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 Machine Learning
2. Overview of Neural Networks
3. Codelab: Sign Language Recognition
4. Quirks of Real World AI Development

01

Introduction to Machine Learning

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)

AI can only be as good as the examples and techniques used to train it

Thinking about a problem from a ML Perspective: From programs to experiments

Step	Example
1. Set the research goal.	I want to predict how heavy traffic will be on a given day.
2. Make a hypothesis.	I think the weather forecast is an informative signal.
3. Collect the data.	Collect historical traffic data and weather on each day.
4. Test your hypothesis.	Train a model using this data.
5. Analyze your results.	Is this model better than existing systems?
6. Reach a conclusion.	I should (not) use this model to make predictions, because of X, Y, and Z.
7. Refine hypothesis and repeat.	Time of year could be a helpful signal.

Identifying good problems for ML

Start with the problem, and not the solution

YOU GET A MACHINE LEARNING
EVERYONE GETS A MACHINE LEARNING
YOU GET A MACHINE LEARNING
MEMECAPTAIN

Identifying good problems for ML: Aim to make decisions, not just predictions.

"I trained a model that predicts the probability that someone will want to watch a video and still click "thumbs down" on youtube!"

When is traditional computing better than machine learning?

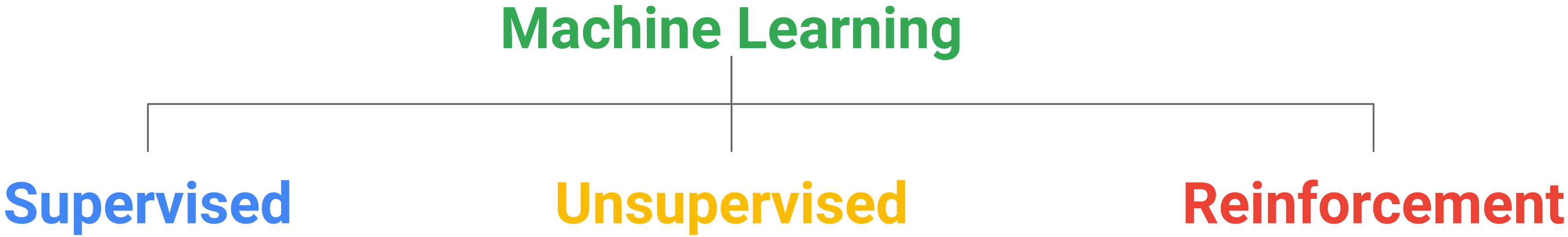
Not enough data

Noisy Data

No time & money

Simple problem to solve

Types of Machine Learning Problems



Machine Learning

Supervised

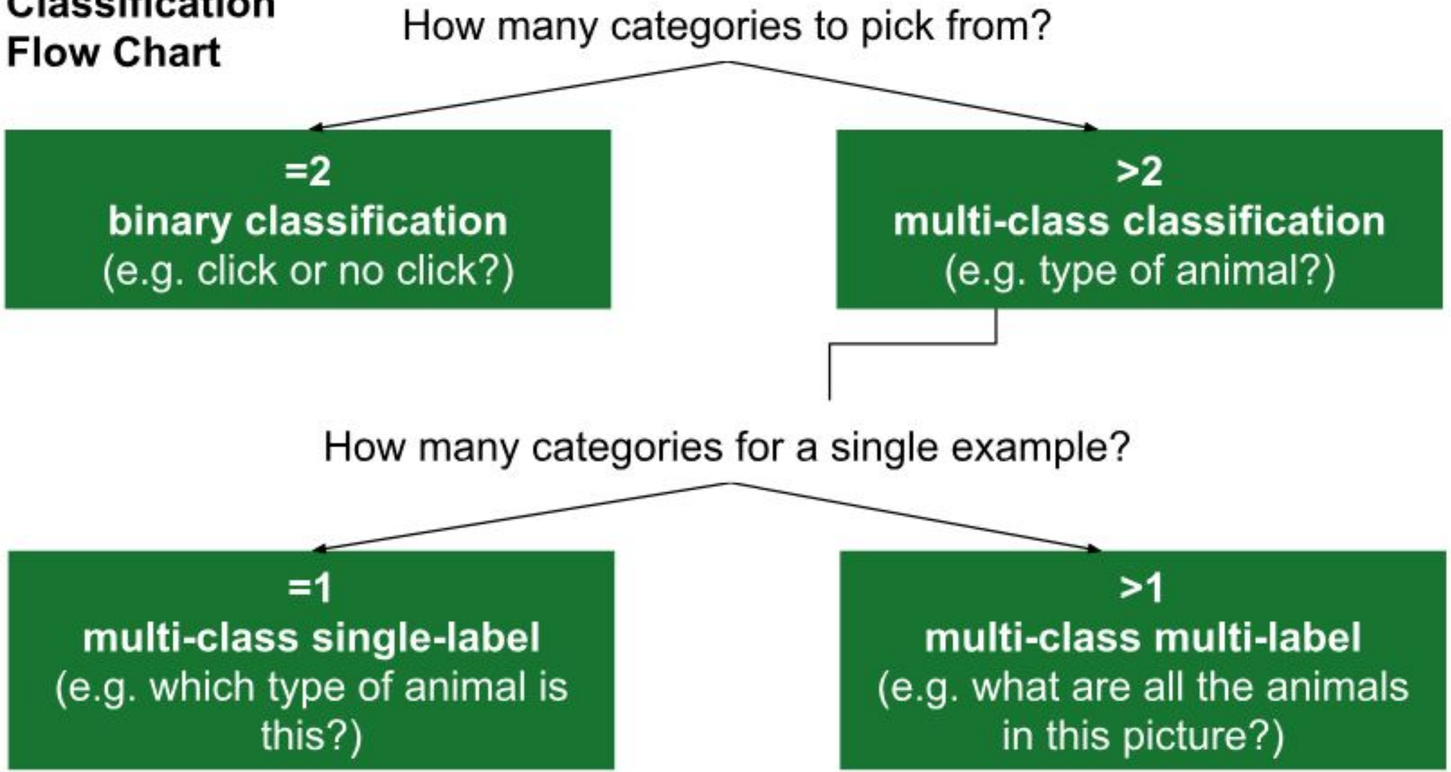
- Classification
- Regression

Unsupervised

- Clustering
- Association Mining

Reinforcement

Classification Flow Chart



How many categories to pick from?

=2
binary classification
(e.g. click or no click?)

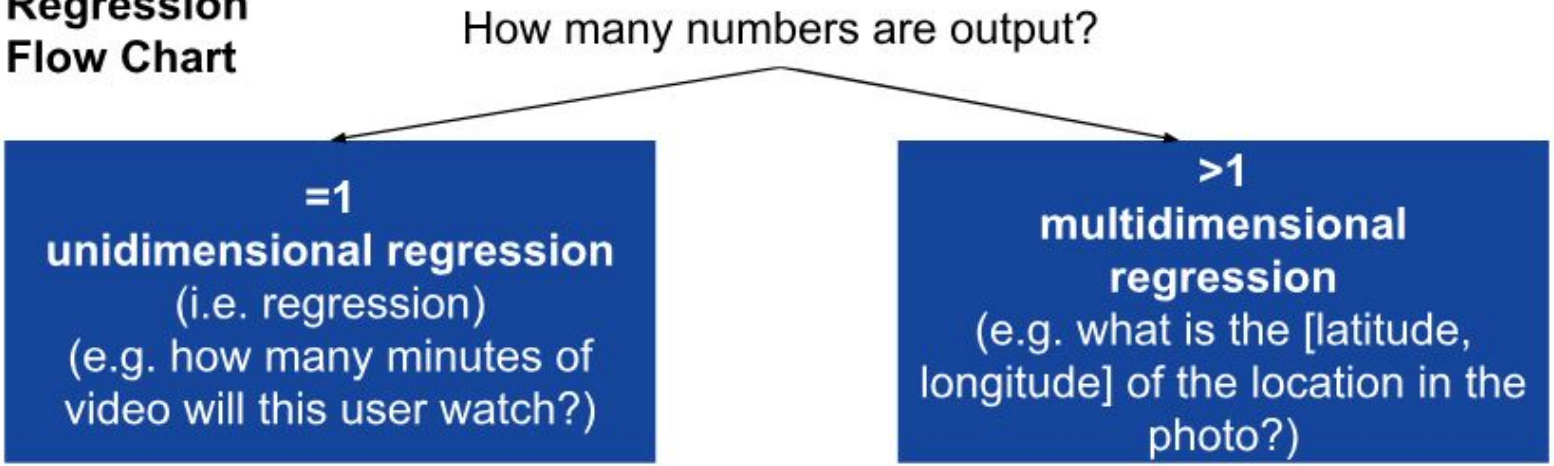
>2
multi-class classification
(e.g. type of animal?)

How many categories for a single example?

=1
multi-class single-label
(e.g. which type of animal is this?)

>1
multi-class multi-label
(e.g. what are all the animals in this picture?)

Regression Flow Chart

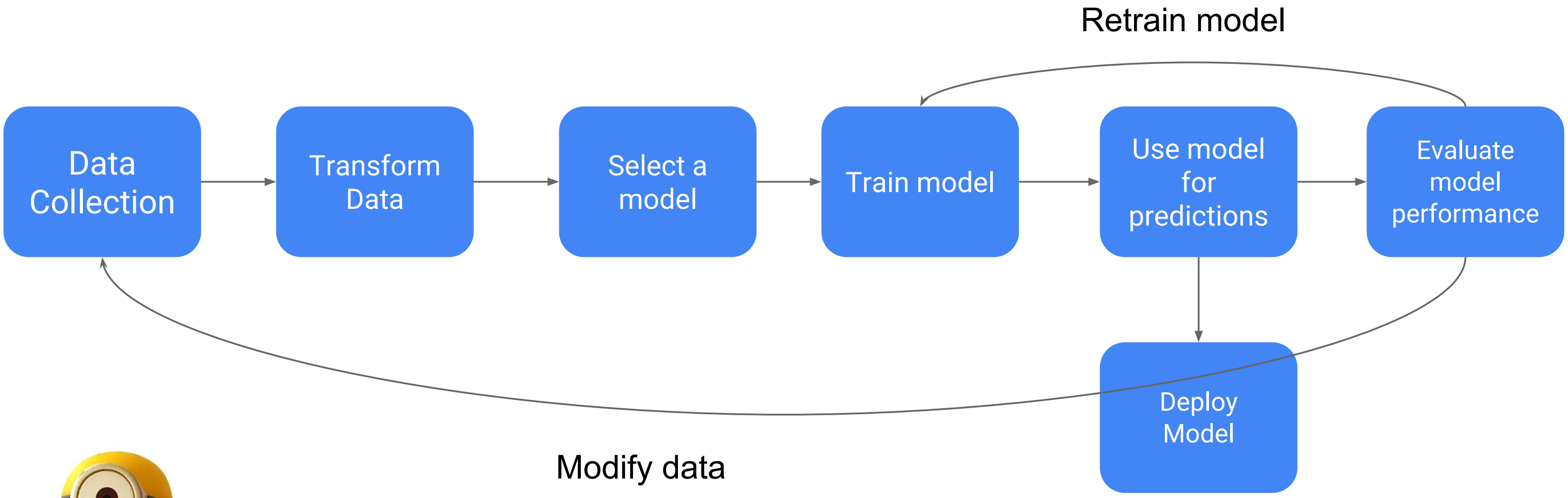


How many numbers are output?

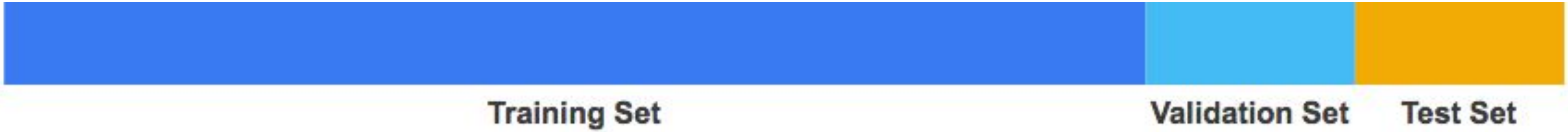
=1
unidimensional regression
(i.e. regression)
(e.g. how many minutes of video will this user watch?)

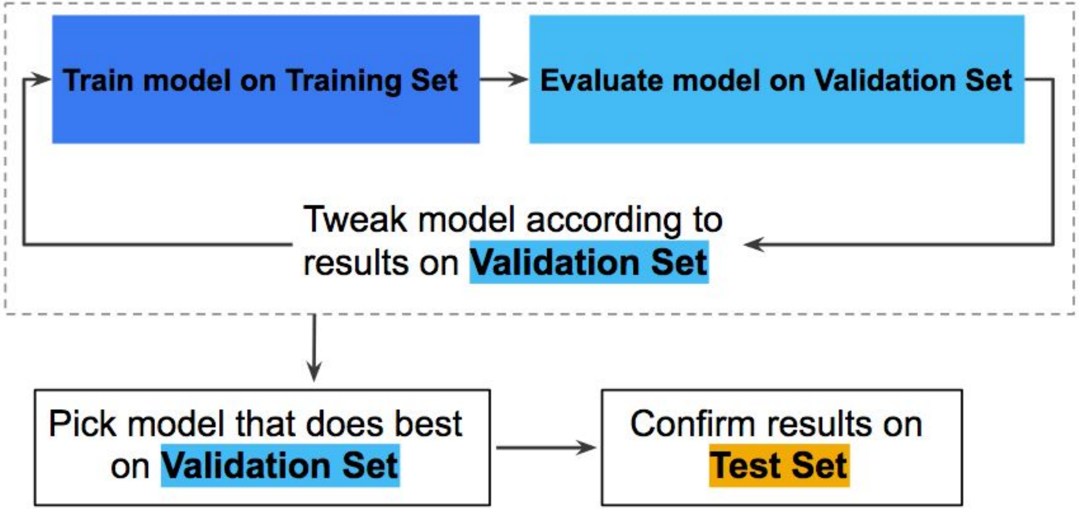
>1
multidimensional regression
(e.g. what is the [latitude, longitude] of the location in the photo?)

End to End ML Pipeline



Preparing your dataset:

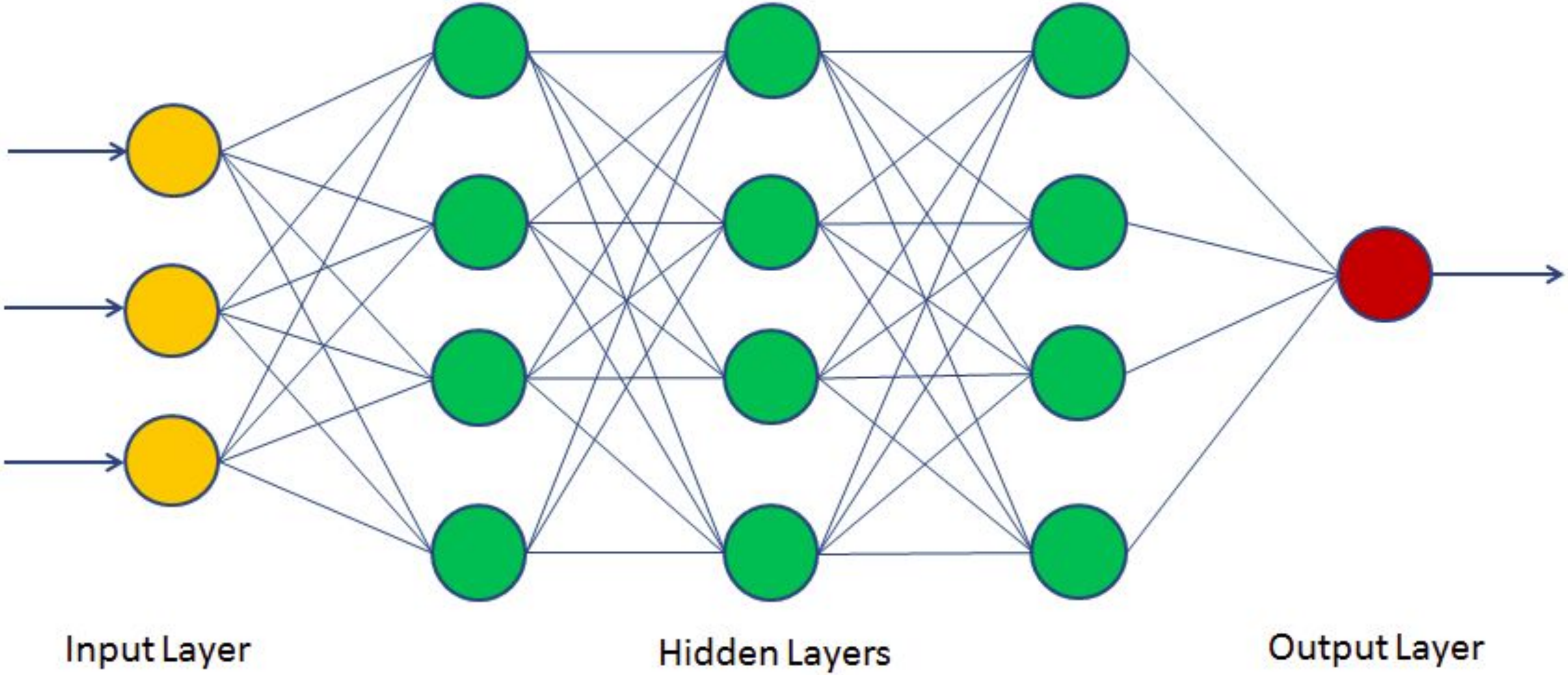



Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

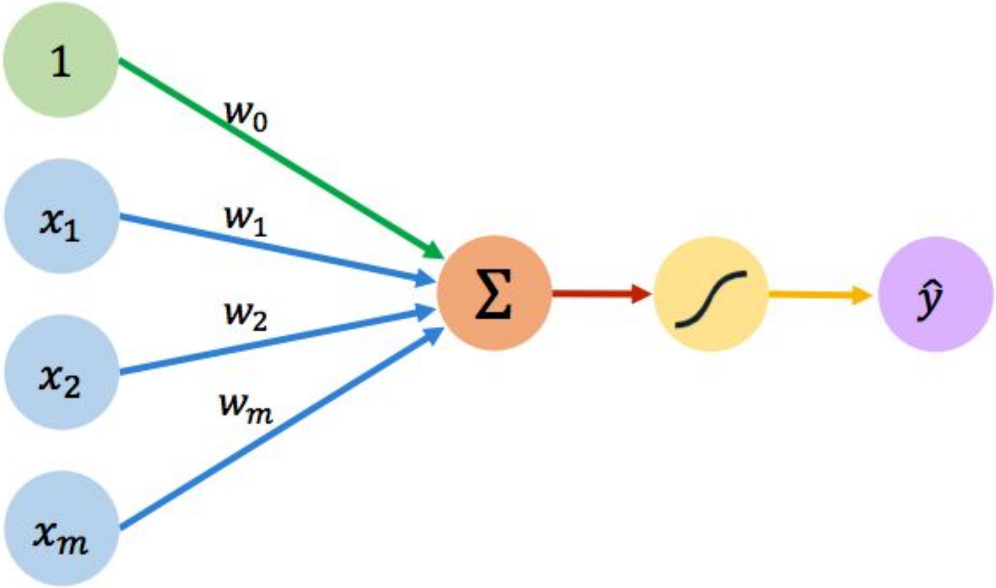
Overview of Neural Networks

02

Neural Network



The Perceptron



Output

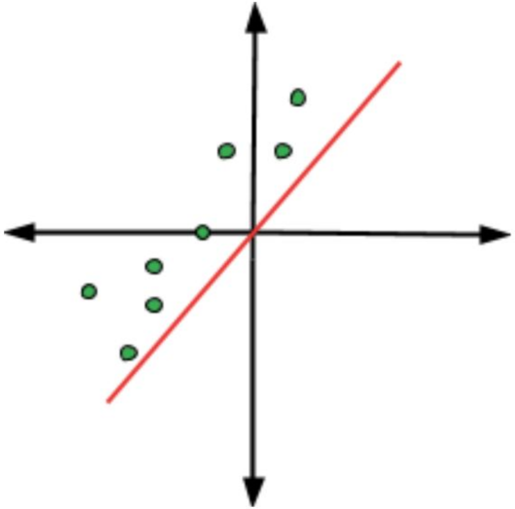
Linear combination of inputs

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

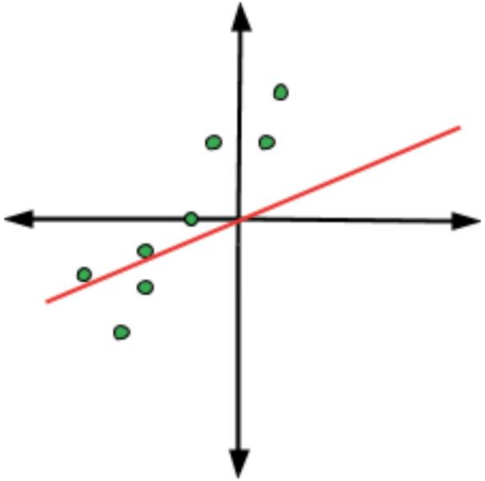
Non-linear activation function

Bias

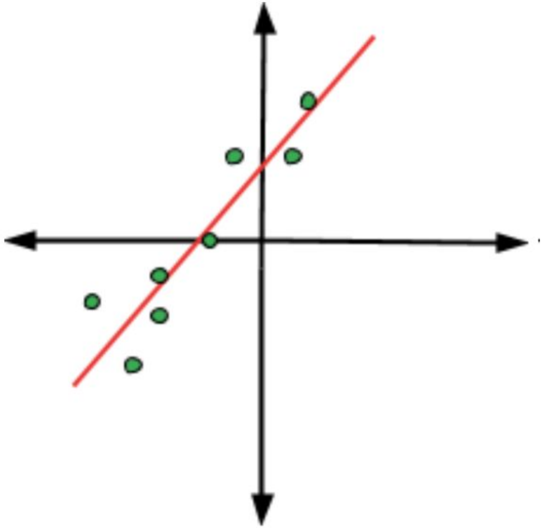
Bias



$y = w_1 x$



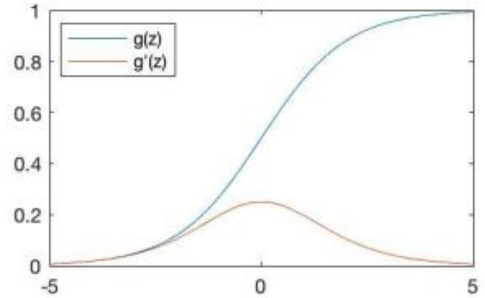
$y = w_2 x$



$y = w_0 + w_1 x$

Activation Functions: Introduce Non-Linearity

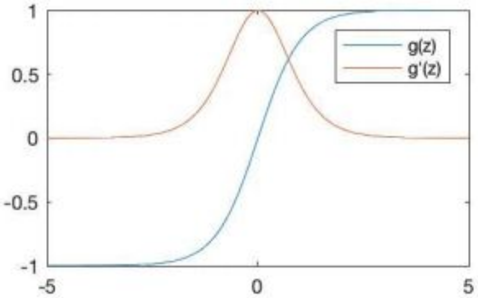
Sigmoid Function



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

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

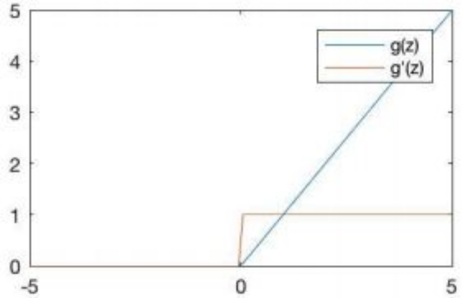
Hyperbolic Tangent



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

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

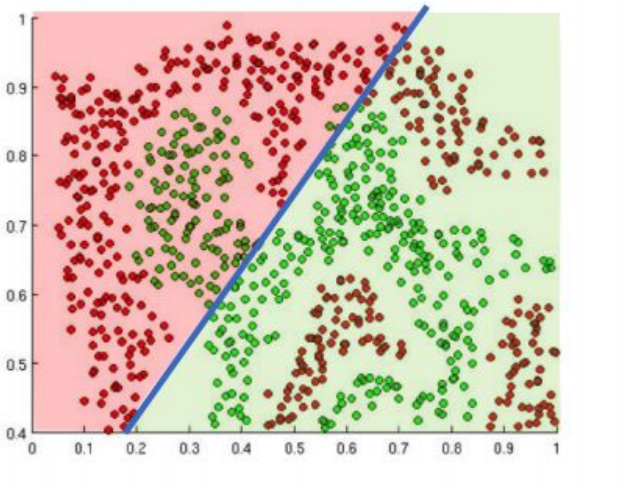
Rectified Linear Unit (ReLU)



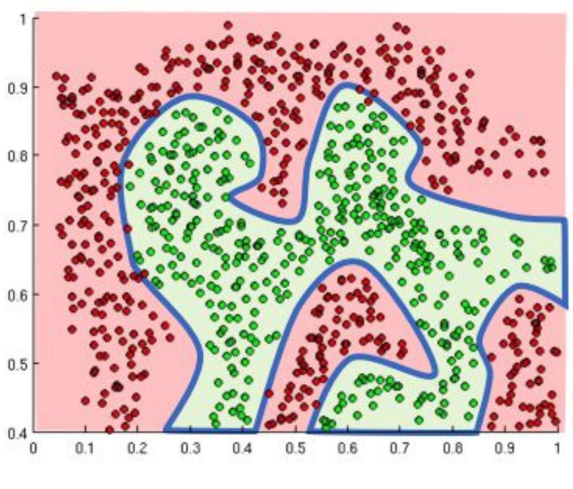
$$g(z) = \max(0, z)$$

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

Activation Functions: Introduce Non-Linearity



Linear Activation functions produce linear decisions no matter the network size

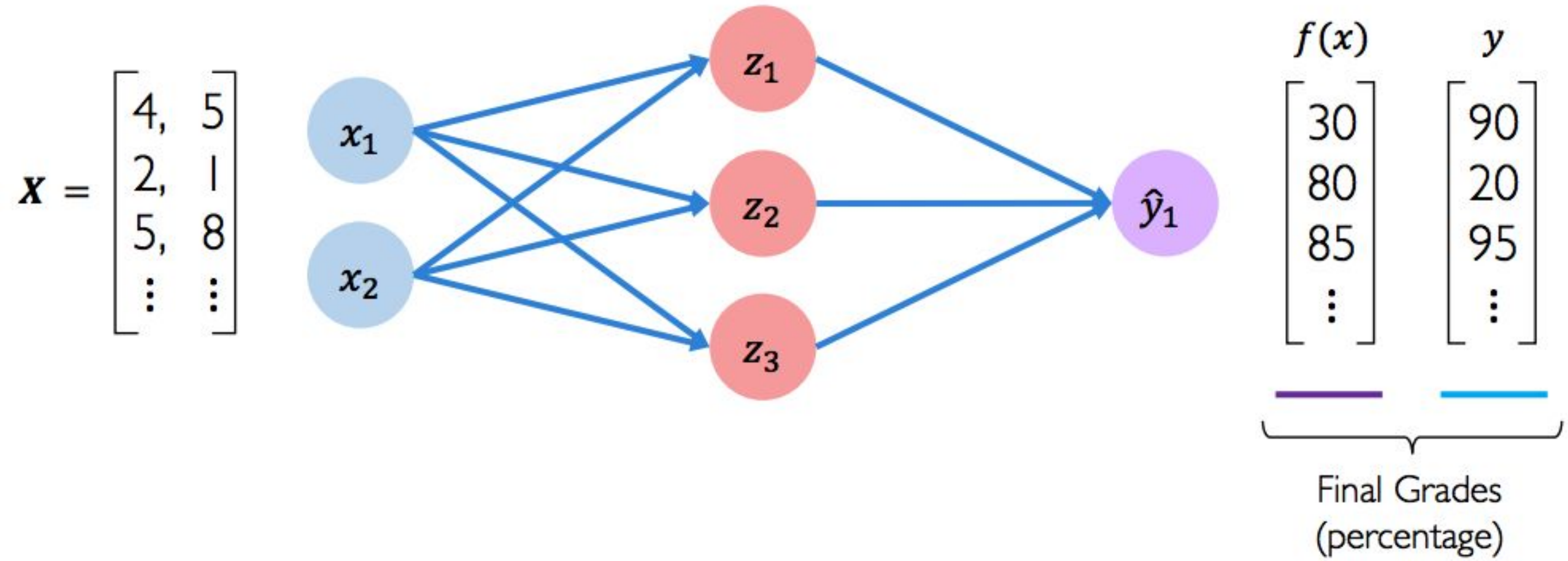


Non-linearities allow us to approximate arbitrarily complex functions

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.

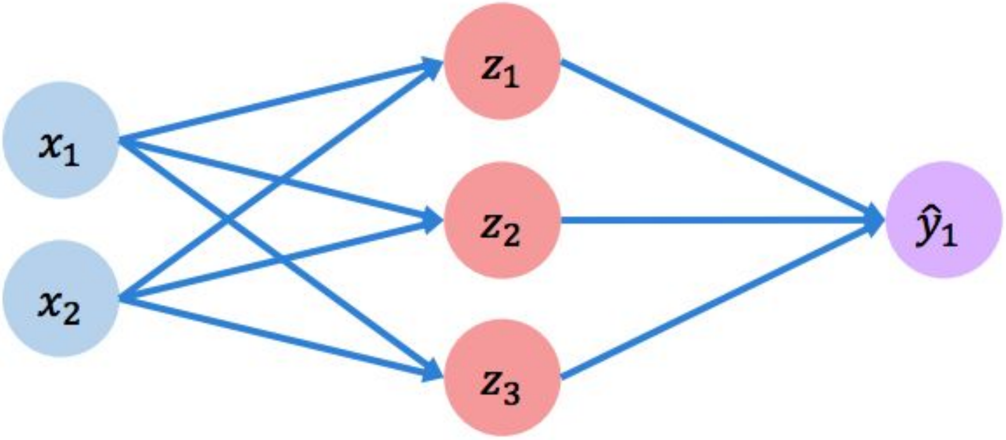
Predicted Actual



Mean Squared Loss

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

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



Predicted Actual Score

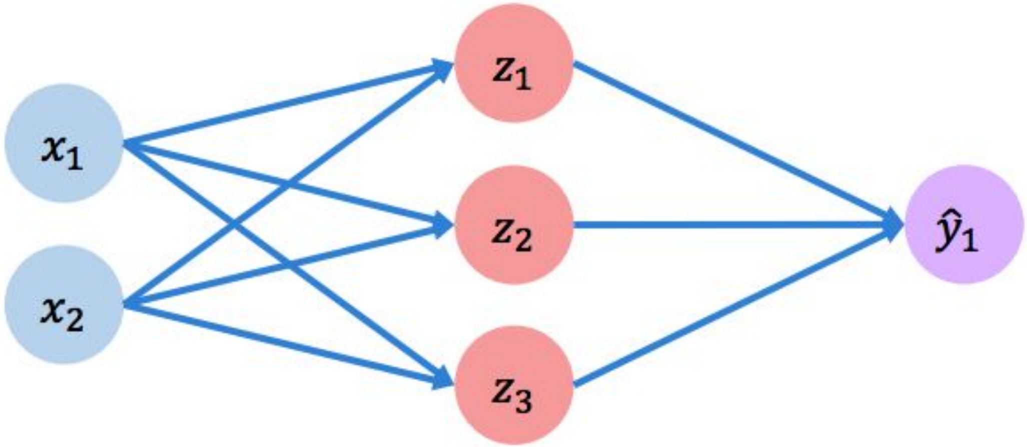
$$\underbrace{\begin{array}{cc} f(x) & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}}_{\text{Final Grades (percentage)}}$$

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

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.

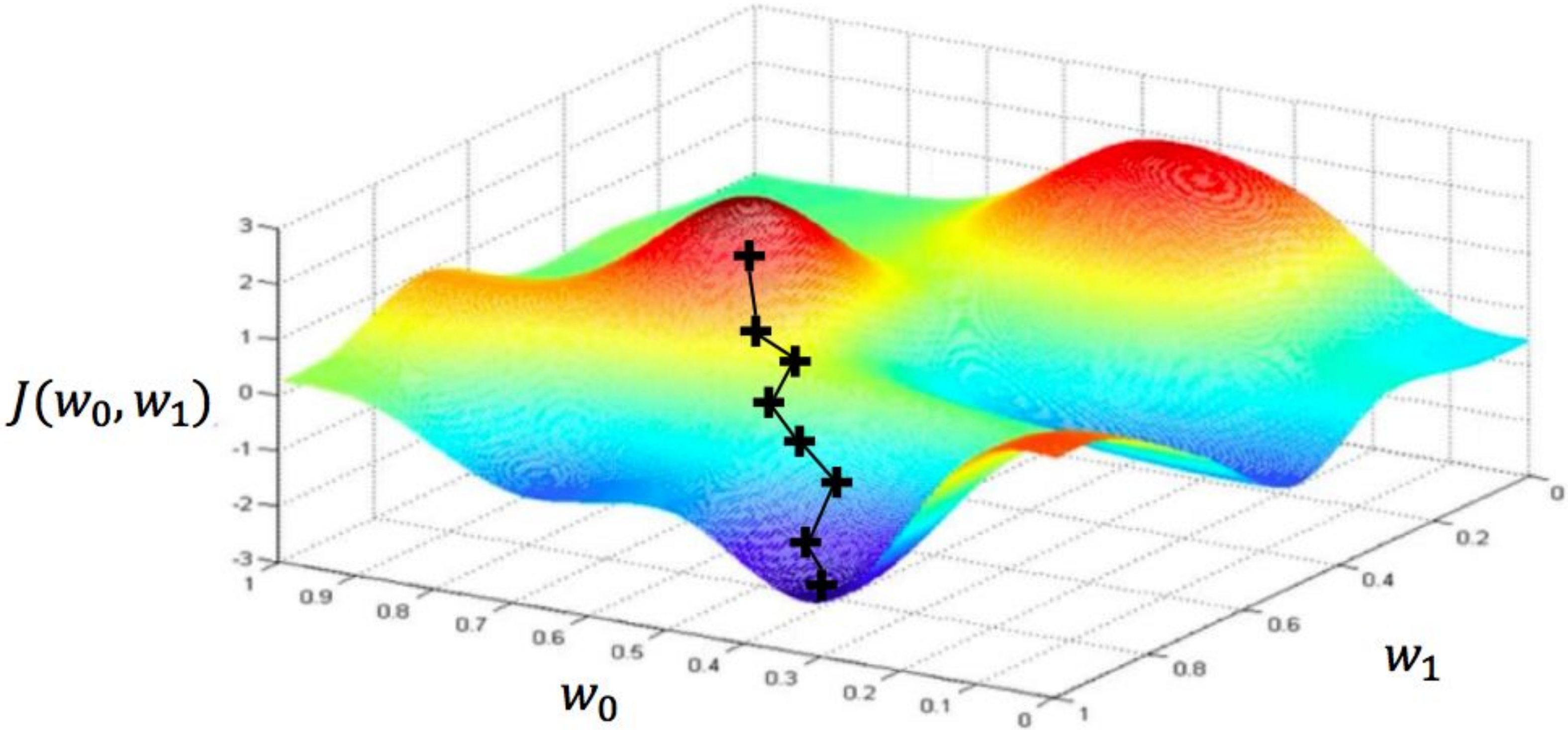
$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



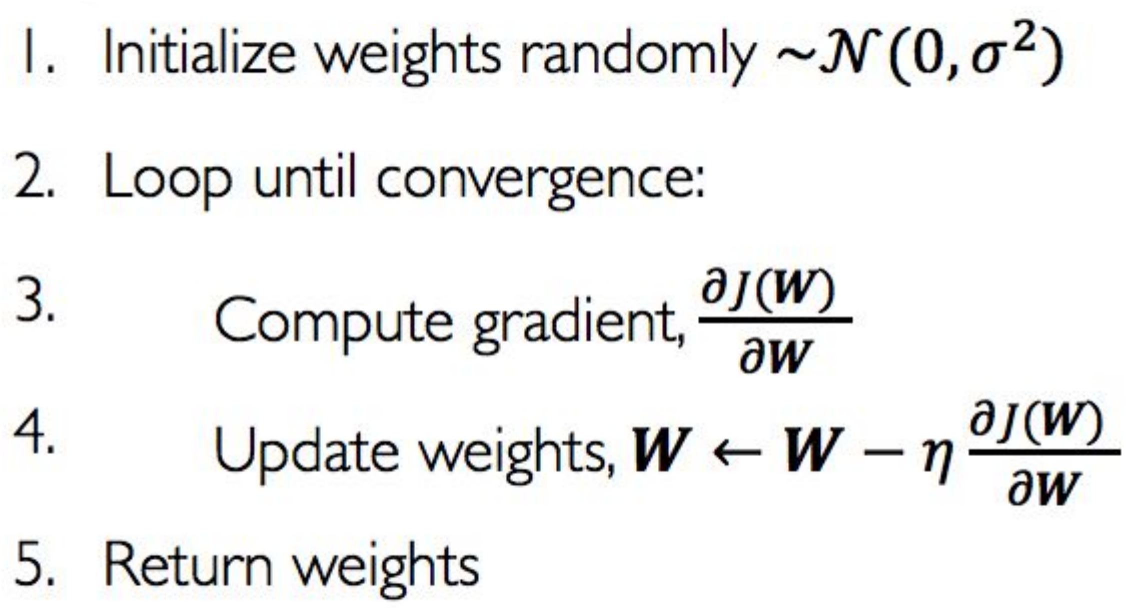
Predicted	Actual Score
$f(x)$	y
$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$

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

Visualizing our Loss Function

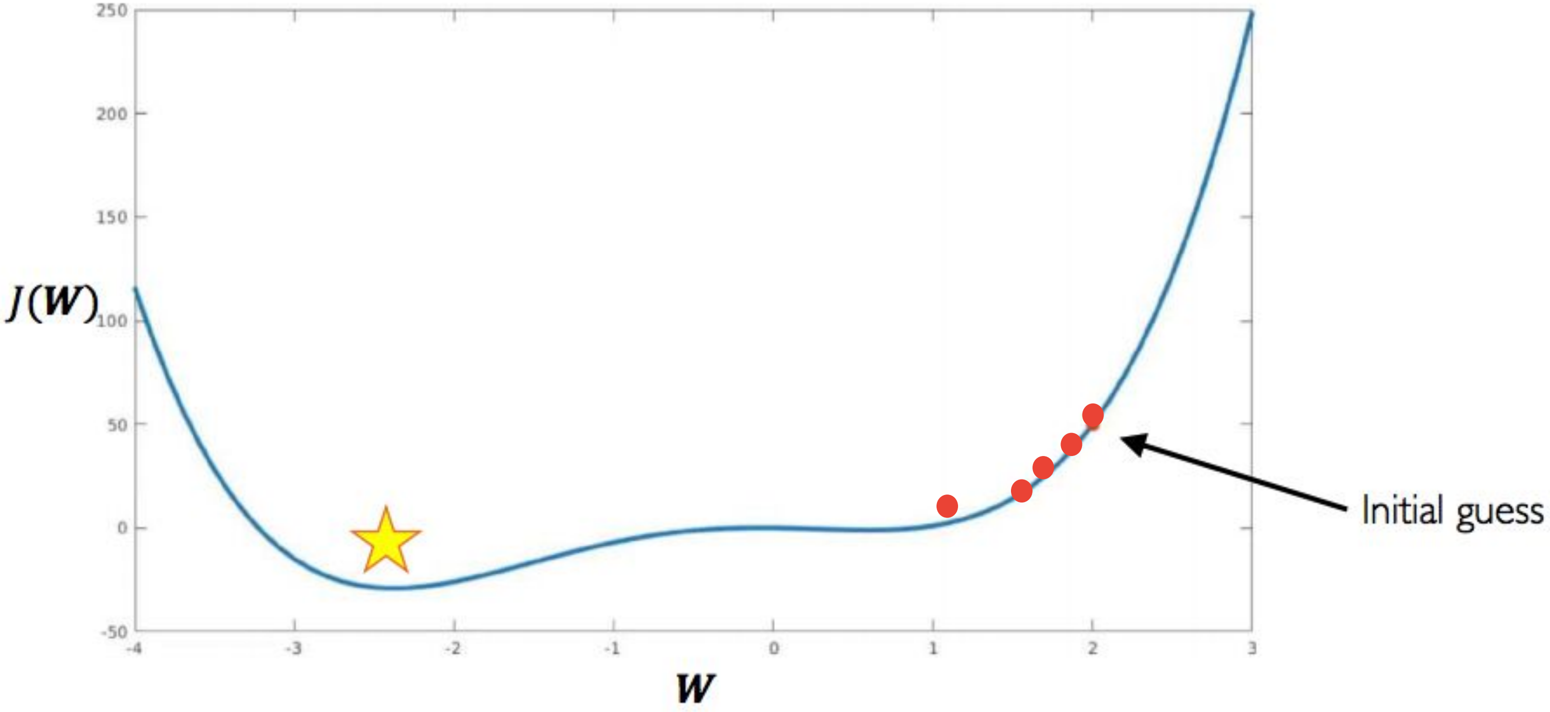


Gradient Descent Algorithm

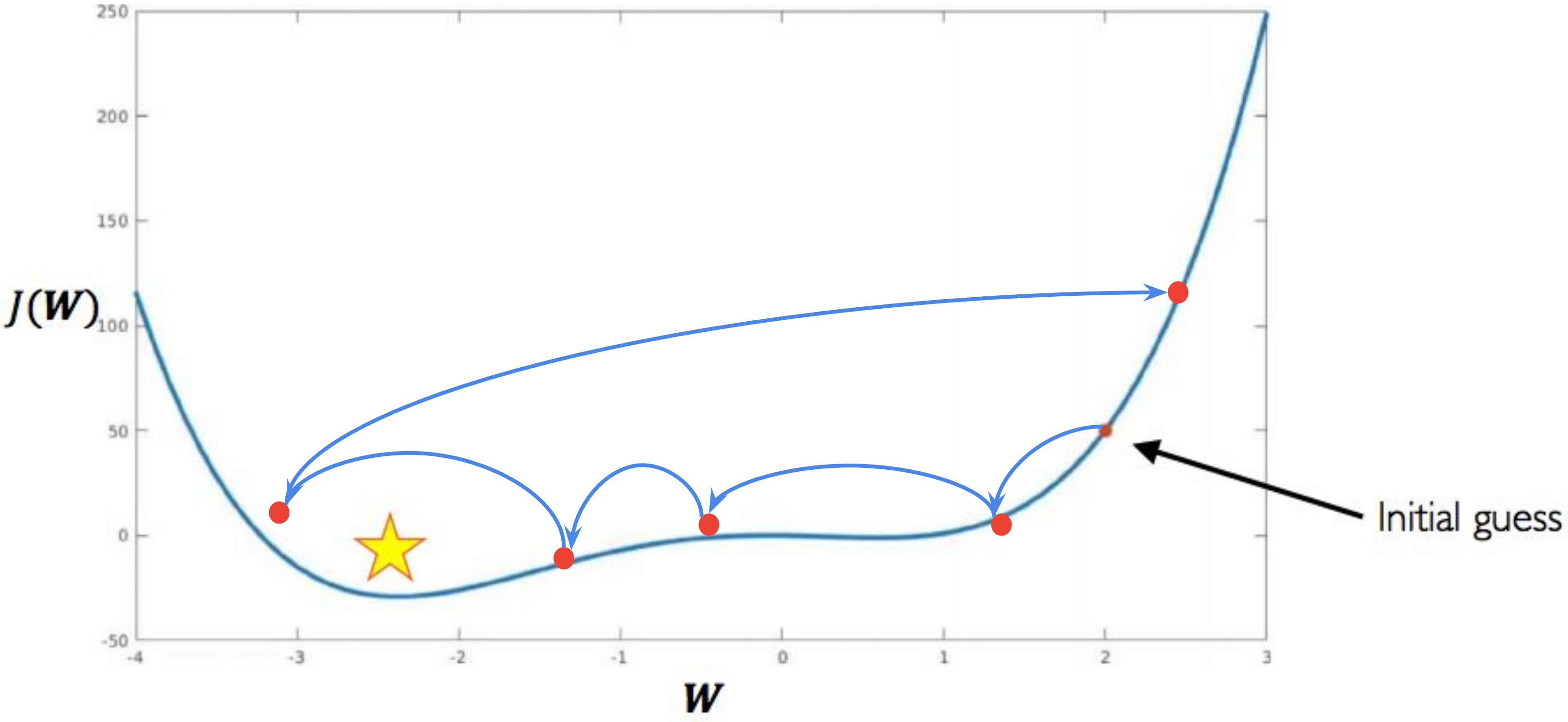


```
1. Initialize weights randomly $\sim\mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3.         Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4.         Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial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

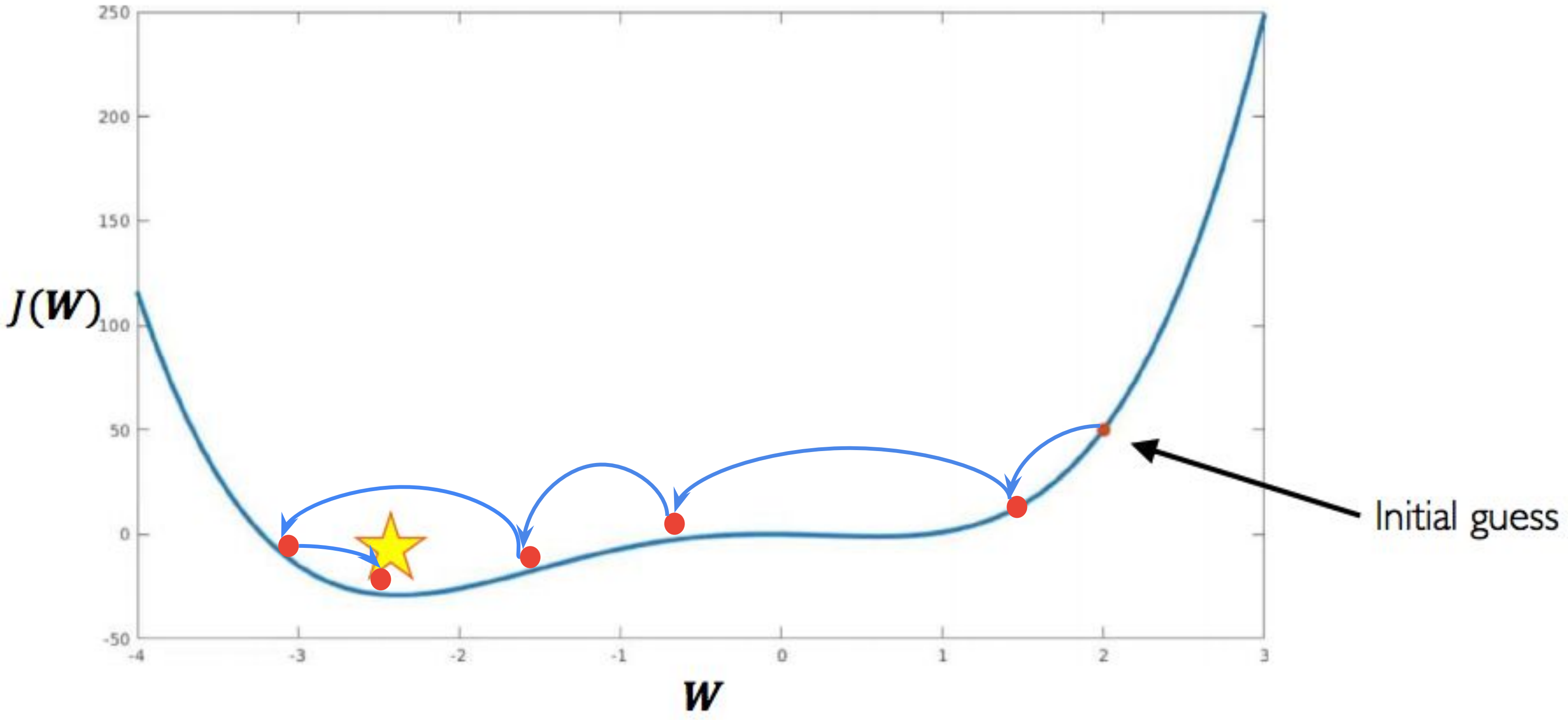
Low Learning Rate



High Learning Rate



Good Learning Rate



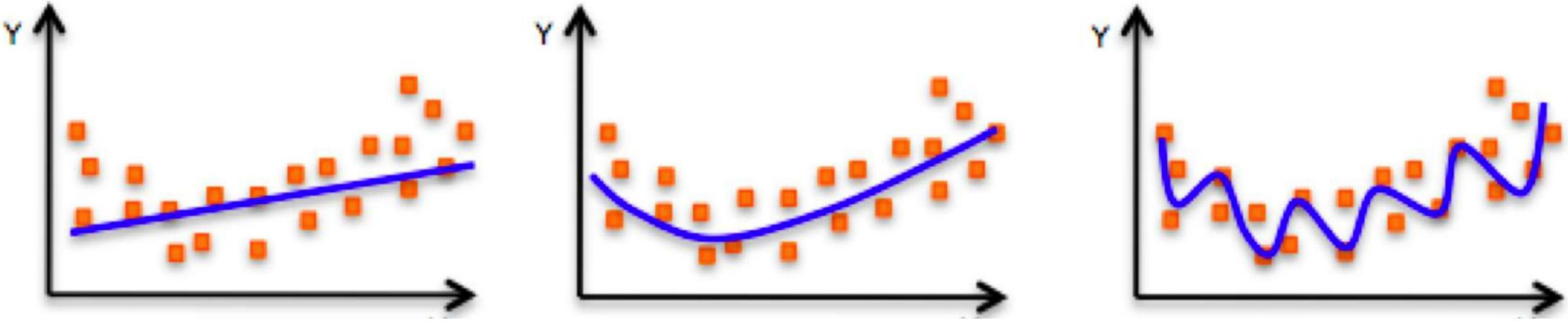
Adaptive Learning Rate Algorithms

- 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



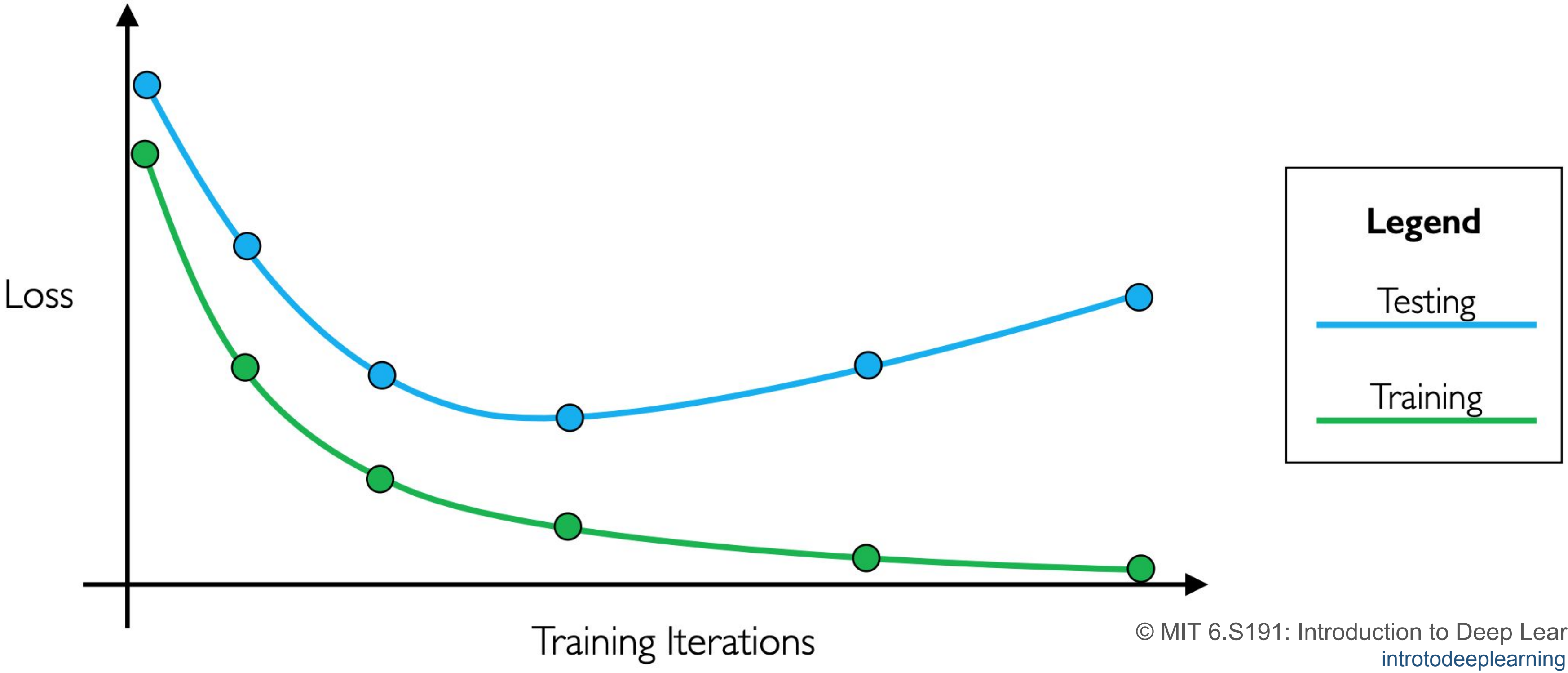
Underfitting
Model does not have capacity
to fully learn the data

← **Ideal fit** →

Overfitting
Too complex, extra parameters,
does not generalize well

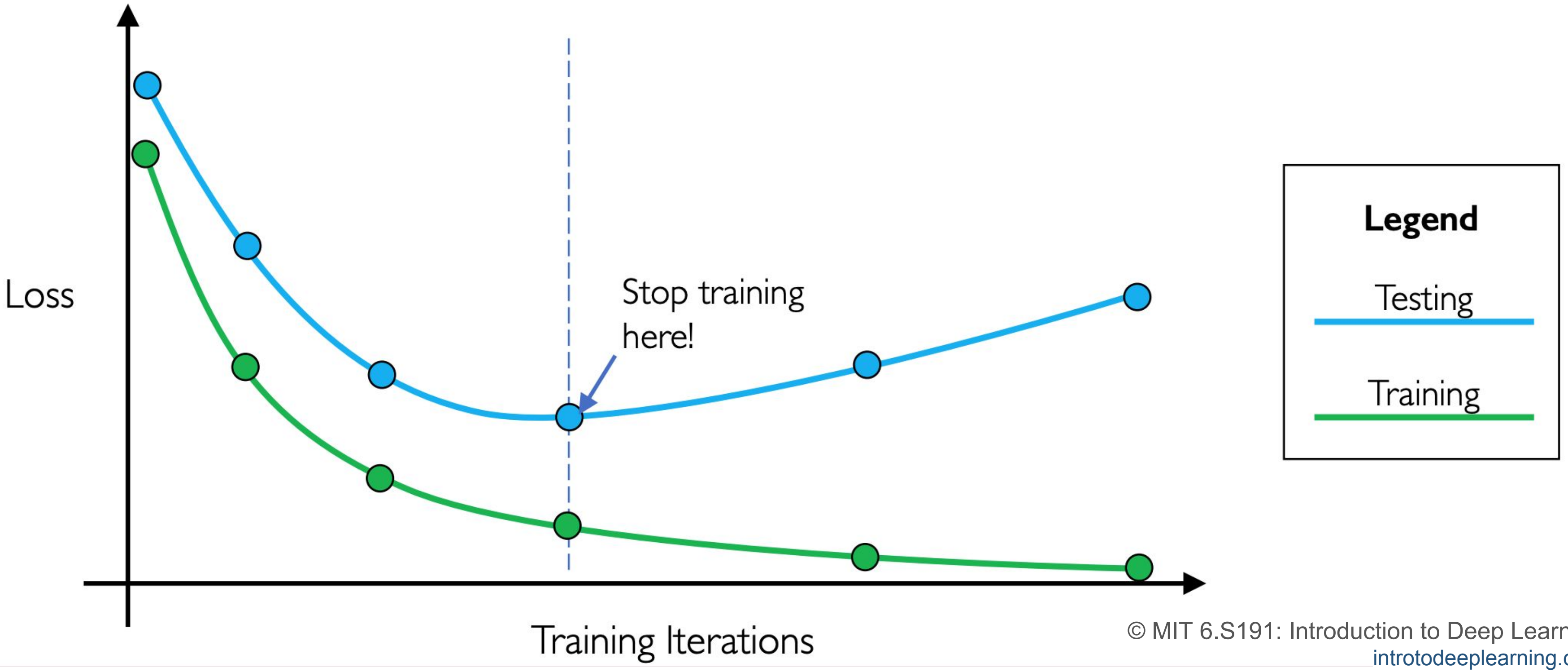
Early Stopping

Stop training before we start overfitting.



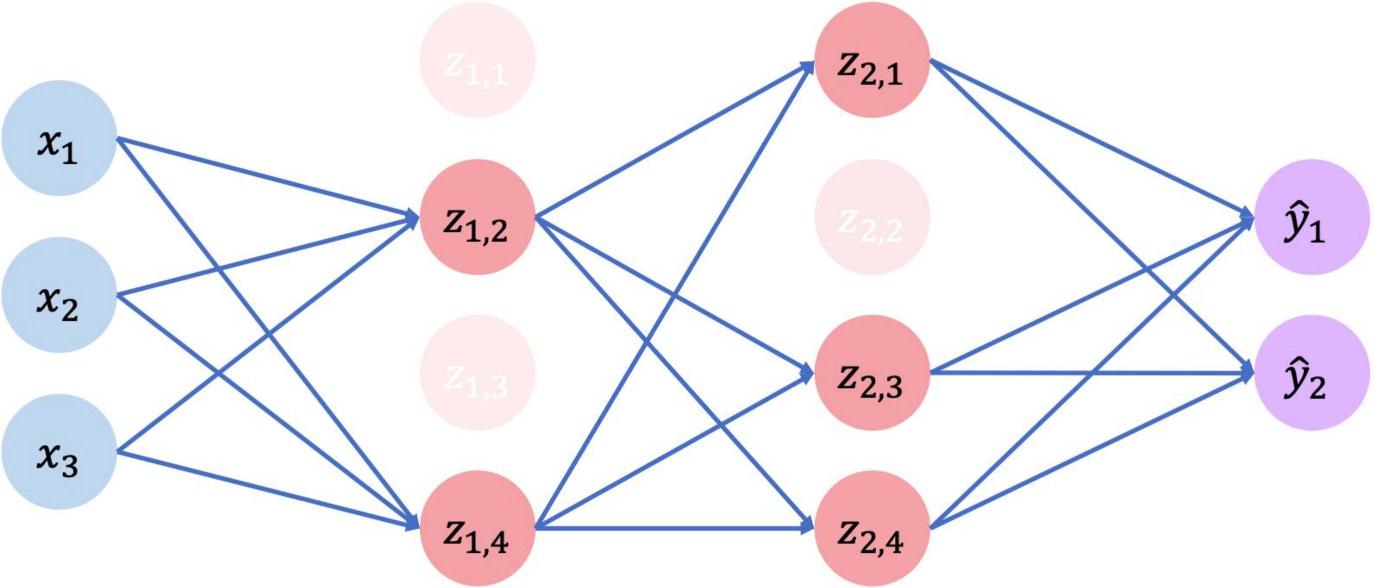
Early Stopping

Stop training before we start overfitting.



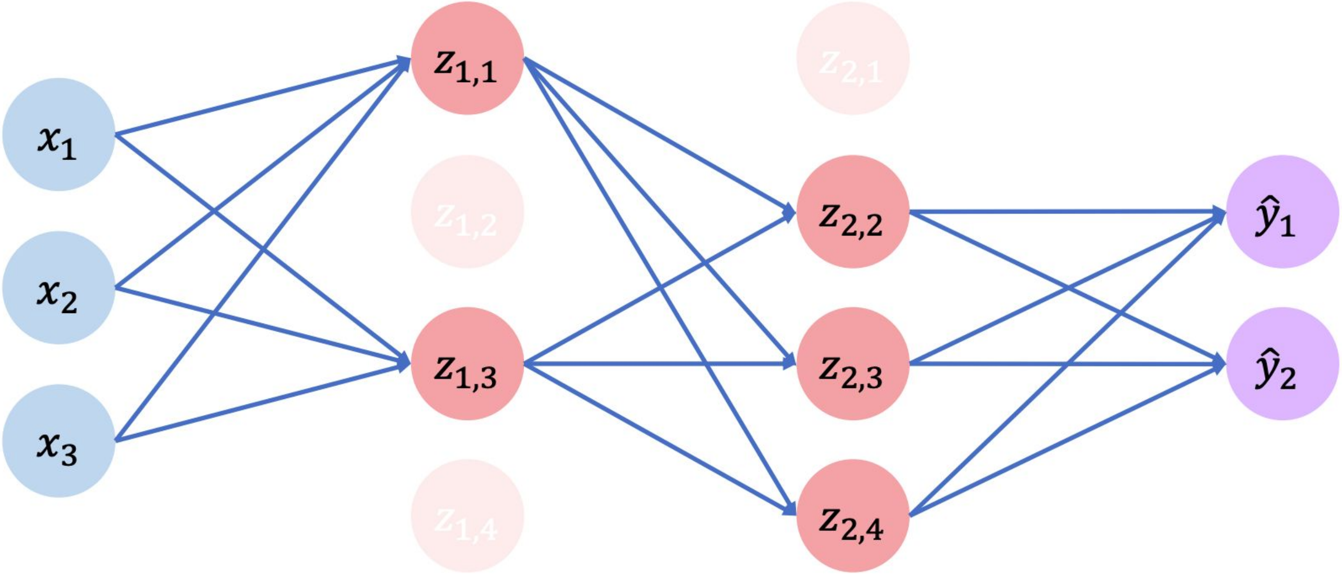
Dropouts

Randomly set some activations to 0



Dropouts

Randomly set some activations to 0



Codelab: Sign Language Recognition

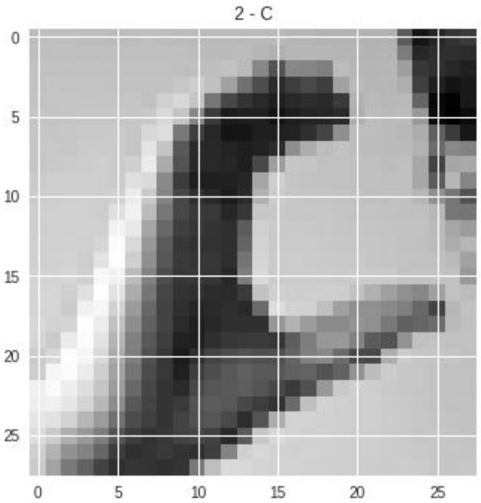
03

Demo

Audience Colab - https://colab.research.google.com/drive/1eOSEaVt1zBbMs_58v-vXOG94equtrqrD

Reference Colab - https://colab.research.google.com/drive/1IeN0COdZ_bz0GUh_yY1USjflfr3Hm-uF

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.

Quirks of Real World AI Development

04

Quirks of Real World AI Development

- From unquantifiable goals to measurable metrics
- Getting a good labelled dataset
- Serving an ML Model
- Developer productivity

From unquantifiable goals to measurable metrics

- Finding the right metric
- Measuring success on real data
- Metrics may be influenced by other changes

Getting a good labelled dataset

- Direct or Derived labels
- Finding a good sample to label
- Wrong labels in large datasets
- Bias in data

Serving an ML Model

- Kind of model and features to use

- Offline
- Online
- Nearline
- On device

Developer Productivity

- Versioning of models and datasets
- Searchable and reproducible experiments
- Monitoring performance, A/B testing, Debugging

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/