Hands on Generative AI

Text Classification

Data Trainers

Text Classification

Feed forward

Networks

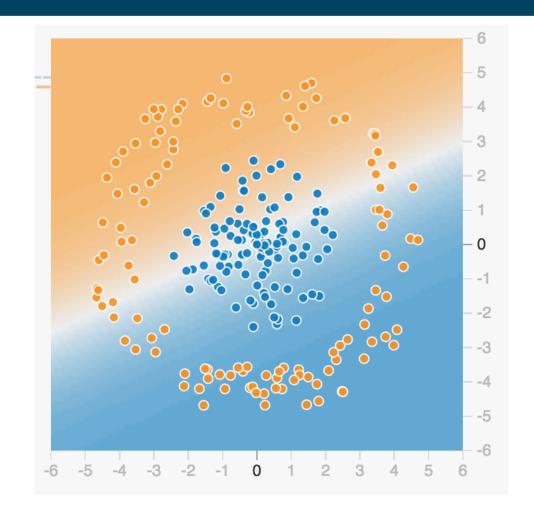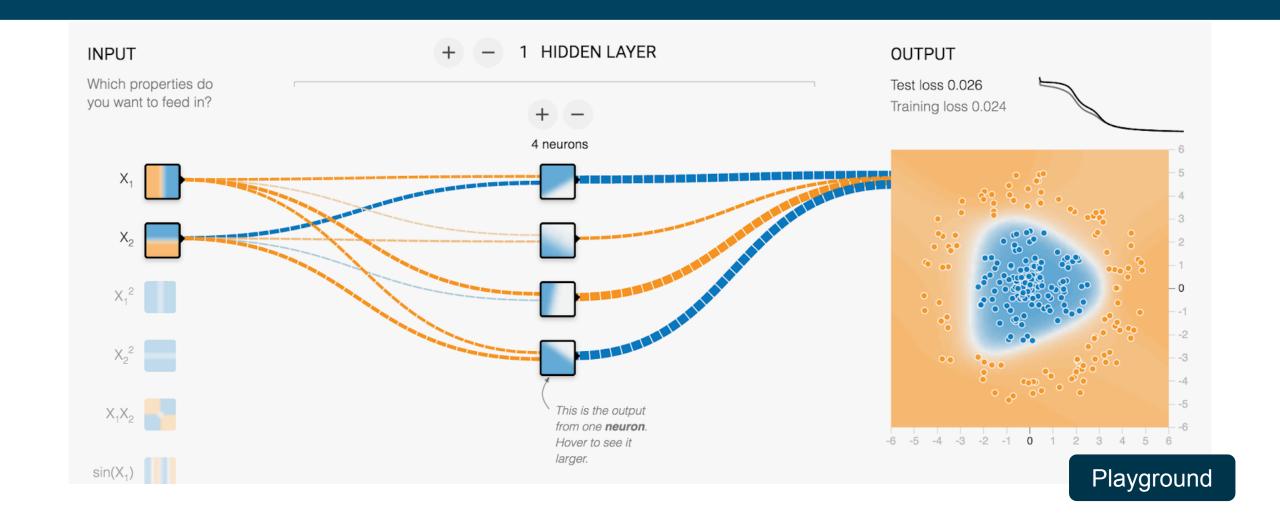▶ Can we separate this points to

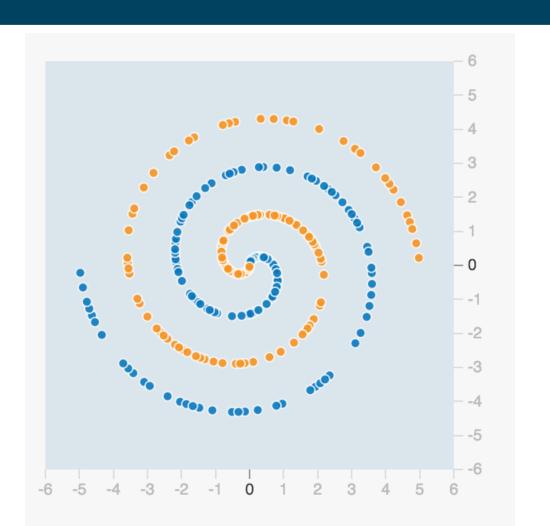Classify correctly the class

(seniment, category, etc… ?)

Playground

More neurons == nonlinearities

INPUT

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$sin(X_1)$

+ − 1 HIDDEN LAYER

+ −

4 neurons

This is the output from one **neuron**. Hover to see it larger.

OUTPUT

Test loss 0.026
Training loss 0.024

Playground

How can we do to split these?

In code

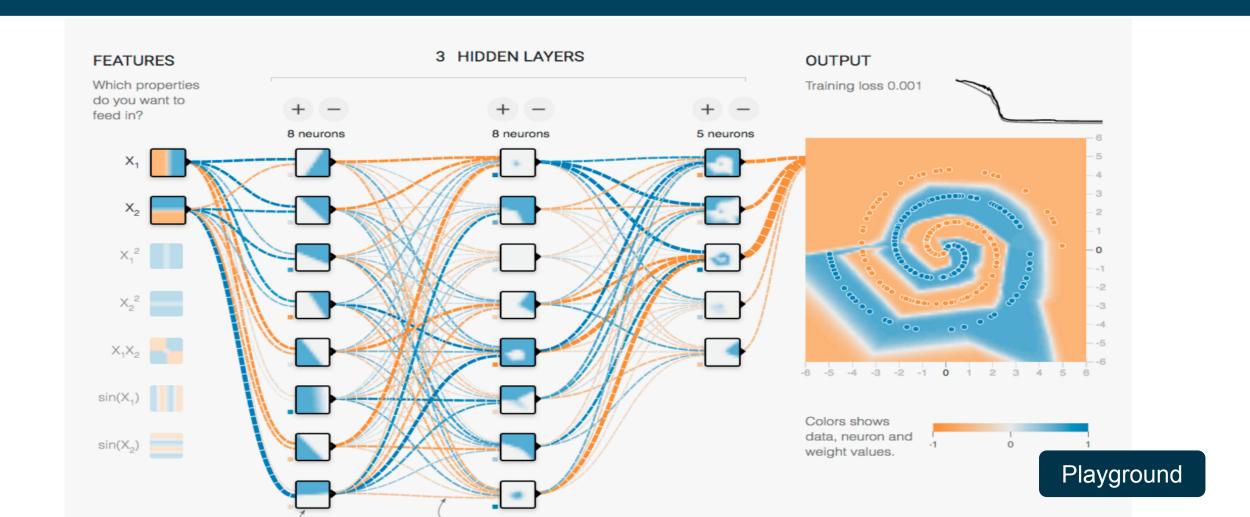# Define a Sequential model

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Add an embedding layer to the model

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Size of the vocabulary

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Size of the embedding space

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Maximum length of input sequences

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None, embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Initializer for the embedding weights

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

The embedding weights should be updated during training

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Add a fully connected layer of 25 hidden units

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```

In code

Calculate the mean of values

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None, embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```
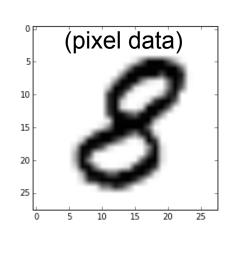
In code

Add a fully connected layer of 10 hidden units

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(1, activation='softmax'))
```
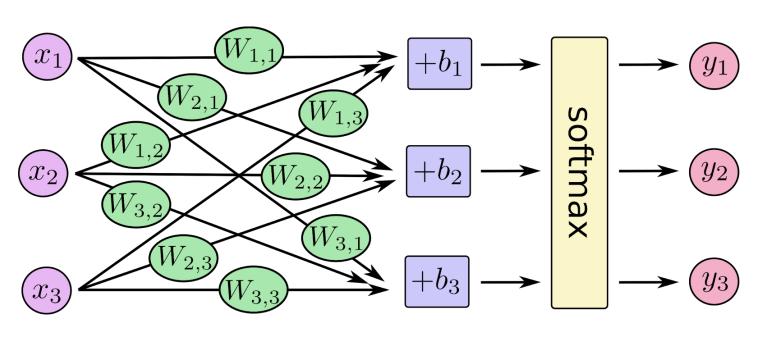
In code

Softmax returns probabilities

```python
model = Sequential()

model.add(Embedding(input_dim=256,
                    output_dim=100,
                    input_length=188,
                    embeddings_initializer=Constant(weights),
                    trainable=True)
)

model.add(Dense(25, activation=leaky_relu))

model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(None,
embedding_dim)))

model.add(Dense(10, activation=leaky_relu))

model.add(Dense(4, activation='softmax'))
```
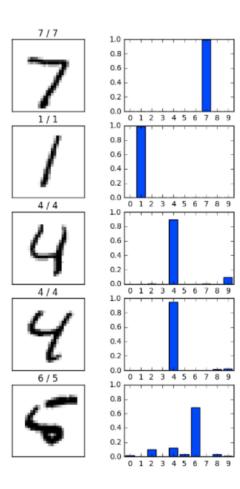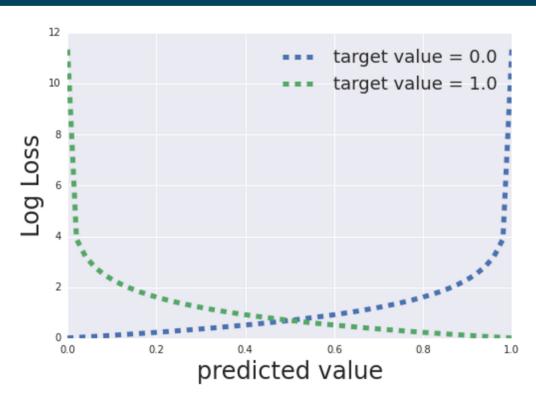
Input vector

(pixel data)

$$p(y = j|\mathbf{x}) = \frac{exp(\mathbf{w}_j^T \mathbf{x} + b_j)}{\sum_{k \in K} exp(\mathbf{w}_k^T \mathbf{x} + b_k)}$$

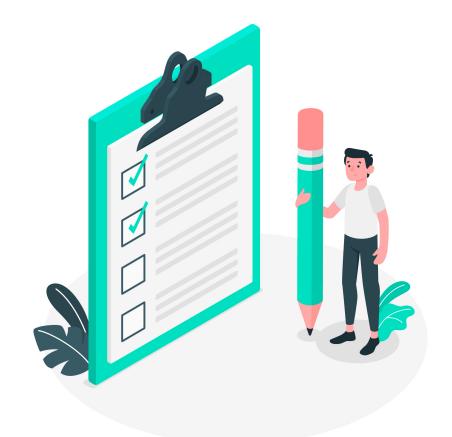# BinaryCrossEntropy (from logits if no sigmoid)

If doing multi class, use CategoricalCrossEntropy

$$LogLoss = \sum_{(\mathbf{x},y)\in D} -y\log(y') - (1-y)\log(1-y'))$$

LAB

News Classifier with NLP

▶ Use a feed forward neura

network to classify news

headlines!

Summary

- Adding more layers to neural networks results in a nonlinear decision boundary.

- This way we can make classification on text.

- FFN have ton of parameters, so they tend to not scale when the vocal is big, or the training set is complex.

- One thing you can do as homework is add fine-tuning of a pertained embedding layer to imporve results!