Multi-Layered Perceptron (Part 2)

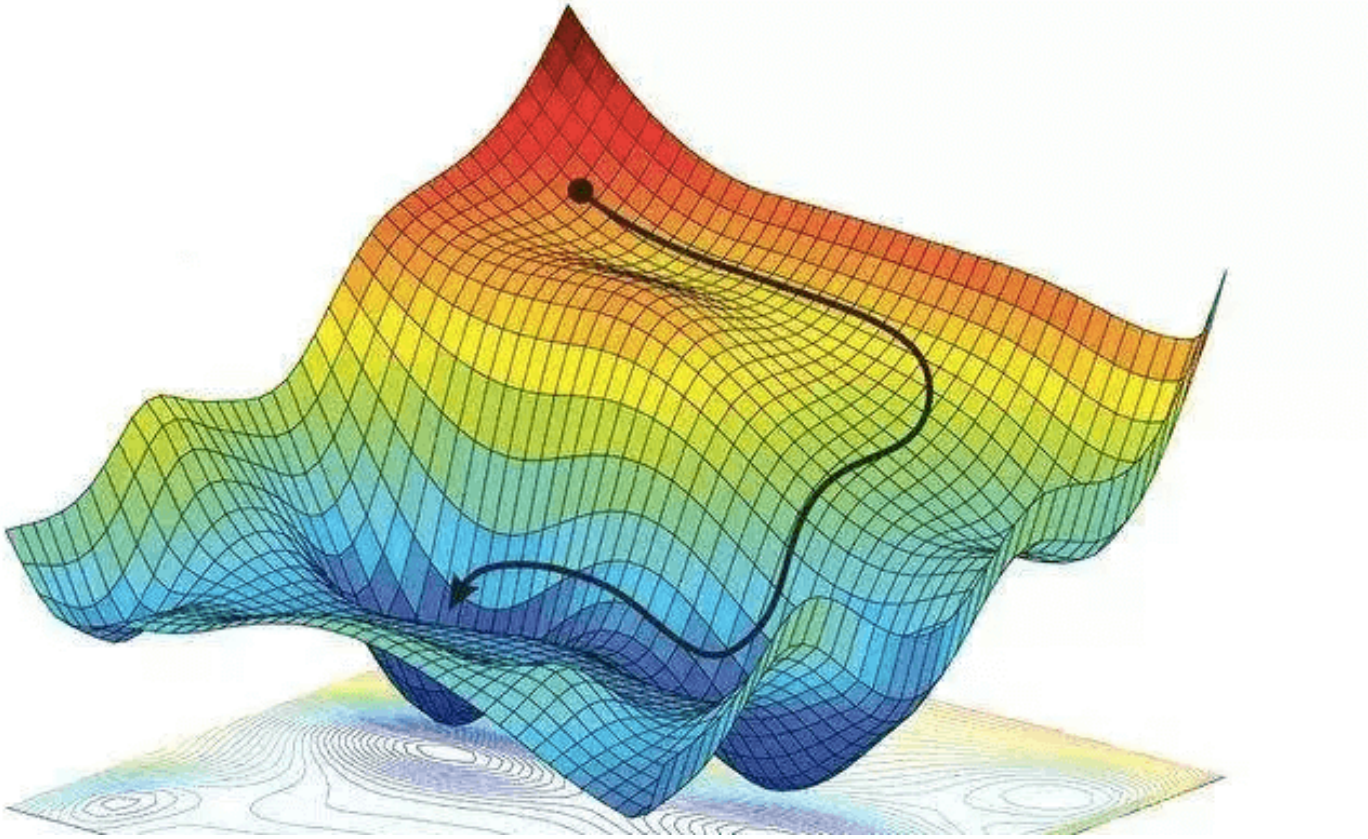
John Tan Chong Min

What to focus on

- Focus more on the concepts, less on the math
- No need to worry about the math equations
- Ask questions anytime: All questions are good. Questions help with understanding
- Every week: Theory first, then coding practical

Recap: Gradient Descent

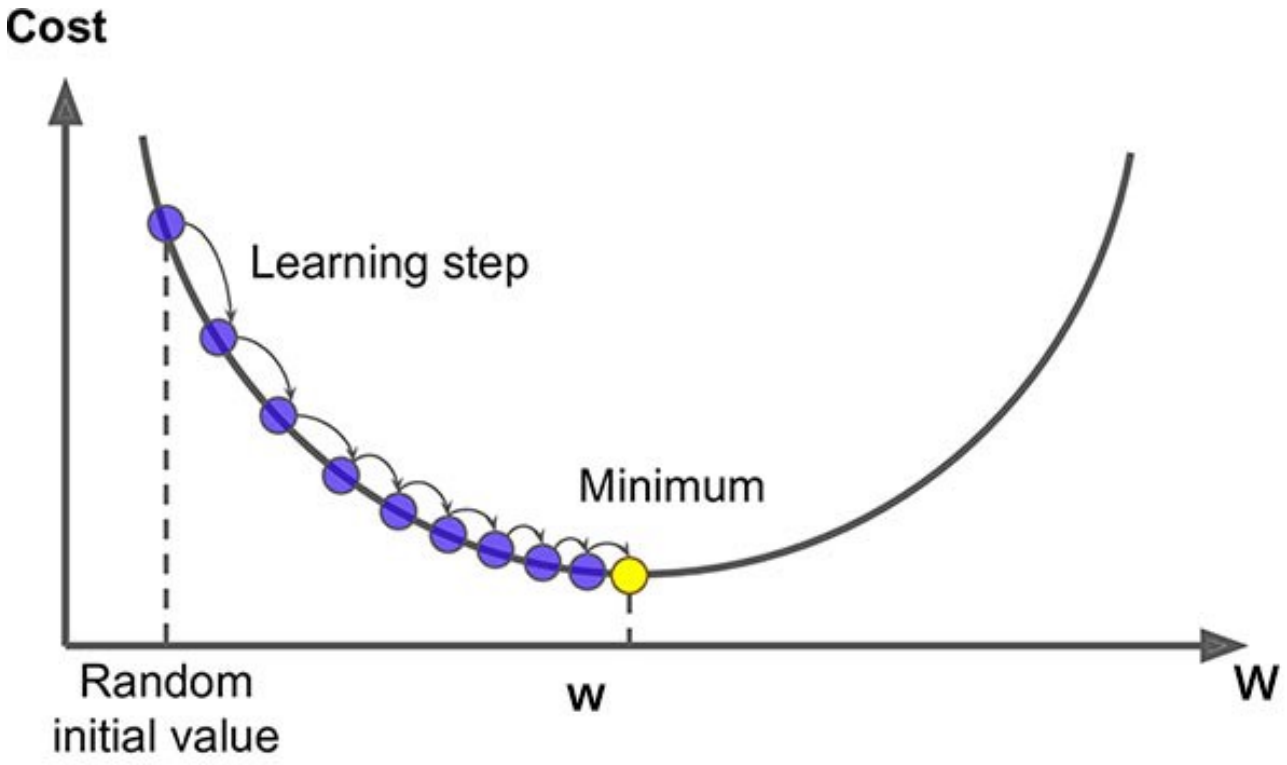
Loss Landscape



If we want to go down a hill, without knowing the full landscape but only the surrounding area.

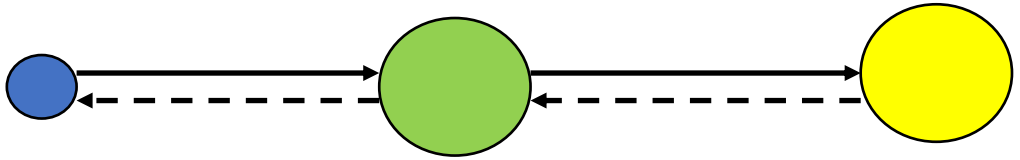
How do we go down?

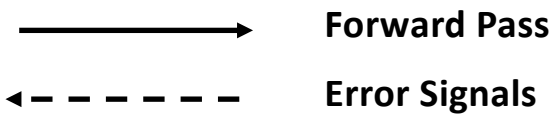
Gradient Descent



$$w = w - lr \cdot \frac{\partial L}{\partial w}$$

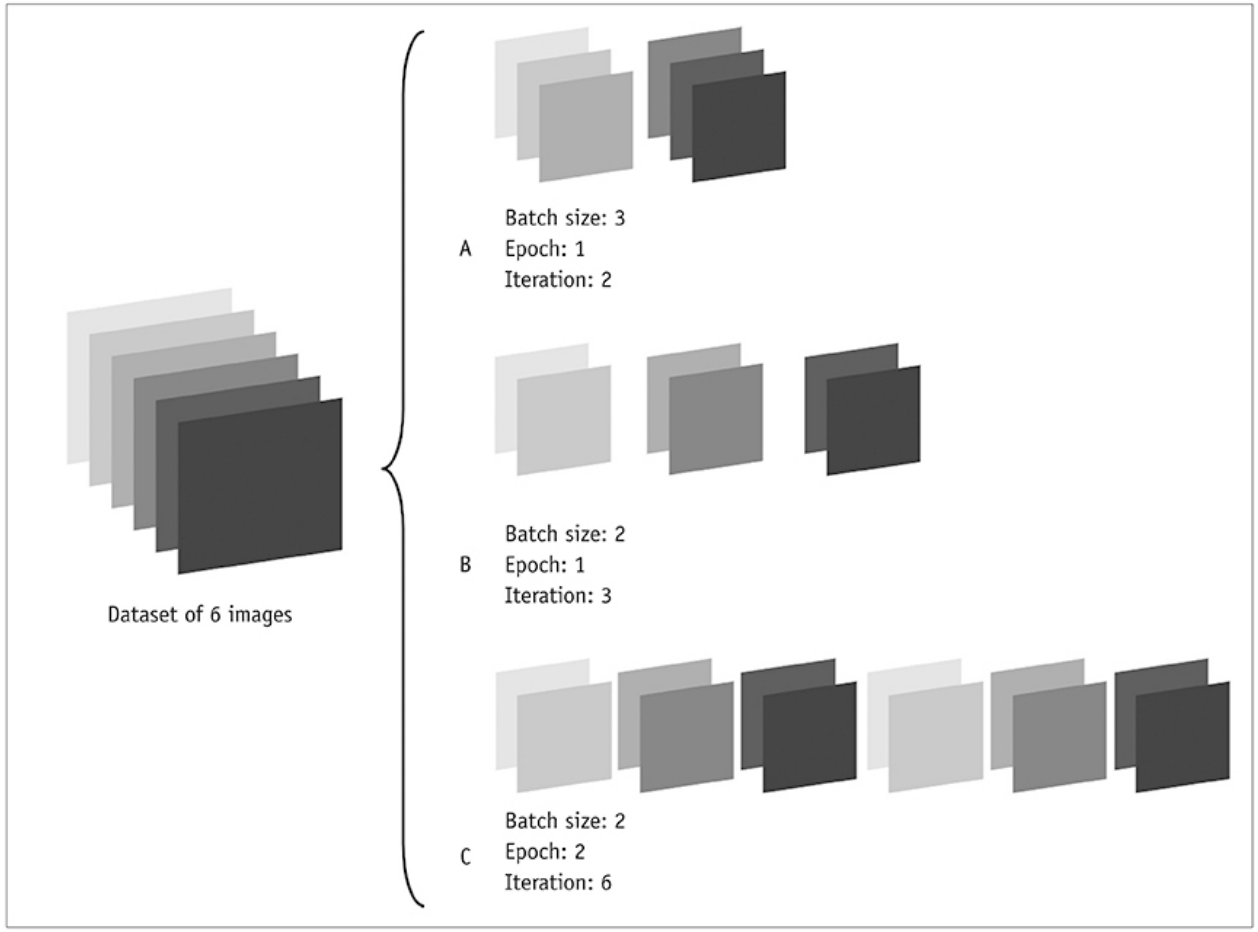
Backpropagation in Practice





Types of Gradient Descent

Batch, Epoch, Iteration

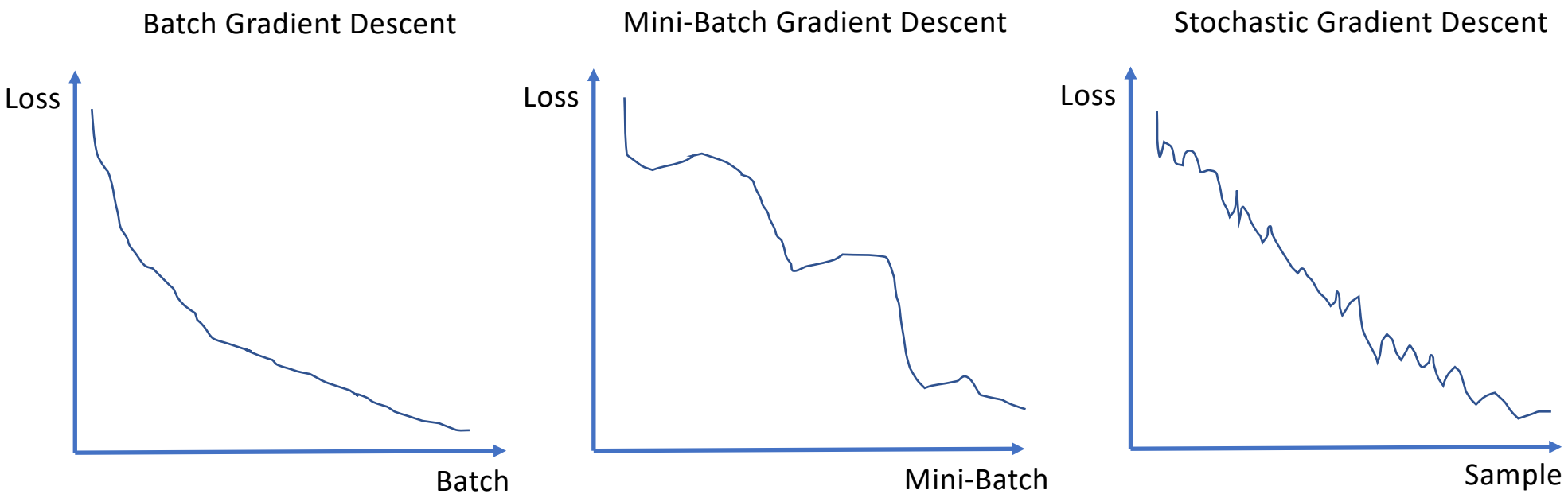


Batch size: Number of samples per pass through neural network

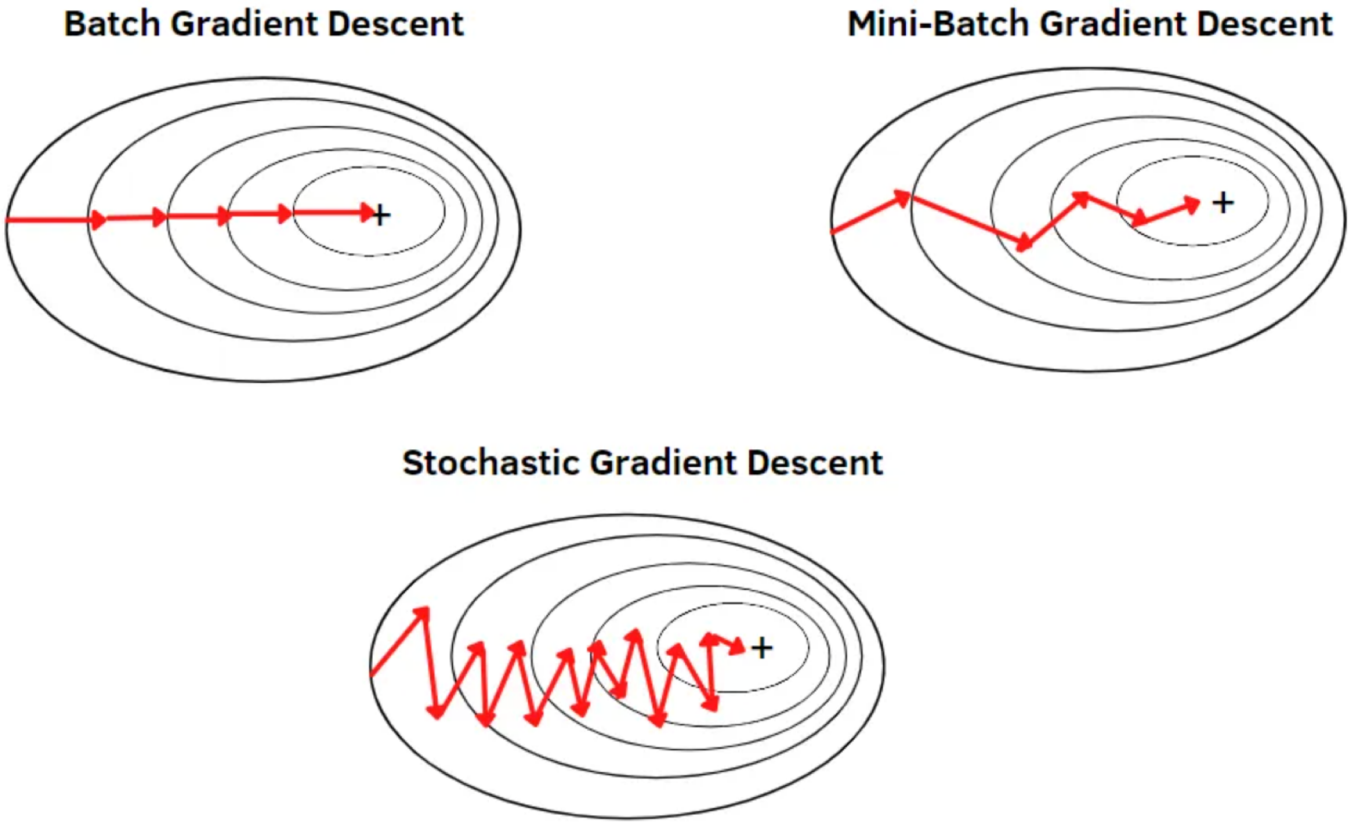
Epoch: Number of times we pass the entire batch of samples

Iteration: Number of passes through neural network

Batch vs Stochastic Gradient Descent



Batch vs Stochastic Gradient Descent



Optimizers

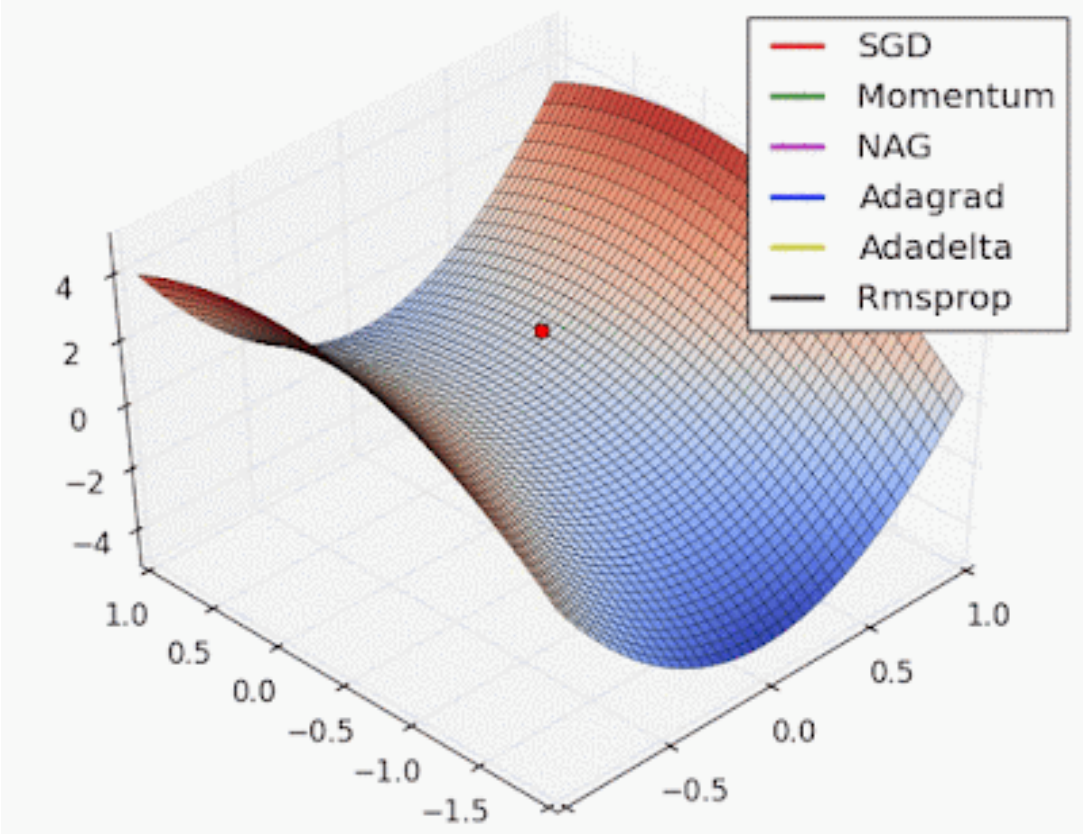
- Basic: Stochastic Gradient Descent (SGD) $$w = w - \mathrm{lr} \cdot \frac{\partial L}{\partial w}$$

- Accumulated gradients
  - **Momentum:** Keep track of previous gradients & move a bit there
    - Ball rolling down a hill
  - **Nesterov Accelerated Gradient (NAG):** Momentum + Forward-prediction for gradient
    - Ball rolling down the hill with future position prediction

- Adaptive Learning Rates
  - **Adagrad:** Scale learning rate for each parameter by root mean squared of accumulated gradients
  - **RMSProp:** Adagrad with decaying accumulated gradients
  - **Adadelta:** RMSProp with learning rate approximated by previous parameter updates
  - **Adam:** (roughly) RMSProp with Momentum

See Math at:
https://ruder.io/optimizing-gradient-descent/

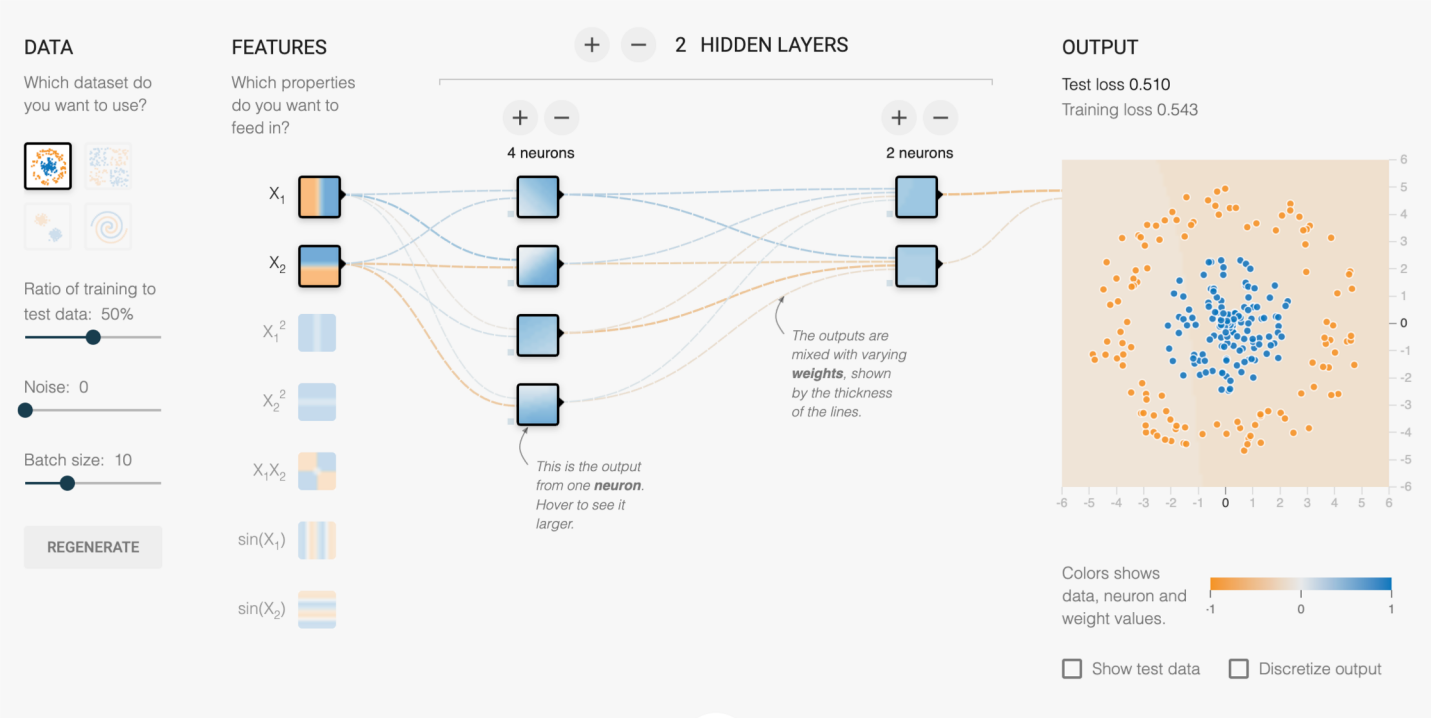
Optimizers



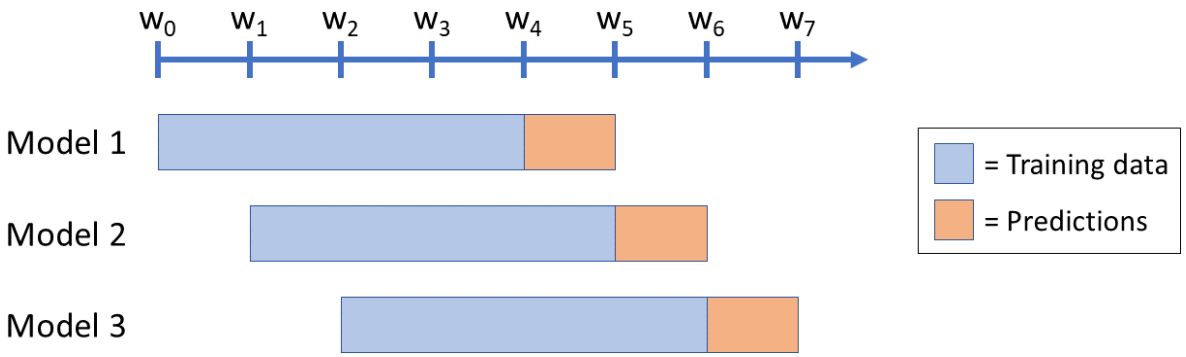
http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

TensorFlow PlayGround

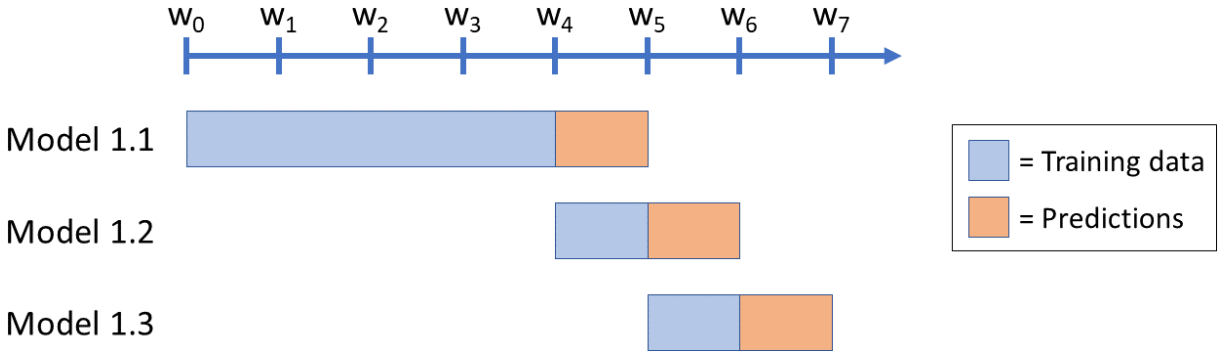
- https://playground.tensorflow.org/



Continual Learning



Static Learning:
Train once per window,
Restart training from scratch with new data



Continual Learning:
Use previous model state
Train model with new data every week

https://towardsdatascience.com/you-dont-need-neural-networks-to-do-continual-learning-2ed3bfe3dbfc

Softmax

- $Softmax(x_i) = \frac{e^{x_i\ /T}}{\sum_j e^{x_j\ /T}}$


```
import numpy as np
def softmax(x):
    return np.exp(x)/sum(np.exp(x))
```

Softmax

- $Softmax(x_i) = \frac{e^{x_i\ /T}}{\sum_j e^{x_j\ /T}}$


```
import numpy as np
def softmax(x):
    return np.exp(x)/sum(np.exp(x))
```

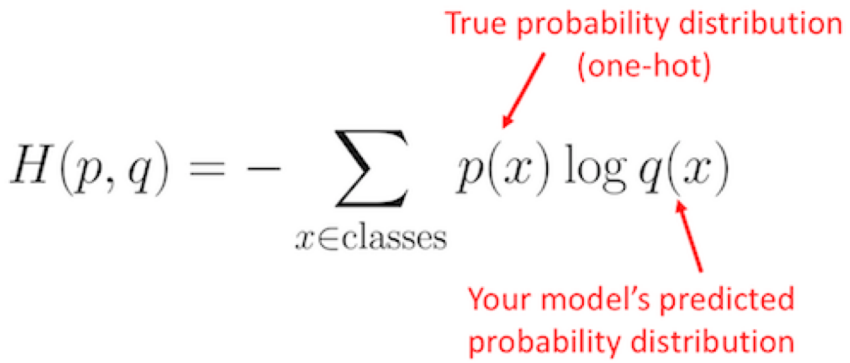
- Worked Example:

Class	Logits	Probabilities (T=0.5)	Probabilities (T=1)	Probabilities (T=2)
Apple	1.5	0.02	0.09	0.19
Banana	2.5	0.12	0.24	0.31
Pear	3.5	0.86	0.67	0.50

Smoother Probability Distribution

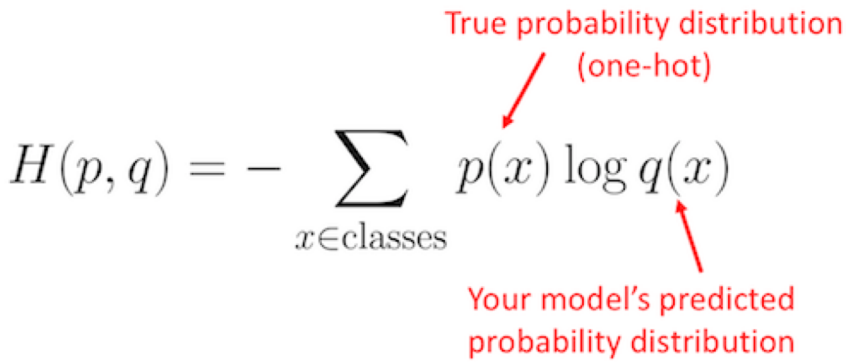
Cross-Entropy

- Cross-Entropy
  - $-\sum y \, log\hat{y}$
  - y: actual output probabilities
  - $\hat{y}$: predicted output probabilities



Cross-Entropy

- Cross-Entropy
  - $-\sum y \, log\hat{y}$
  - y: actual output probabilities
  - $\hat{y}$: predicted output probabilities



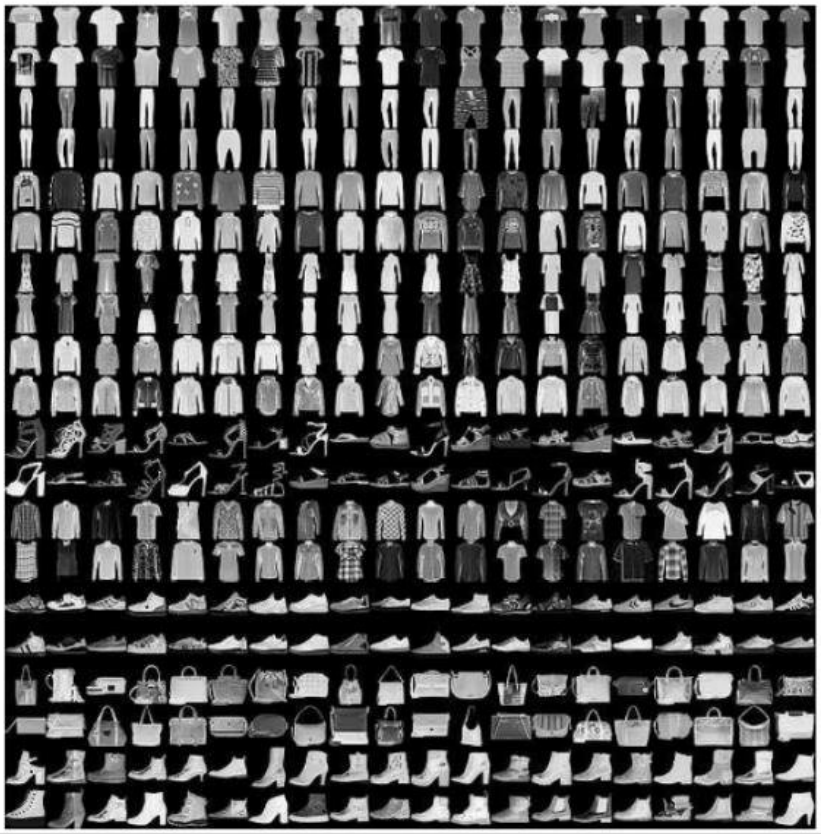
- Worked Example:
  - Actual probabilities: [0, 0, 1]
  - Not-so-perfect Prediction
    - Predicted probabilities: [0.2, 0.2, 0.6]
    - Cross-Entropy = -0*log(0.2) - 0*log(0.2) - 1*log(0.6) = -0 - 0 - (-0.74) = 0.74
  - Perfect Prediction
    - Predicted probabilities: [0, 0, 1]
    - Cross-Entropy = 0*log0 - 0*log0 - 1*log(1) = -0 - 0 - 0 = 0

Fashion MNIST Classification

With MLP in TensorFlow

Fashion MNIST

- 10 classes
- 60000 training images
- 10000 testing images
- Image Size: 28*28 pixels
  - (grayscale)

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Basic TensorFlow Loop

- Load Dataset
- Define Model
- model.compile
- model.fit
- model.predict / model.evaluate

TensorFlow – Load Dataset

The Fashion MNIST data is available directly in the tf.keras datasets API. You load it like this:

```
mnist = tf.keras.datasets.fashion_mnist
```

Calling load_data on this object will give you two sets of two lists, these will be the training and testing values for the graphics that contain the clothing items and their labels.

```
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
```

```
training_images.shape, training_labels.shape, test_images.shape, test_labels.shape
```

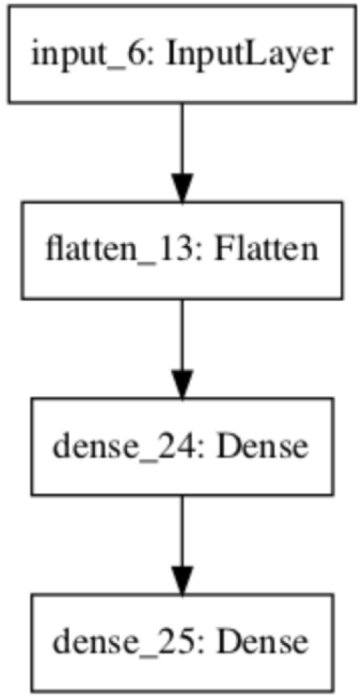
```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

TensorFlow – Define Model (Sequential)

- TensorFlow has many layers, like Lego blocks to build the model
- Sequential: Stack up Lego blocks in a line
  - Pipeline-style

```
model = tf.keras.models.Sequential(
    [tf.keras.layers.Input(shape = (28,28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)]
)
```

```
tf.keras.utils.plot_model(model)
```



TensorFlow – Define Model (Functional)

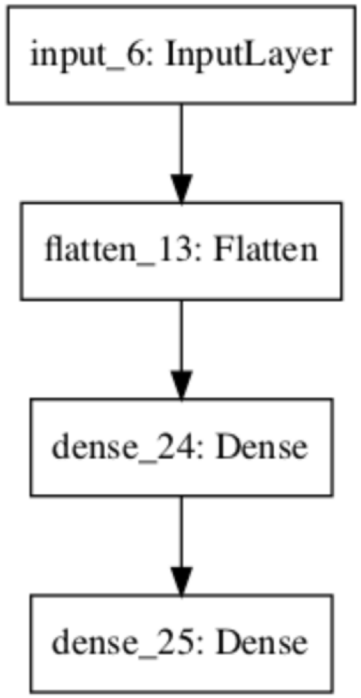
- TensorFlow has many layers, like Lego blocks to build the model
- Functional: Use Functions to write flow
  - More flexibility

```
inputs = tf.keras.layers.Input(shape = (28,28))

x = tf.keras.layers.Flatten()(inputs)
x = tf.keras.layers.Dense(128, activation = tf.nn.relu)(x)
outputs = tf.keras.layers.Dense(10, activation = tf.nn.softmax)(x)

model = tf.keras.Model(inputs = inputs, outputs = outputs)
```

```
tf.keras.utils.plot_model(model)
```



TensorFlow – Define Model (Parameters)

- 32 in the first parameter of the output shape is the batch size
- First Dense Layer has 784*128 weights + 128 biases = 100480 params
- Second Dense Layer has 128*10 weights + 10 biases = 1290 params

```
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (32, 784)                 0
_________________________________________________________________
dense_7 (Dense)              (32, 128)                 100480
_________________________________________________________________
dense_8 (Dense)              (32, 10)                  1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
```

TensorFlow – model.compile

- Specify the optimizer, loss function, any additional metrics

```
model.compile(optimizer = tf.optimizers.Adam(),
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

TensorFlow – model.fit

- Specify the training inputs, labels, validation split (if you want validation set), epochs

```
model.fit(training_images, training_labels, validation_split = 0.1, epochs=5)
```

```
Epoch 1/5
1688/1688 [==============================] - 1s 773us/step - loss: 0.5046 - accuracy: 0.8241 - val_loss: 0.4037 - val_accuracy: 0.8557
Epoch 2/5
1688/1688 [==============================] - 1s 731us/step - loss: 0.3774 - accuracy: 0.8635 - val_loss: 0.3768 - val_accuracy: 0.8668
Epoch 3/5
1688/1688 [==============================] - 1s 730us/step - loss: 0.3408 - accuracy: 0.8766 - val_loss: 0.3680 - val_accuracy: 0.8668
Epoch 4/5
1688/1688 [==============================] - 1s 734us/step - loss: 0.3143 - accuracy: 0.8848 - val_loss: 0.3511 - val_accuracy: 0.8757
Epoch 5/5
1688/1688 [==============================] - 1s 734us/step - loss: 0.2955 - accuracy: 0.8919 - val_loss: 0.3267 - val_accuracy: 0.8825
```

TensorFlow – model.predict / model.evaluate

- model.predict gives output on a batch of inputs (here is the post-softmax function)
- model.evaluate gives the loss and additional metrics on the batch of inputs

```
classifications = model.predict(test_images)

print(classifications[0])
```

```
[0.07315972 0.04128658 0.1259507  0.06062017 0.09420821 0.04218186 0.08253103 0.17838682 0.1160433  0.1856316 ]
```

```
model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 0s 545us/step - loss: 0.3436 - accuracy: 0.8751
[0.3436215817928314, 0.8751000165939331]
```

TensorFlow - Overall

```
import tensorflow as tf
print(tf.__version__)

mnist = tf.keras.datasets.fashion_mnist

(training_images, training_labels) ,  (test_images, test_labels) = mnist.load_data()

training_images = training_images/255.0
test_images = test_images/255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])

model.fit(training_images, training_labels, epochs=5)

model.evaluate(test_images, test_labels)
```

TensorFlow Practice

- Let us run the model!

Questions?