photo
• Example 4:	predict price of the house

Least Squares Linear Regression

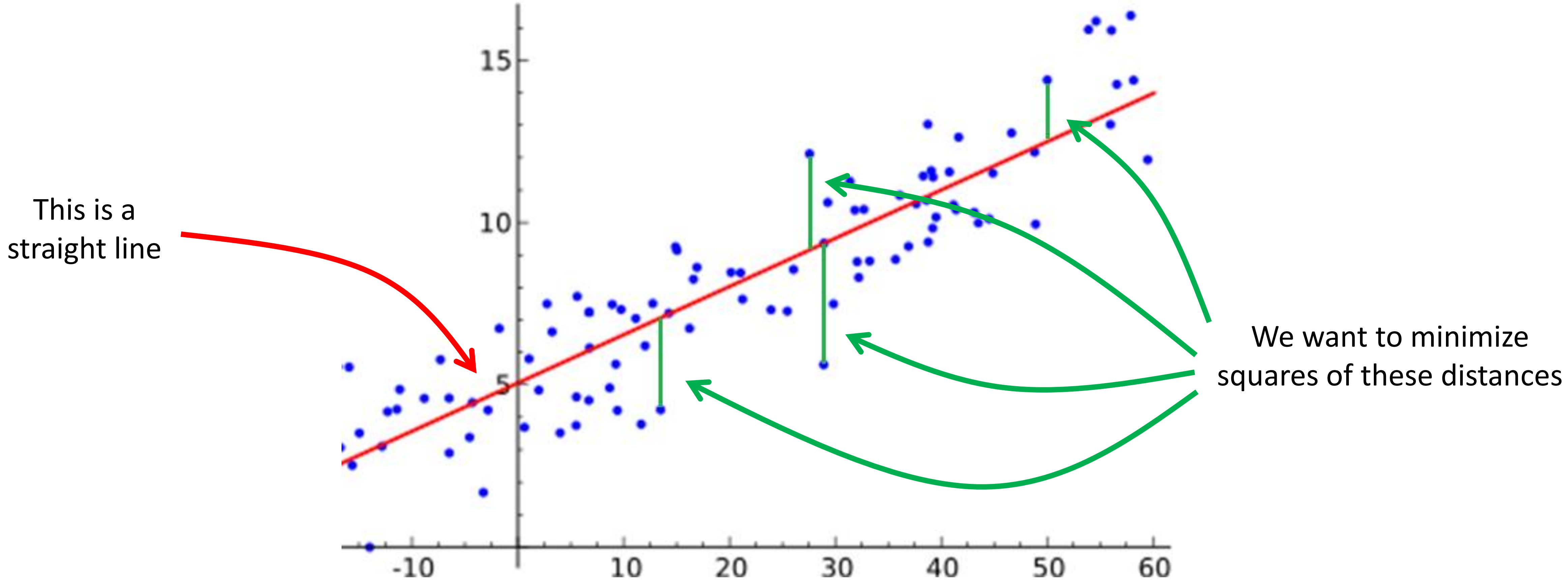


Least Squares Linear Regression



We want to minimize squares of these distances

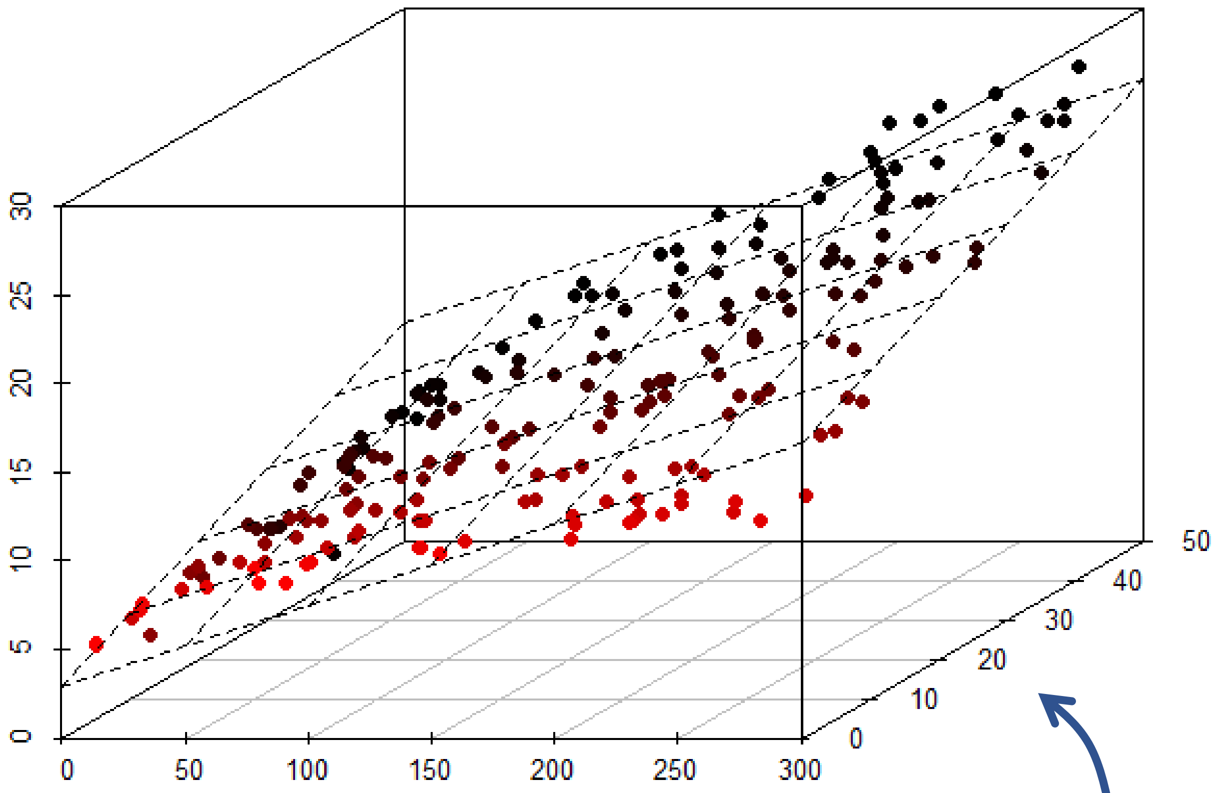
Least Squares Linear Regression



This is a straight line

We want to minimize squares of these distances

Least Squares Linear Regression in 3d



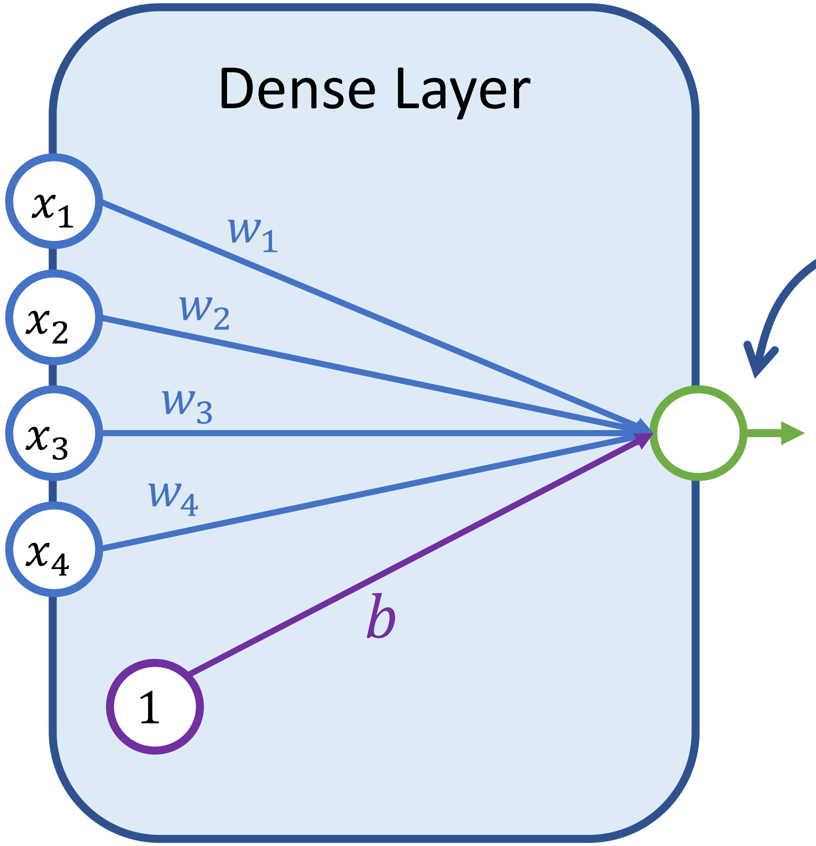
Parameter that we want to predict

Parameters that we know about the data

Least Squares Linear Regression

- We can create a linear model and train its *weights*
- Linear model: $f(x) = w_1 x_1 + \cdots + w_n x_n + b$
- Loss function: $L(y, f(x)) = \left(y - f(x)\right)^2$

Least Squares Linear Regression as a NN

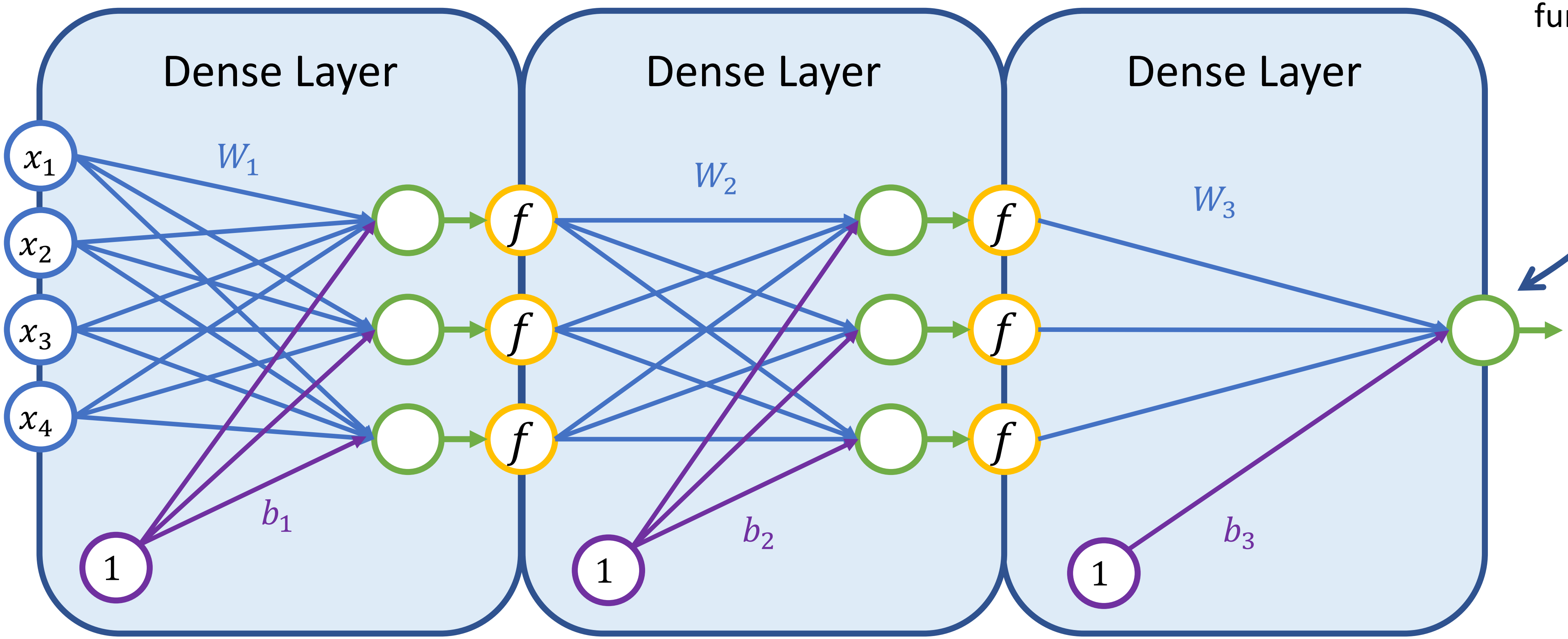


No activation function (non-linearity), thus the value is unbounded

Deeper Regression Models

- We can create a deep Neural Network for regression models similarly as we do for classification
- Absence of non-linearity at the end is not strict (you can bound your regression if you need)

Regression Problem via Deep NN



There is no activation function!

Loss Functions for Regression

The most popular ones:

- Mean Squared Error
- Mean Absolute Error
- Huber Loss

This is It For the Third Lecture