Women Techmakers

Powered by GDG Bangalore

IWD Extended Summit

Machine Learning with TensorFlow

Who are we?

Anisha Mascarenhas
Software Engineer at LinkedIn

Mipsa Patel
Software Engineer at LinkedIn

Agenda

1. Introduction to ML and Neural Networks

2. Codelab: Sign Language Recognition using TensorFlow

Introduction to ML and Neural Networks

01

What is AI?

At the core of every computer program there is a mathematical function at work. It could be as simple as computing the interest on an outstanding loan or as complex as flying an aircraft on autopilot. *Artificial Intelligence*, or *AI*, is a generic name for a computer program whose core mathematical function has been created (almost) automatically; and *Machine Learning*, or *ML*, refers to a collection of techniques which offer ways of creating AI.

Namit Chaturvedi
(PhD in theoretical computer science,
Applied Research Engineer at LinkedIn)

Types of Machine Learning Problems

Machine Learning

Supervised

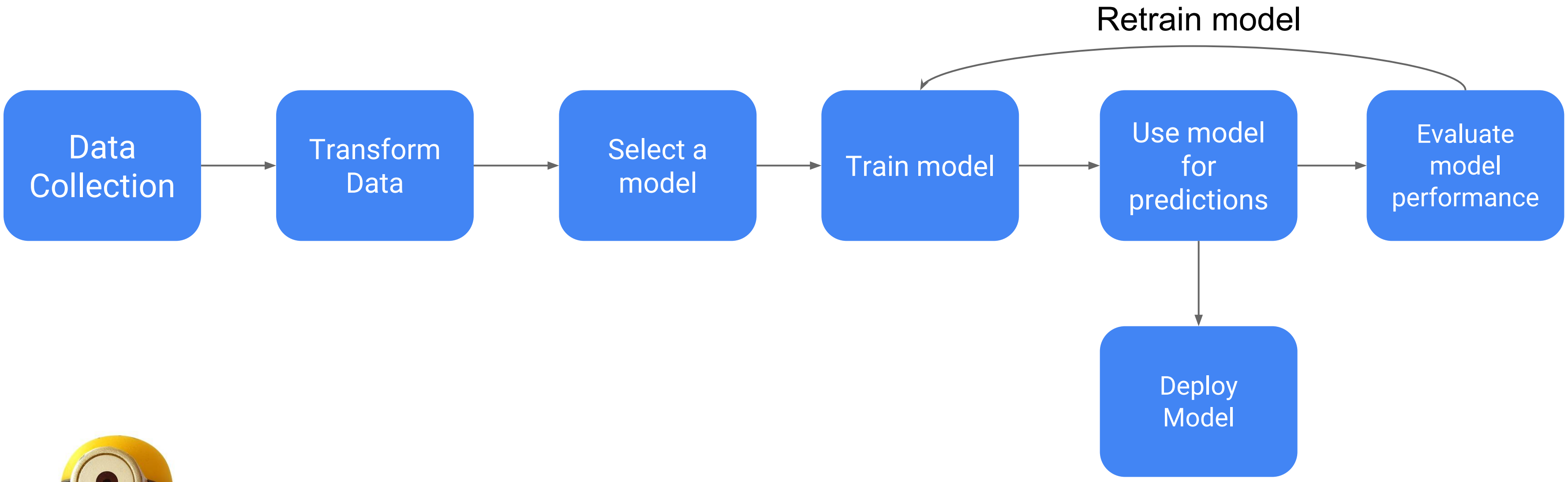
- Classification
- Regression

Unsupervised

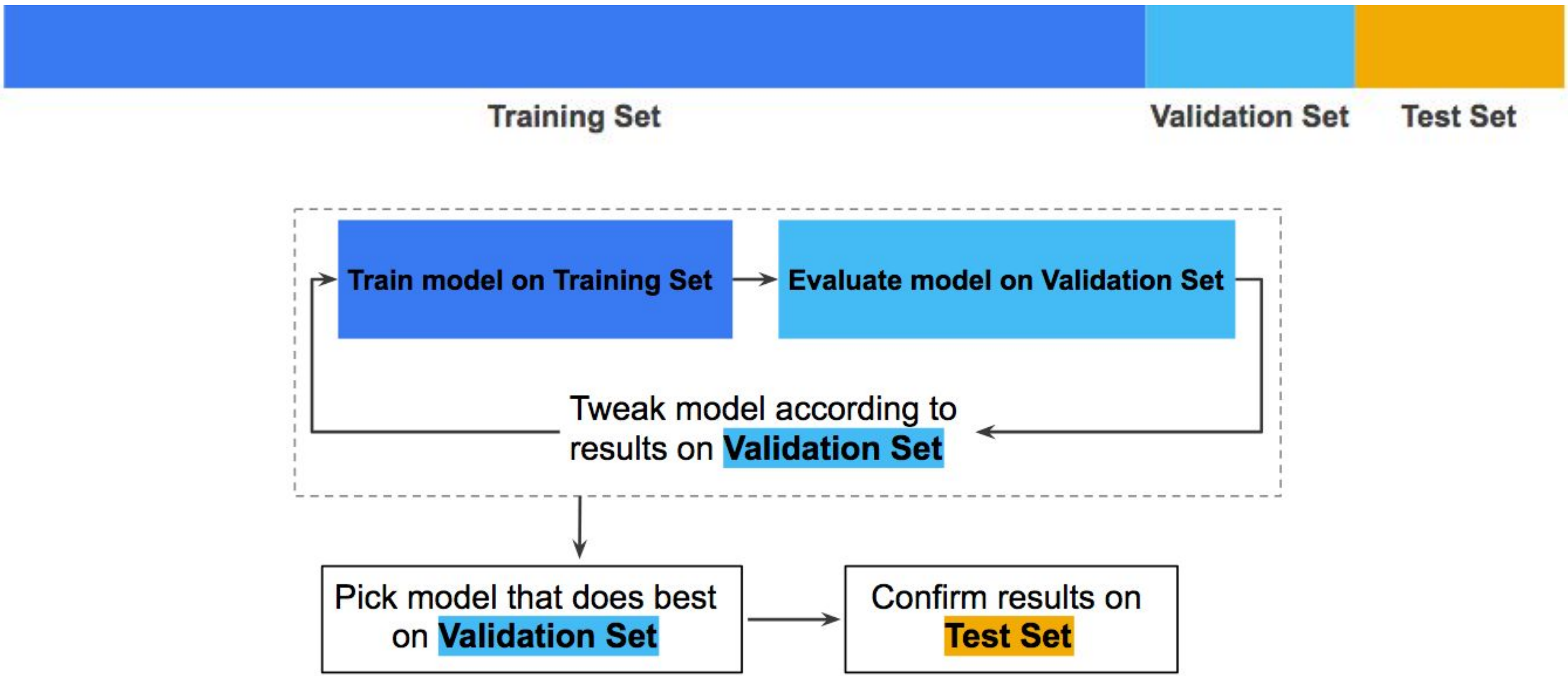
- Clustering
- Association Mining

Reinforcement

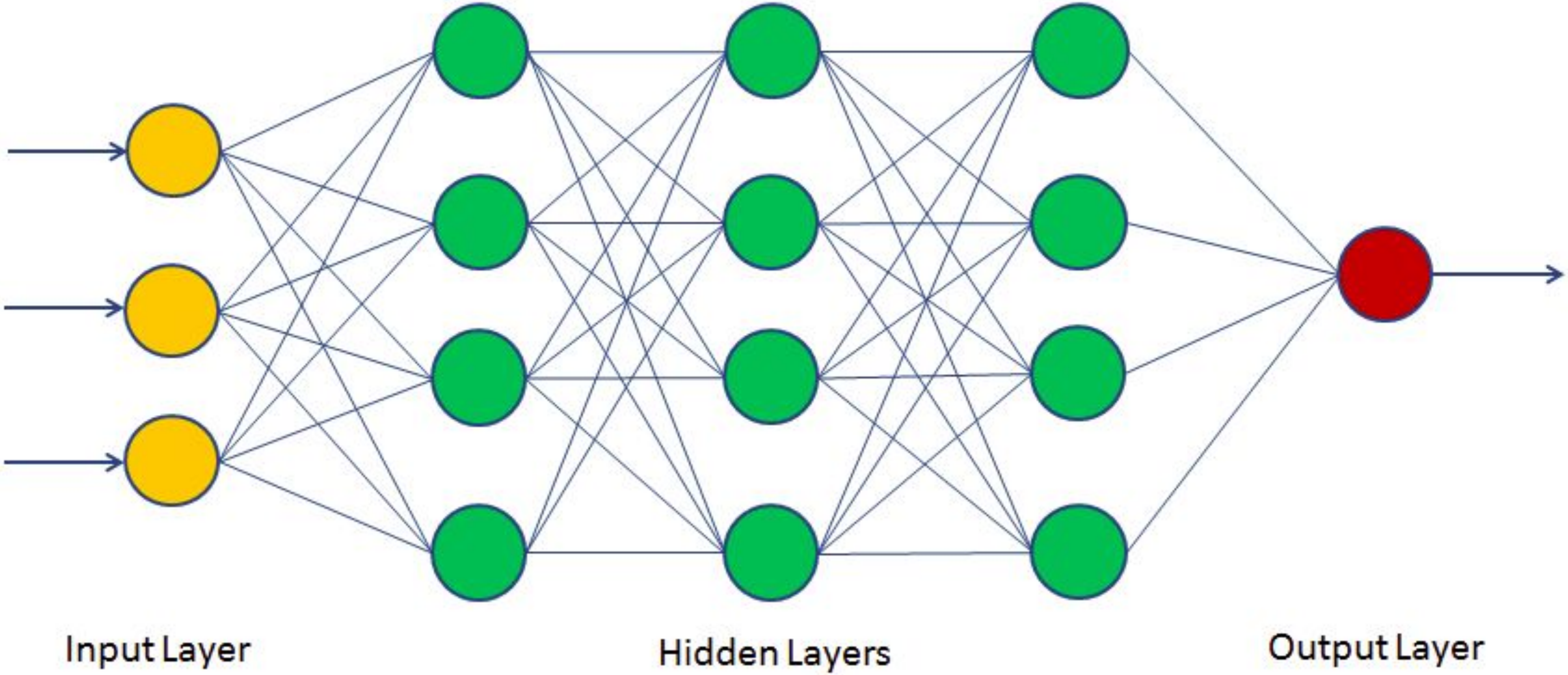
End to End ML Pipeline



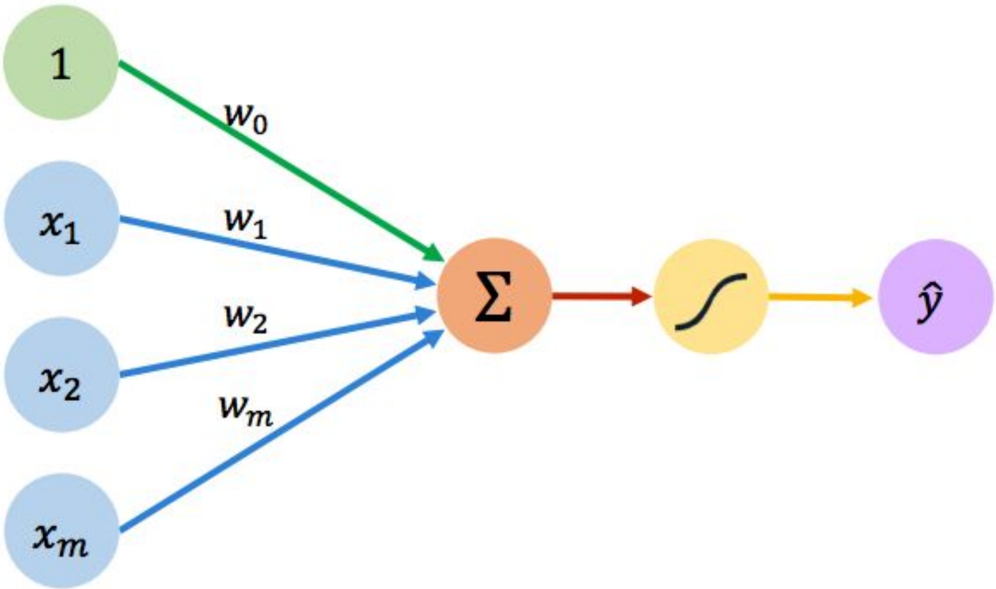
Preparing your dataset:



Neural Network



The Perceptron



Linear combination of inputs

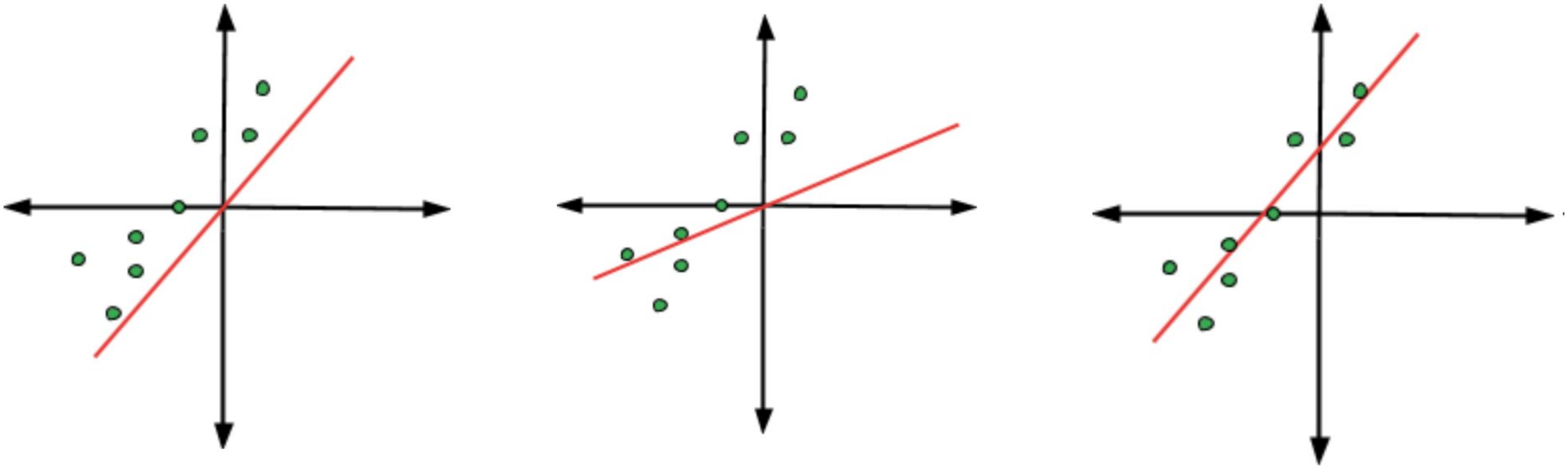
Output

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

Non-linear activation function

Bias

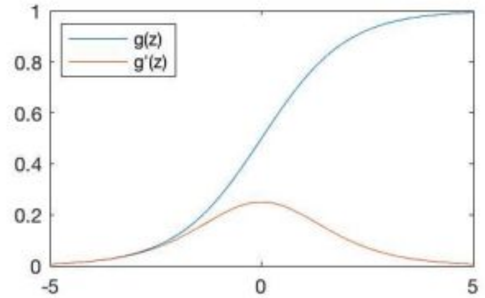
Bias



$y = w_1 x$ $y = w_2 x$ $y = w_0 + w_1 x$

Activation Functions: Introduce Non-Linearity

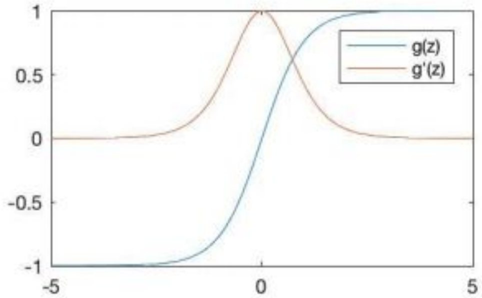
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

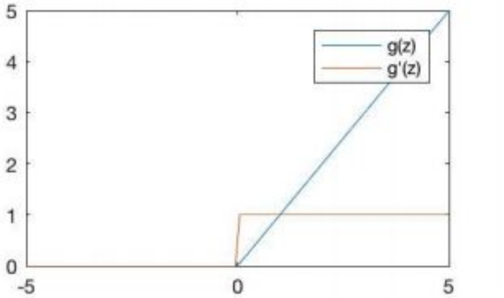
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

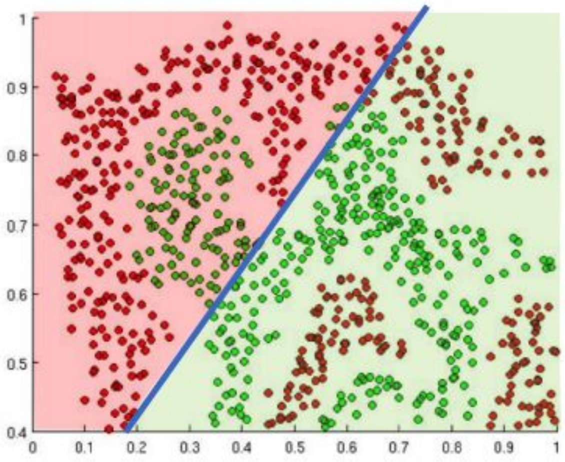
Rectified Linear Unit (ReLU)



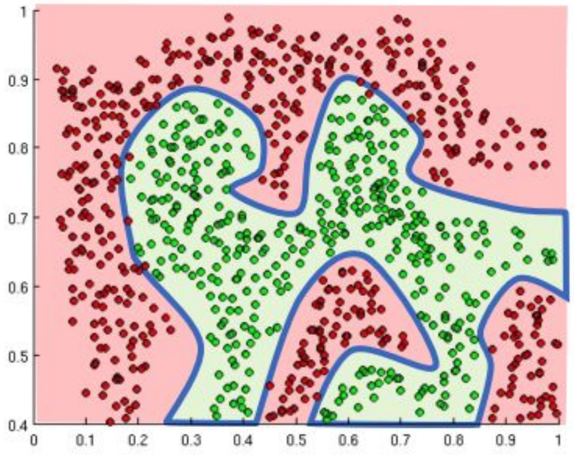
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Activation Functions: Introduce Non-Linearity



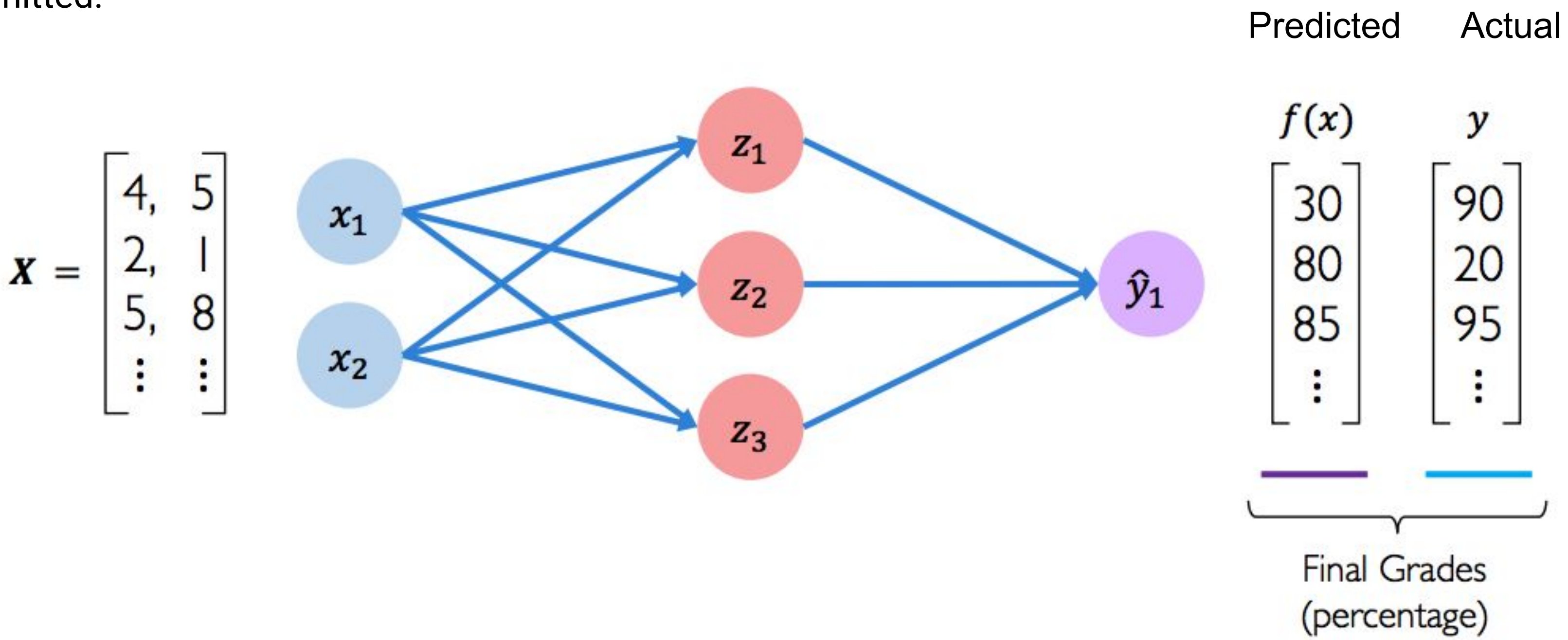
Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Example Neural Network

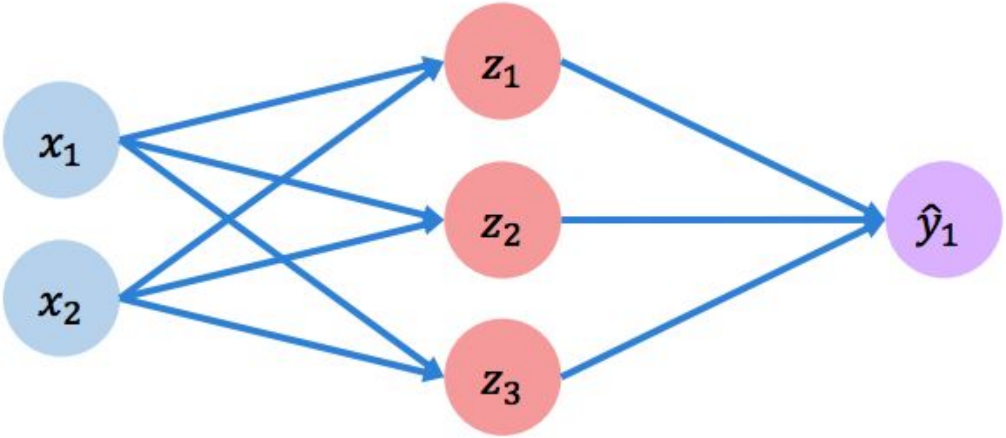
For example: Predicting the final exam score (on 100) of a student given features like number of lectures attended, and number of assignments submitted.



Mean Squared Loss

For example: Predicting the final grade of a student given features like number of lectures attended, and number of assignments submitted.

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



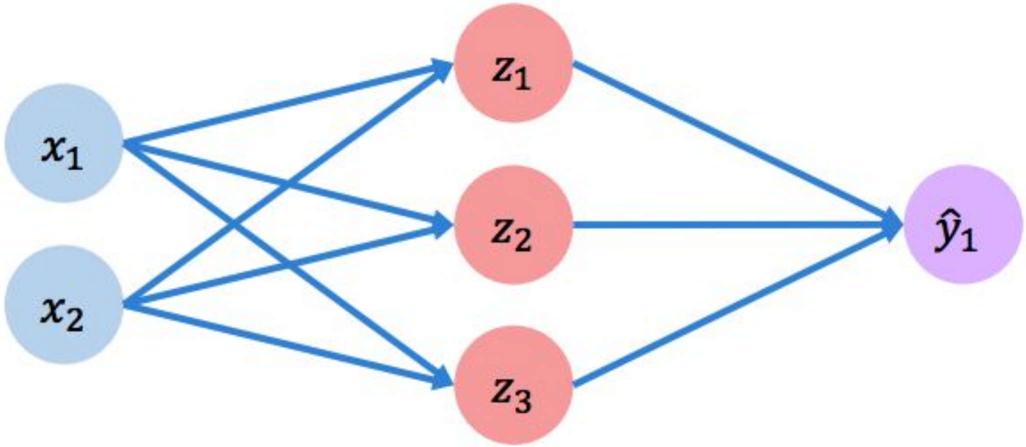
$$\underbrace{\begin{array}{cc} \text{Predicted} & \text{Actual Score} \\ f(x) & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}}_{\text{Final Grades (percentage)}}$$

$$\boldsymbol{J}(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)^2$$

Cross-Entropy Loss

For example: Classifying whether a student will pass or not given features like number of lectures attended, and number of assignments submitted.

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$

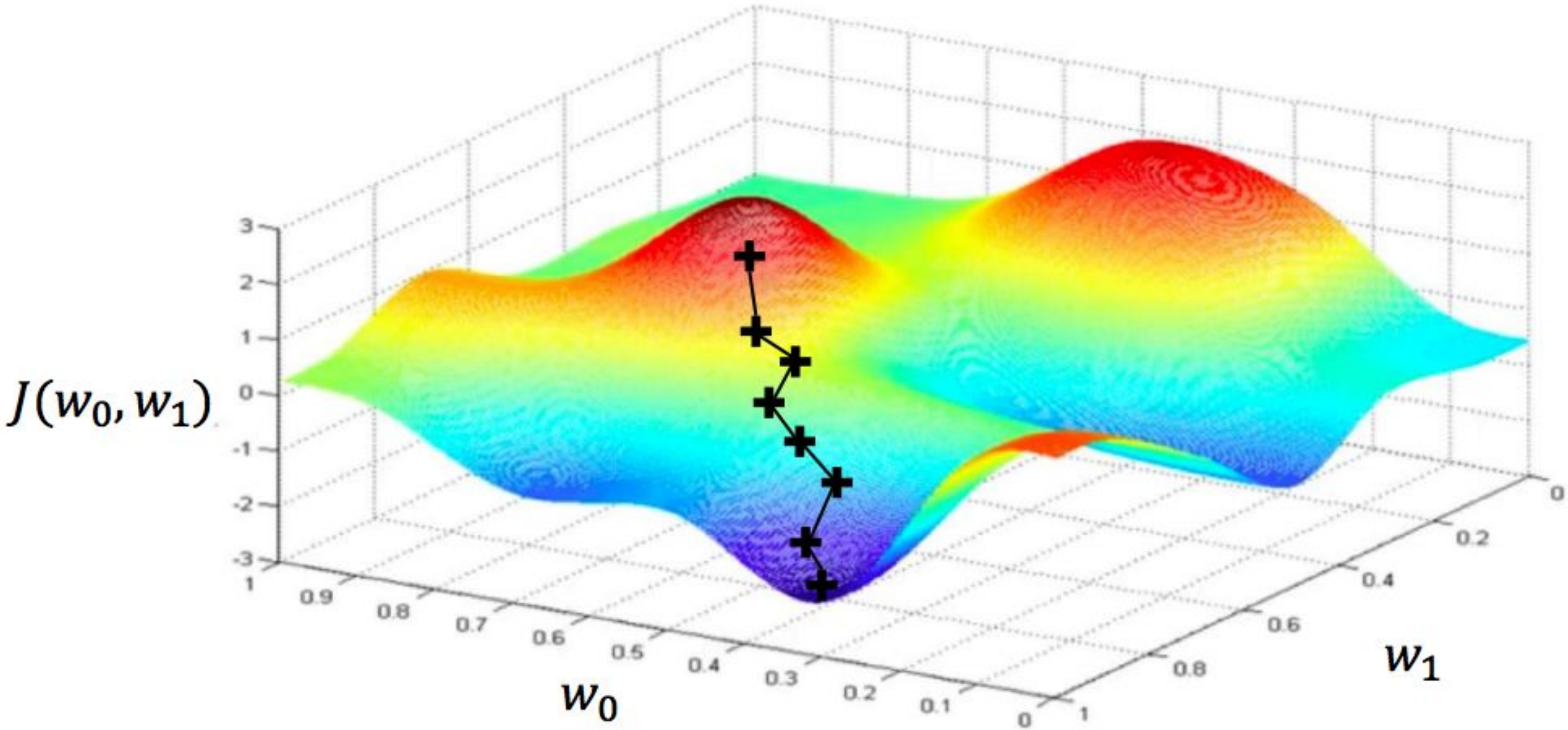


Predicted	Actual Score
$f(x)$	y

$$f(x) = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1-\underbrace{y^{(i)}}_{\text{Actual}})\log\Big(1-\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

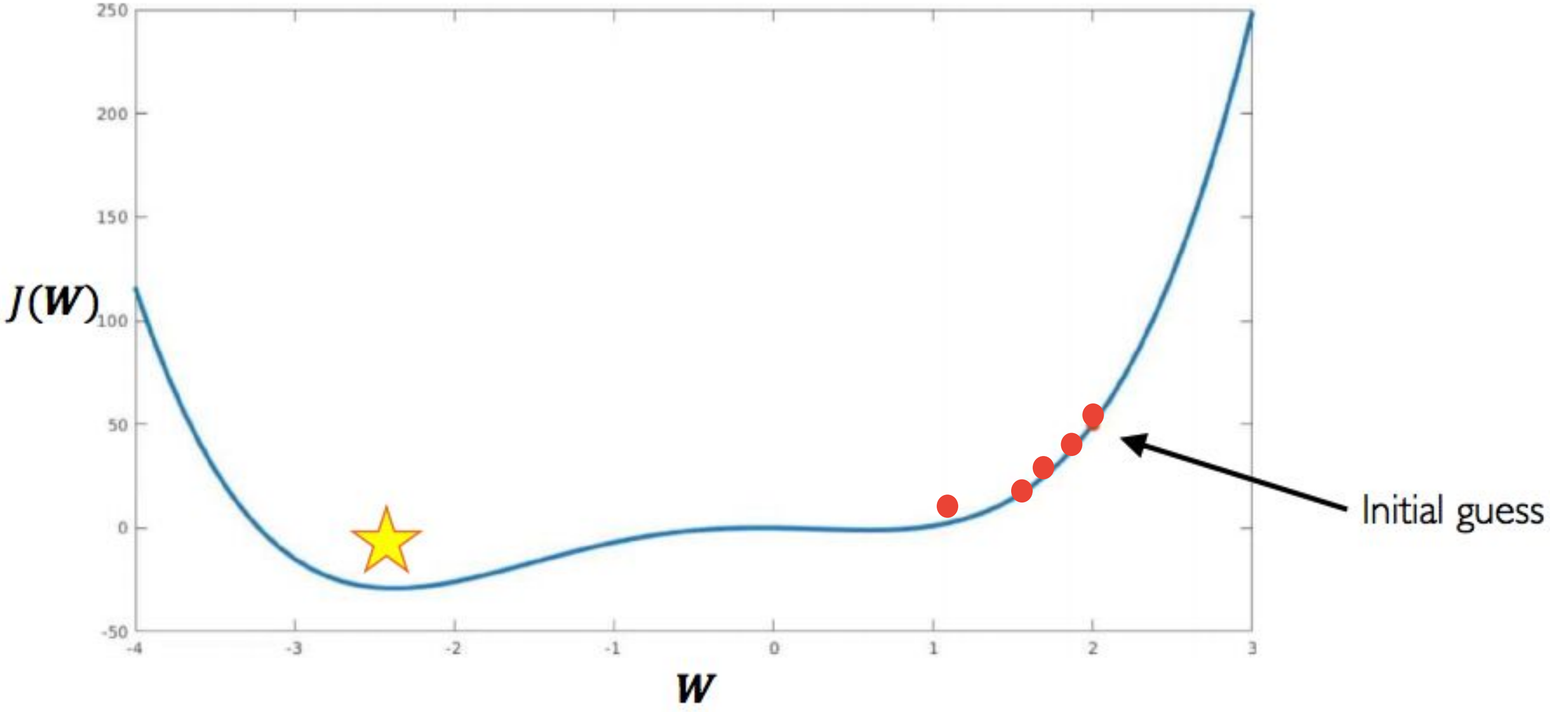
Visualizing our Loss Function



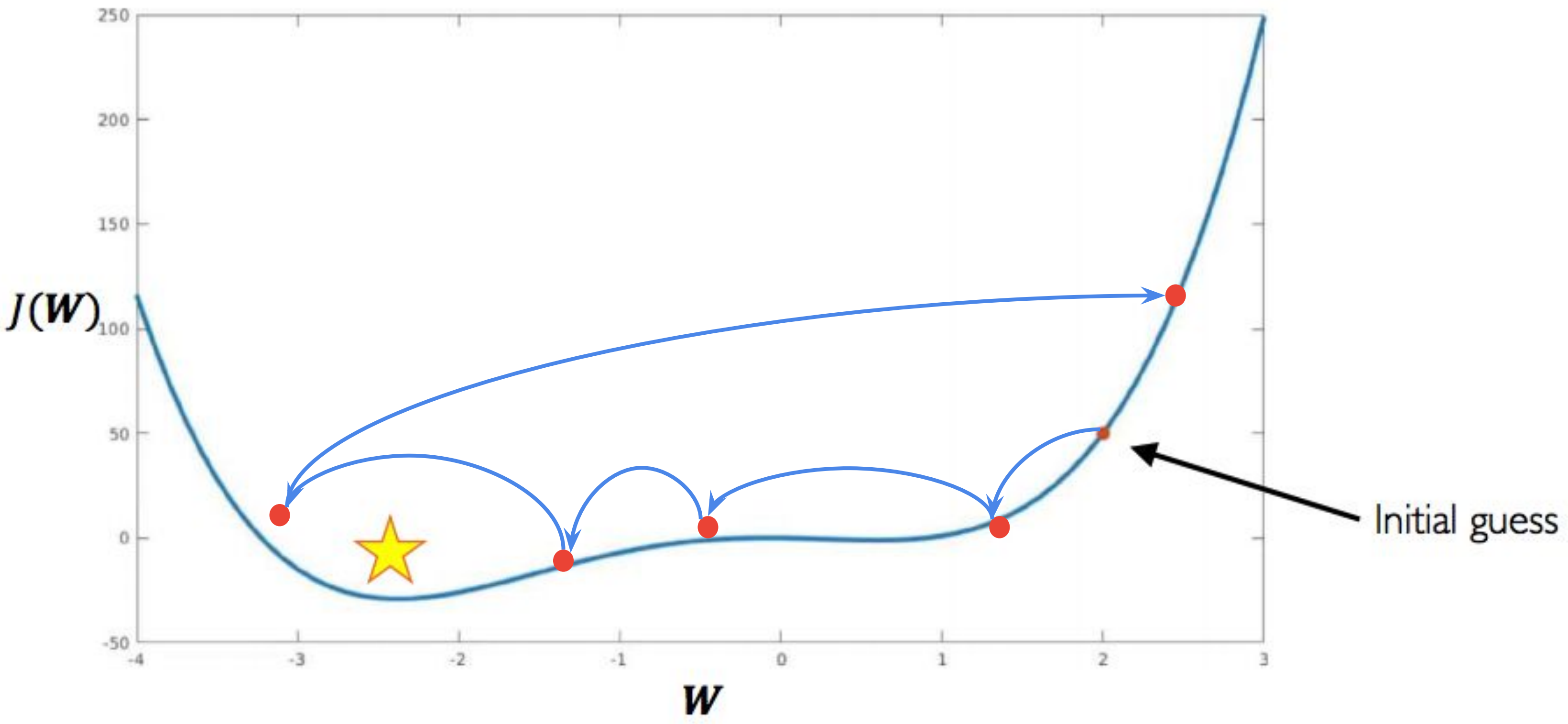
Gradient Descent Algorithm

1. Initialize weights randomly $\sim\mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

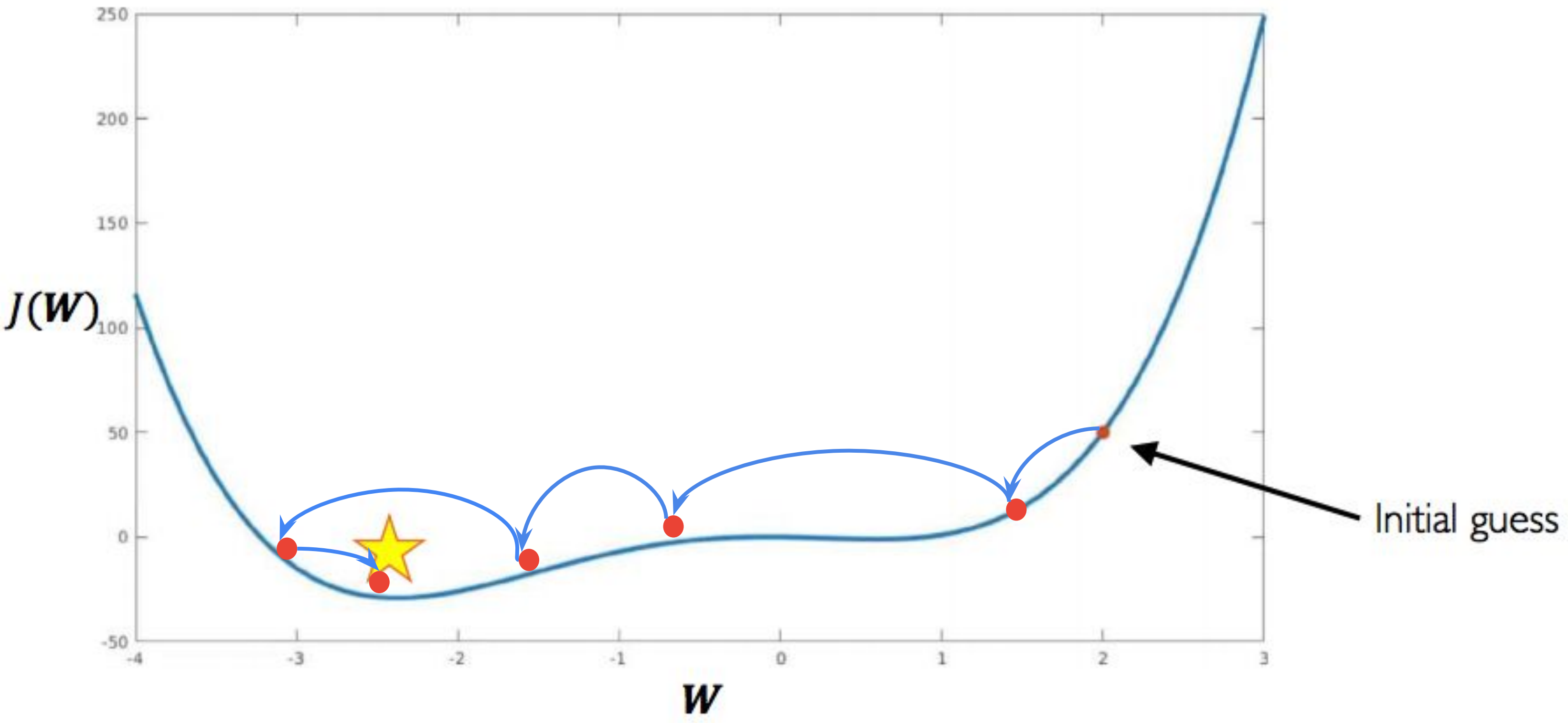
Low Learning Rate



High Learning Rate



Good Learning Rate



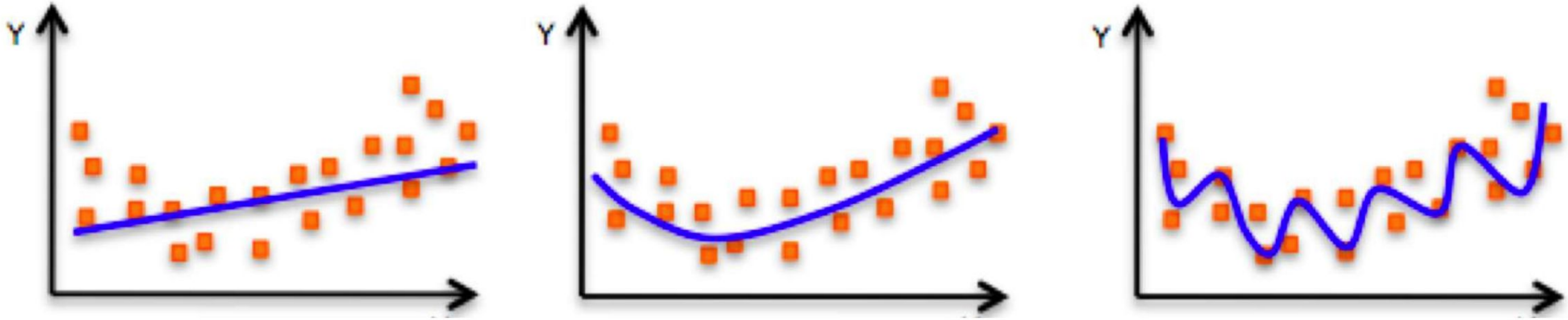
Adaptive Learning Rate Algorithms

- Momentum
- Adagrad
- Adadelta
- Adam
- RMSProp

Batch Size and Epochs

1. Initialize weights randomly $\sim\mathcal{N}(0, \sigma^2)$
2. Loop
3. Pick single data point $\boldsymbol{i}$
4. Compute gradient, $\frac{\partial J_i(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
6. Return weights

Problem of Overfitting



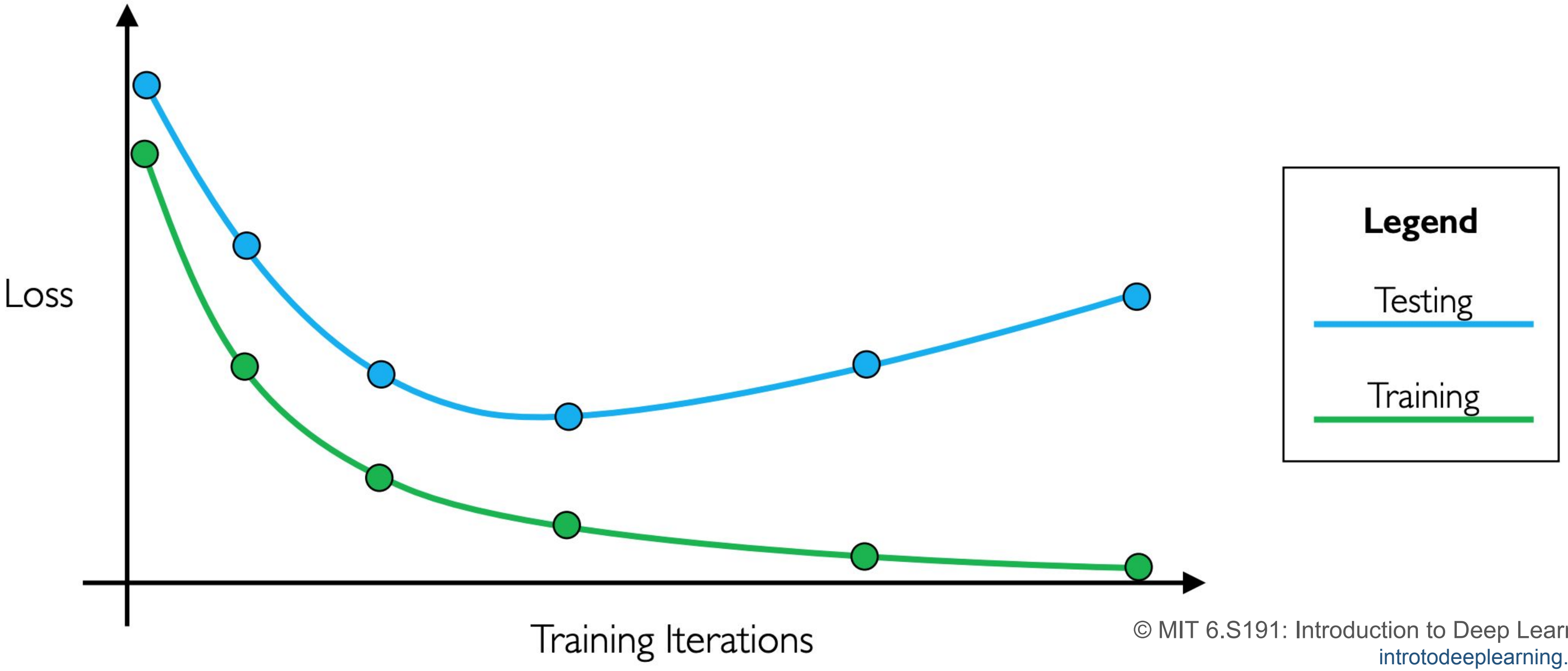
Underfitting
Model does not have capacity to fully learn the data

← **Ideal fit** →

Overfitting
Too complex, extra parameters, does not generalize well

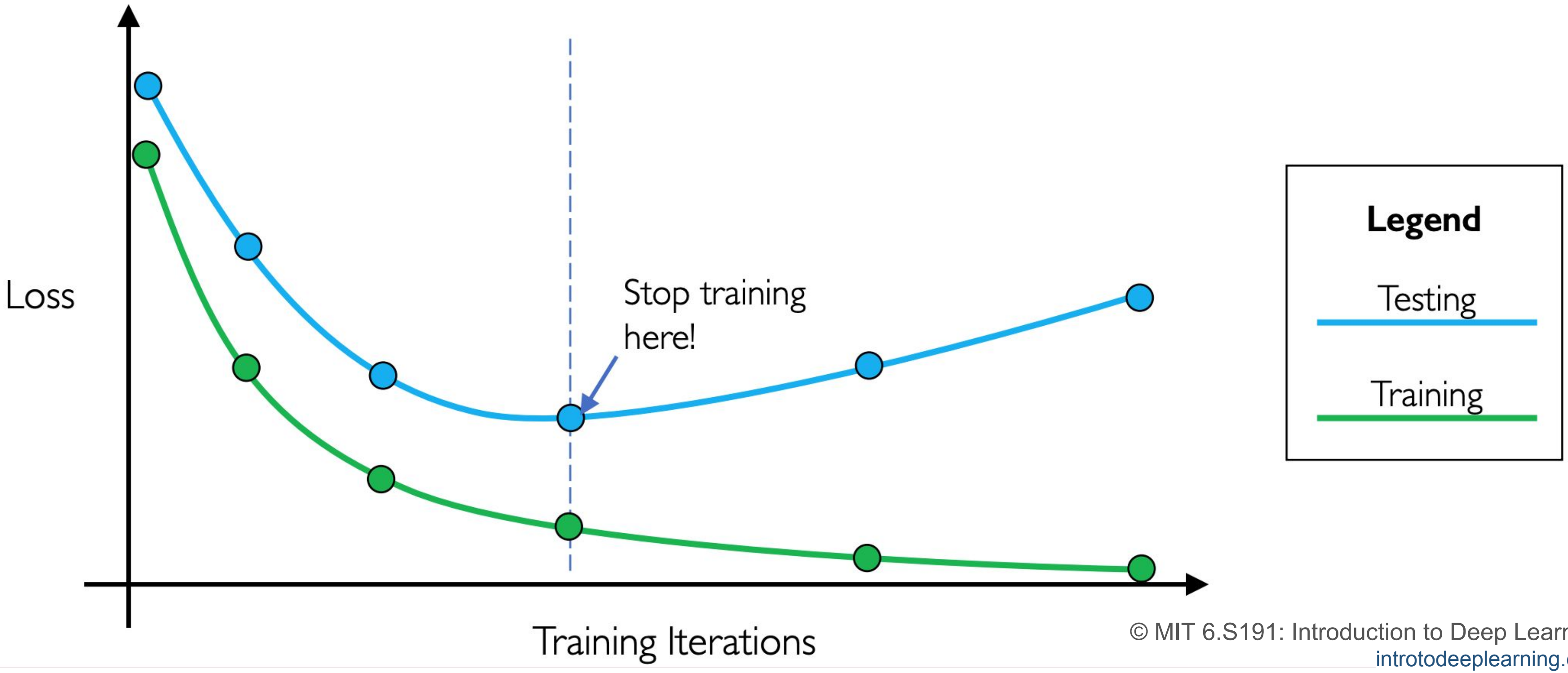
Early Stopping

Stop training before we start overfitting.



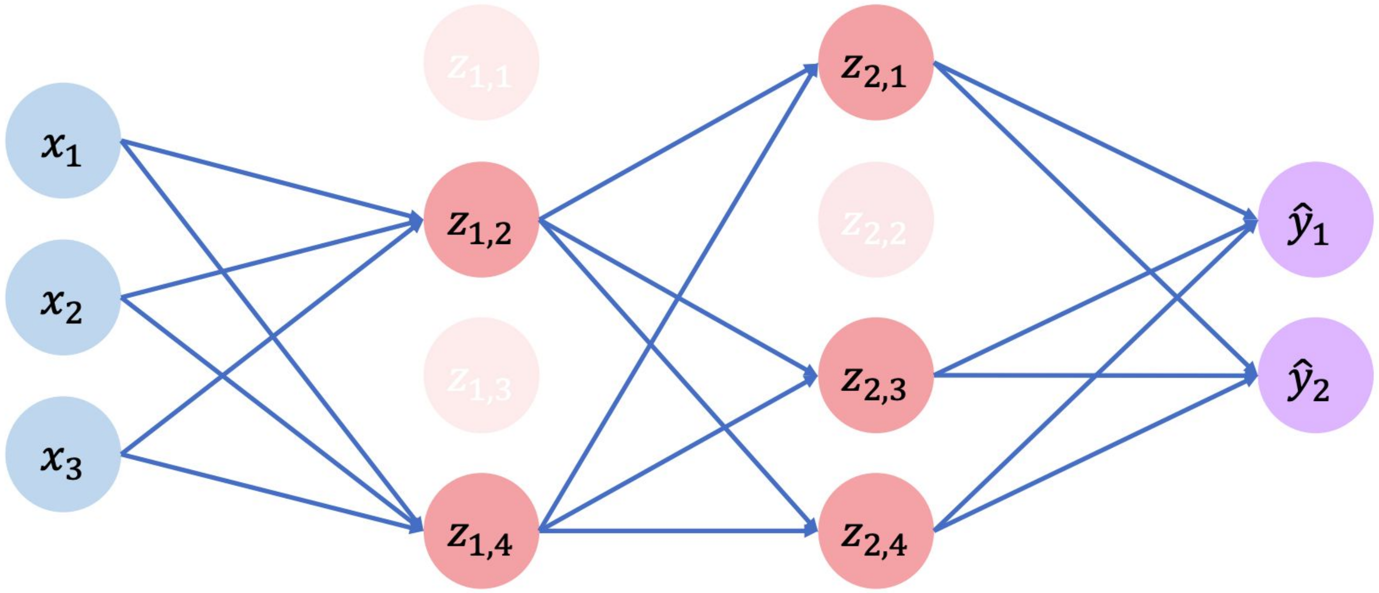
Early Stopping

Stop training before we start overfitting.



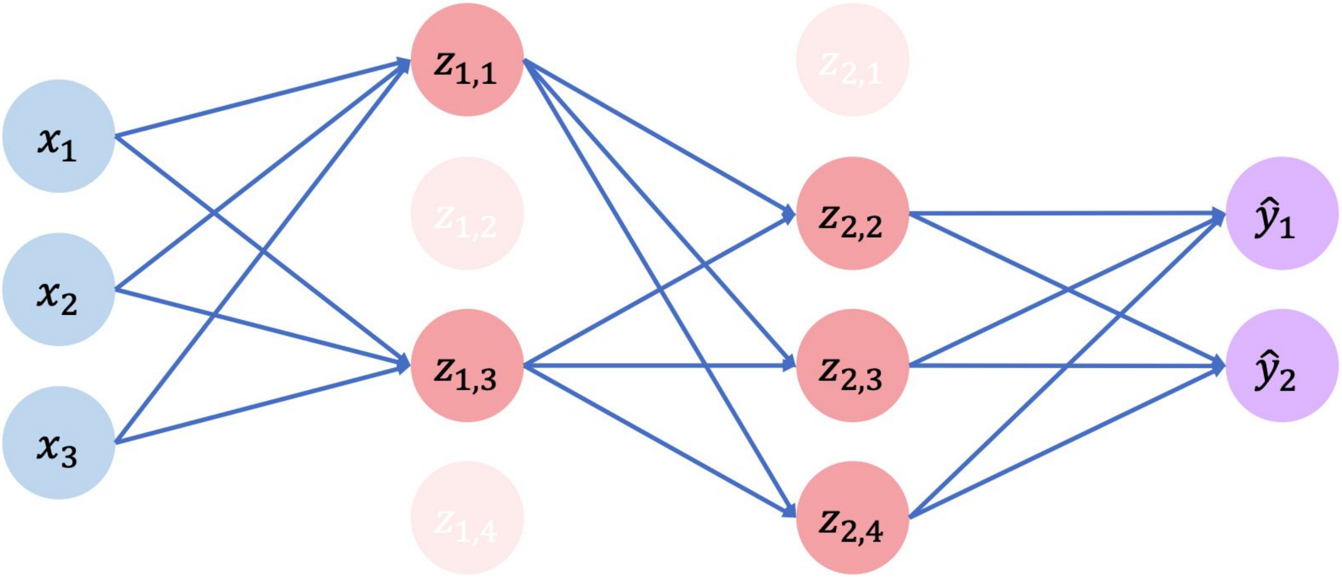
Dropouts

Randomly set some activations to 0



Dropouts

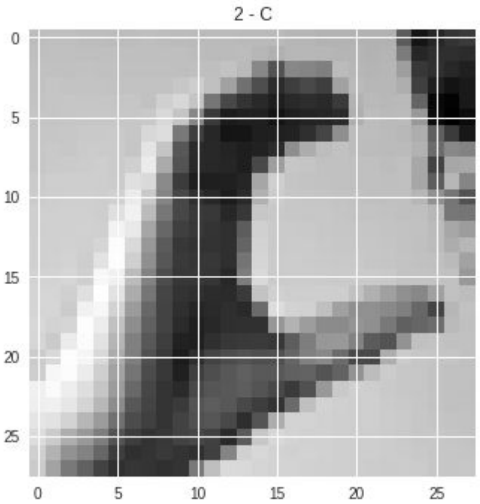
Randomly set some activations to 0



Hands On: Solving a Classification Problem using TensorFlow

Demo

Recognizing Sign Language from Images



Input Image

```
187 188 188 187 187 ....... 28 columns
188 189 189 188 188
...
...
...
191 193 192 192 192 ....... 28 rows
```

Pixel Representation
of 28 x 28 image

$$\begin{array}{c} A \\ B \\ C \\ D \\ \ldots \end{array} \begin{bmatrix} 0.01 \\ 0.003 \\ 0.98 \\ 0.02 \\ \end{bmatrix}$$

Classification Result

Output of the model produces a probability score for the image belonging to a particular class.

http://bit.ly/iwdcolab

Any Questions?

Slides, Code and Links

can be found at github.com/anisham197/WTMExtendedSummit/

Contact Us:

LinkedIn:mipsapatel

Linkedin:anishamascarenhas

Twitter: anisham197

References

- https://ai.google/education
- https://developers.google.com/machine-learning/
- https://research.fb.com/the-facebook-field-guide-to-machine-learning-video-series/
- http://introtodeeplearning.com/#schedule
- ttp://d2l.ai/
- https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- http://ruder.io/optimizing-gradient-descent/