
بسم الله الرحمن الرحيم

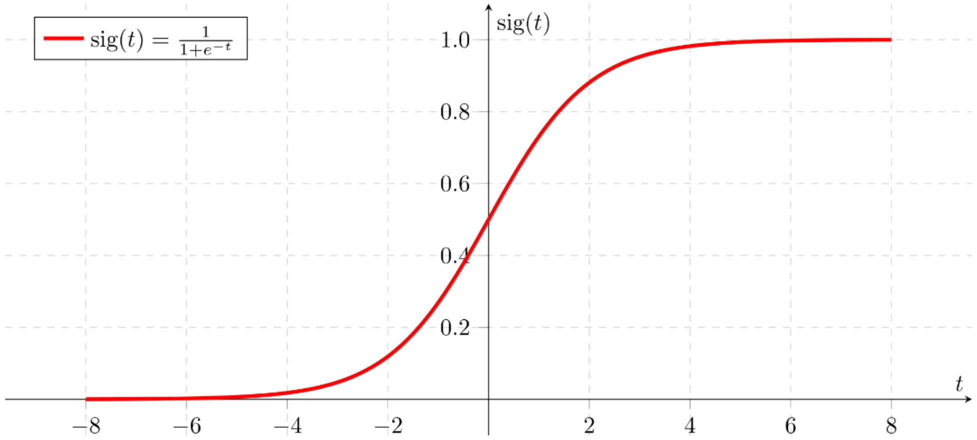
Deep Learning

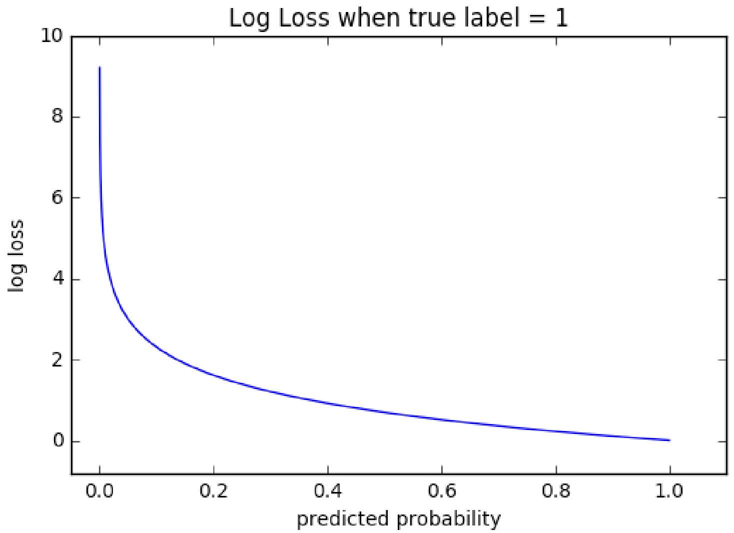
Mohammad Reza Mohammadi

2021

Binary classification

- It is better to use a different approach that ensures there is always a strong gradient whenever the model has the wrong answer
- A sigmoid output unit is defined by $P(y=1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{h} + b)$
- Binary cross-entropy loss: $J(\boldsymbol{\theta}) = -y\log\sigma(z) - (1-y)\log\big(1-\sigma(z)\big)$





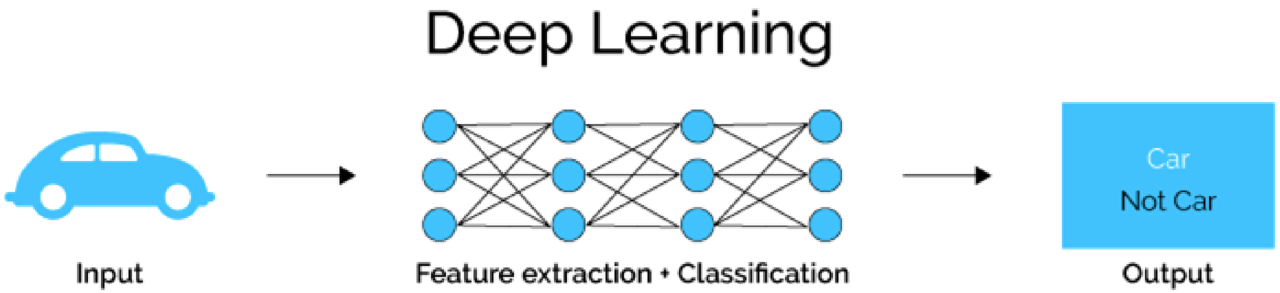
Multiclass classification

- Softmax can be seen as a generalization of the sigmoid function to represent a probability distribution over a discrete variable with n possible values
- We now need to produce a vector $\hat{\boldsymbol{y}}$, with $\hat{y}_i = P(y = i|\boldsymbol{x})$
- First, a linear layer predicts unnormalized log probabilities $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{b}$

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

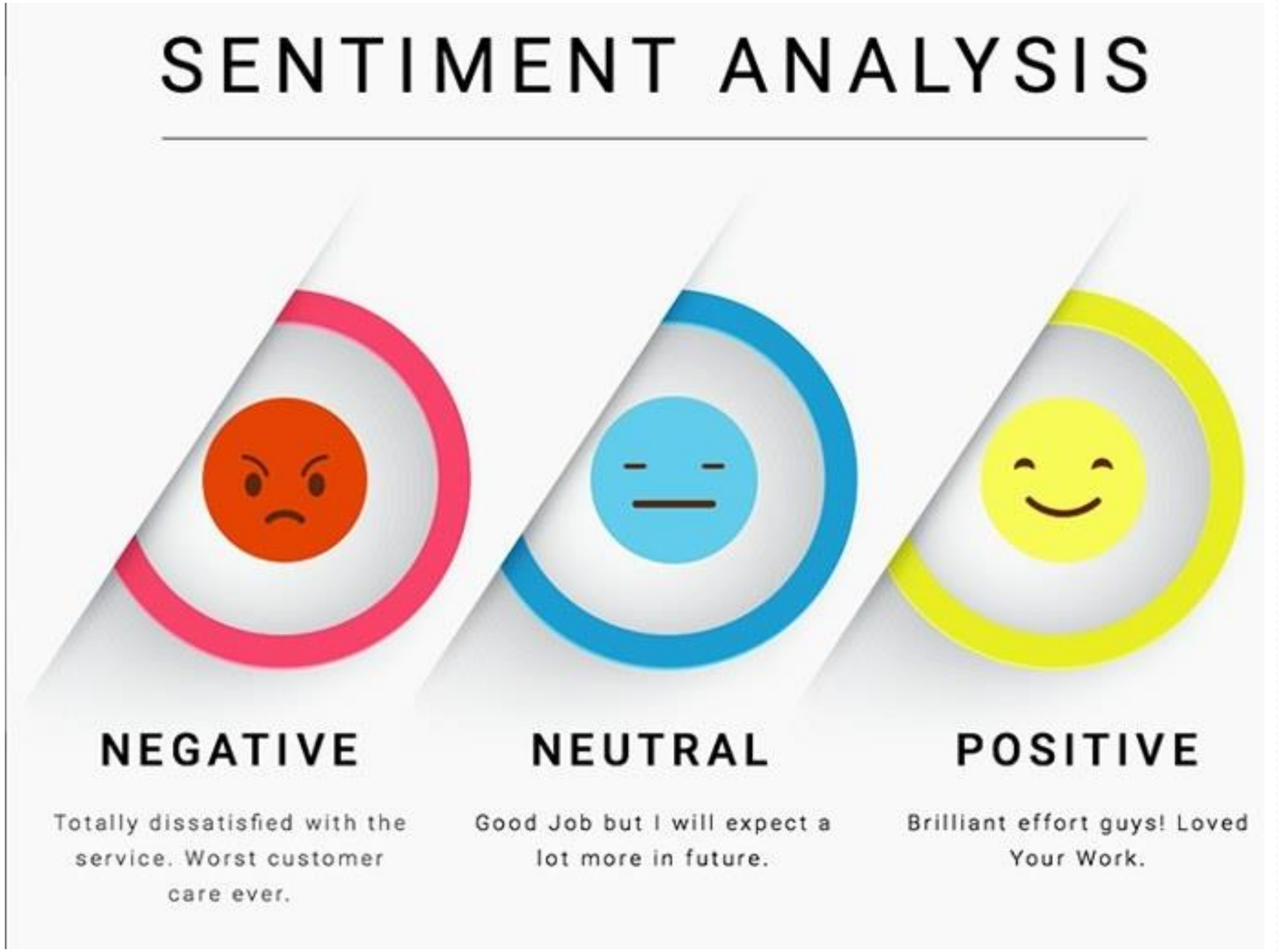
- Training the softmax using cross-entropy

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$



Classifying movie reviews

- IMDB dataset
  - 50,000 highly polarized reviews
  - 25K training, 25K test, each 50-50%
- The reviews (sequences of words) have been turned into sequences of integers, where each integer stands for a specific word in a dictionary



Classifying movie reviews

```
from keras.datasets import imdb
(train_data,train_labels), (test_data,test_labels) = imdb.load_data(num_words=1000)
```

- train_labels and test_labels are lists of 0s and 1s, where 0 stands for negative and 1 stands for positive
- We have to turn the lists (with different lengths) into tensors

```
np.array([len(d) for d in train_data])
```

```
array([218, 189, 141, ..., 184, 150, 153])
```

- Using Embedding layer: Same length by padding the sentences
- One-hot encoding: Vectors of 0’s and 1’s, turning the sequence [3, 5] into a 10,000-dimensional vector that would be all 0s except for indices 3 and 5, which would be 1s

Classifying movie reviews

```
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

```
print(x_train.shape)
print(x_train[0])
```

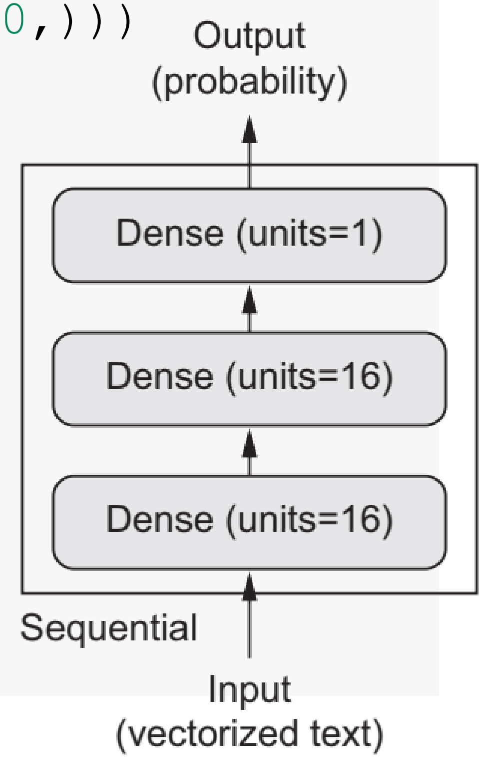
```
(25000, 10000)
[0. 1. 1. ... 0. 0. 0.]
```

```
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

Classifying movie reviews

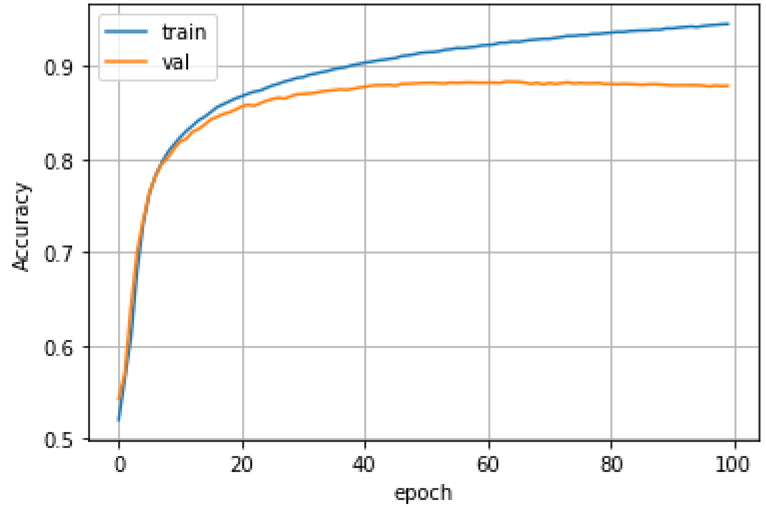
- Building the network

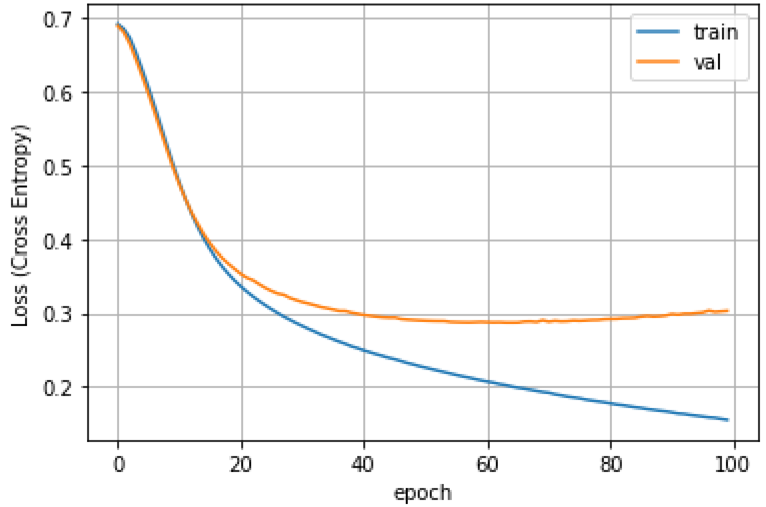
```
model = keras.models.Sequential()
model.add(keras.layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    validation_data=(x_val, y_val),
                    epochs=100,
                    batch_size=512)
```

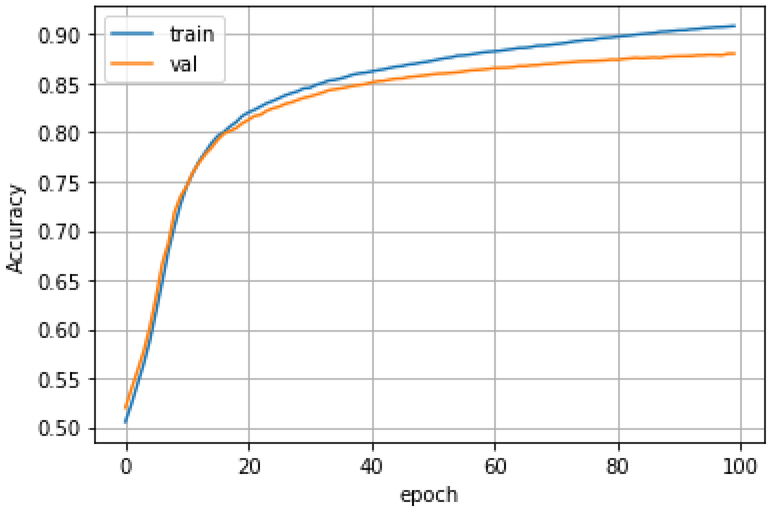


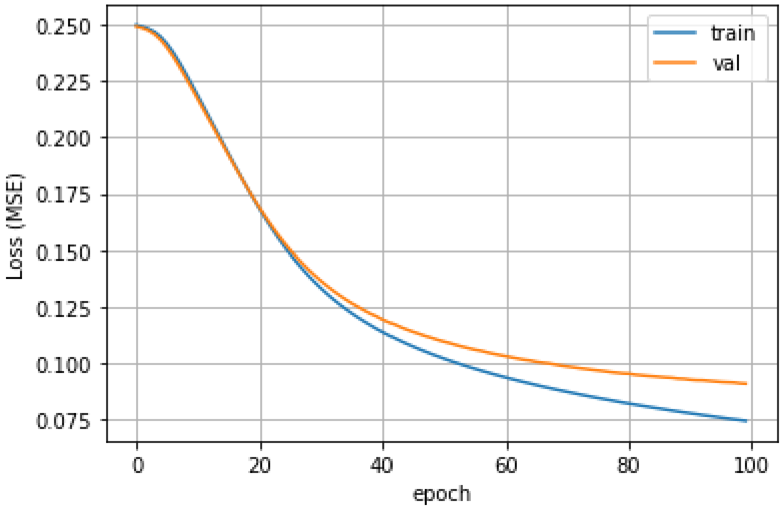
Classifying movie reviews

- SGD optimizer



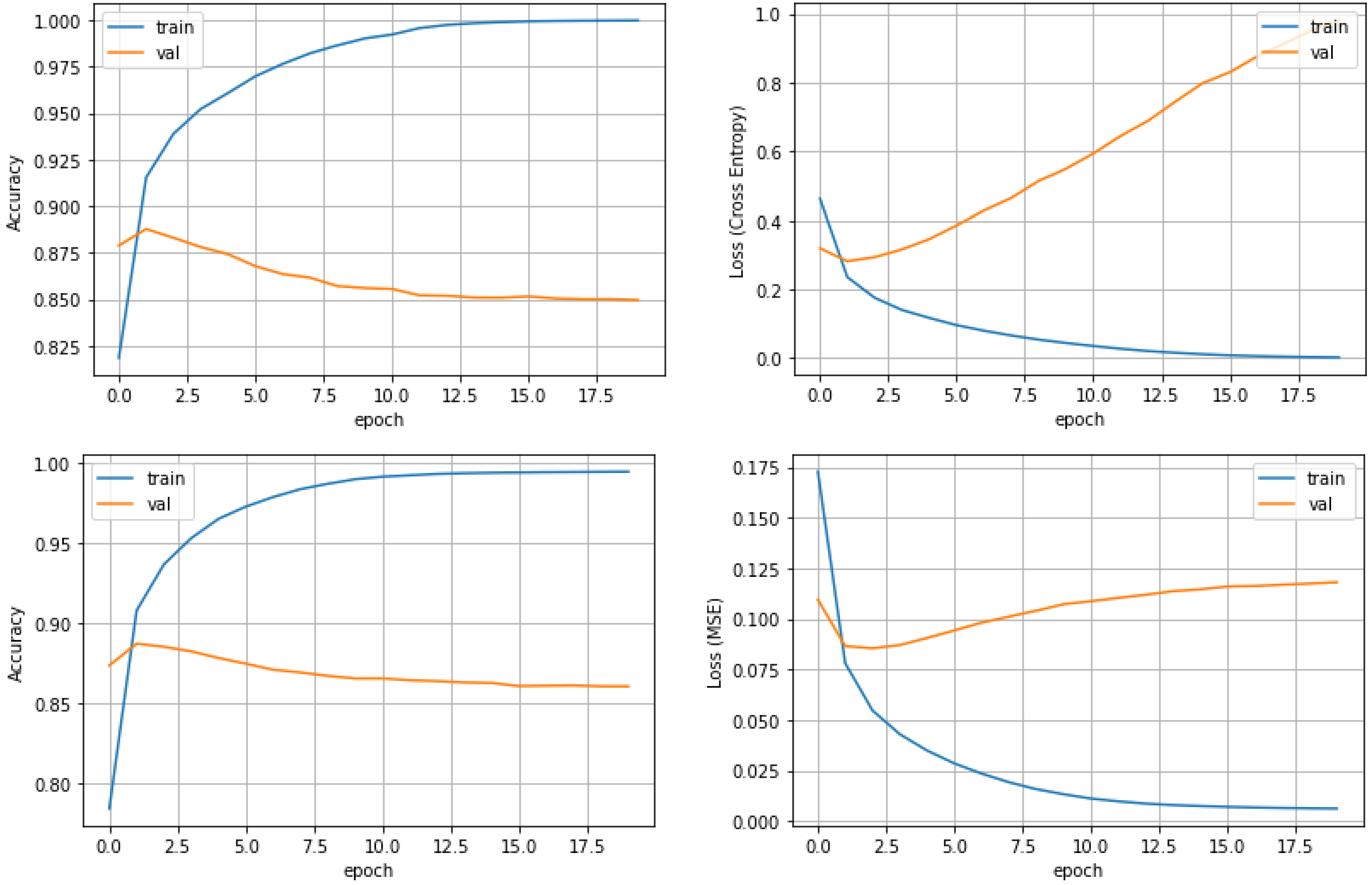






Classifying movie reviews

- Adam optimizer



Classifying newswires

- Classifying Reuters newswires into 46 mutually exclusive topics
- Each data point should be classified into only one category, the problem is more specifically an instance of single-label, multiclass classification
- The Reuters dataset:
  - a set of short newswires and their topics

Classifying newswires

```
from keras.datasets import reuters
(train_data, train_labels), (test_data,test_labels) = reuters.load_data(num_words=10000)
```

- 8,982 training examples and 2,246 test examples

```
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

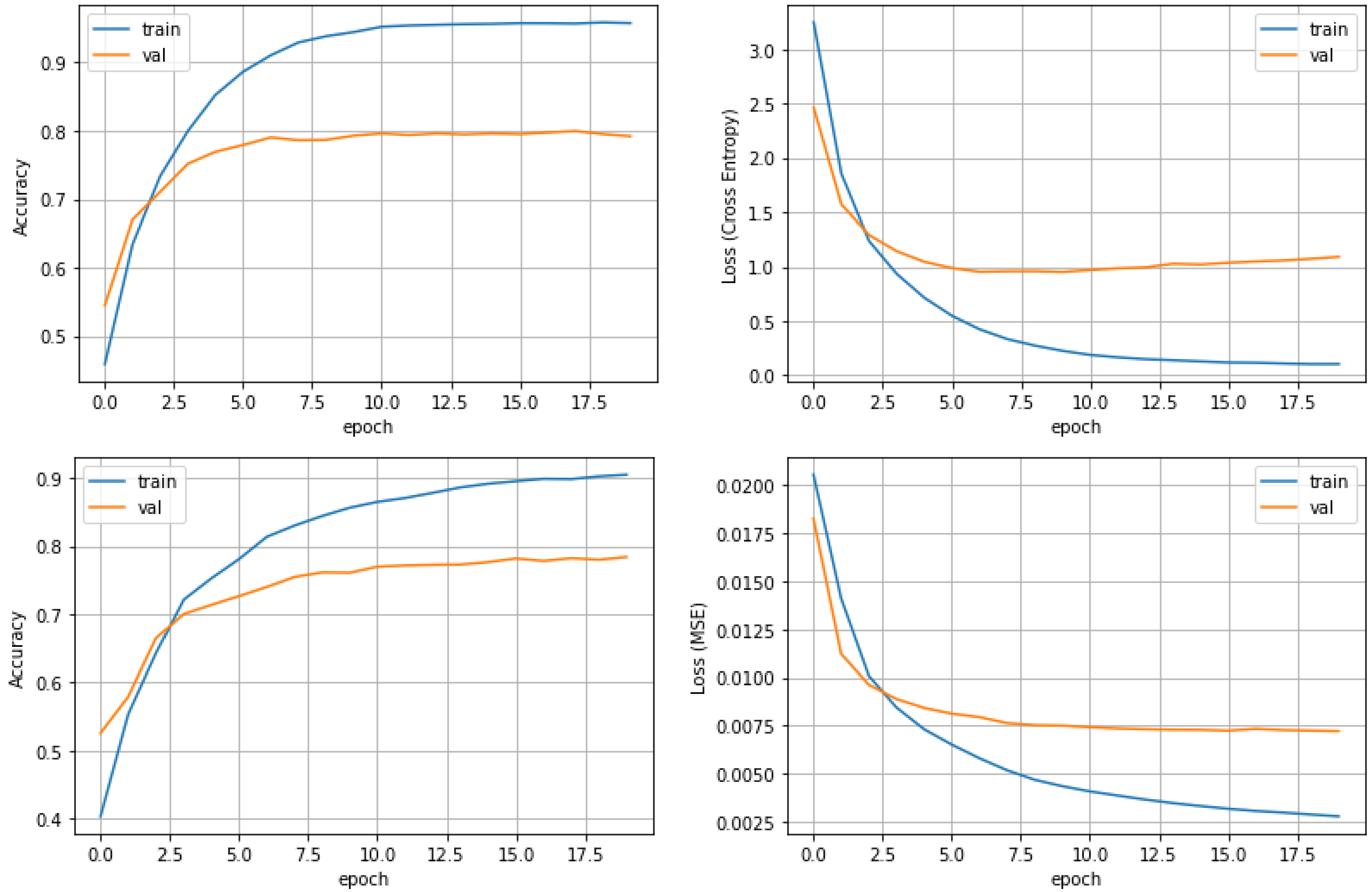
```
y_train = keras.utils.to_categorical(train_labels)
y_test = keras.utils.to_categorical(test_labels)
```

Classifying newswires

- Building the network

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(46, activation='softmax'))

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

history = model.fit(x_train, y_train,
                    validation_data=(x_test, y_test),
                    epochs=20,
                    batch_size=512)
```

Classifying newswires



Information bottleneck

- Intermediate layers should not be significantly smaller than the final one (46)
- For instance, having a 4-dimensional intermediate layer would reduce the accuracy from 80% to 71%
- The network is unable to cram all necessary information into this representation

Other Output Types

- The linear, sigmoid, and softmax output units described above are the most common
- Neural networks can generalize to almost any kind of output layer that we wish
- The principle of maximum likelihood provides a guide for how to design a good cost function for nearly any kind of output layer

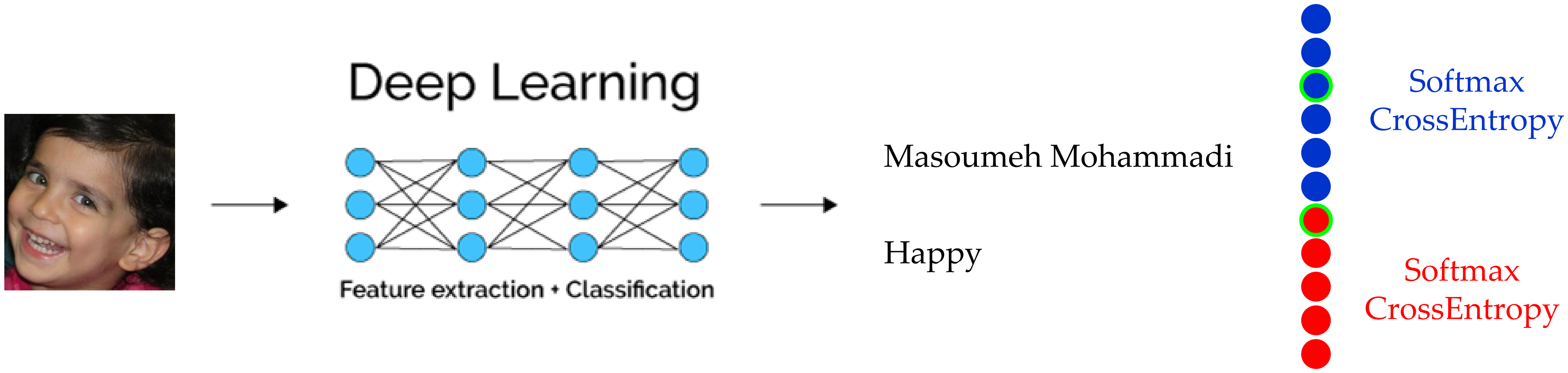
Last-layer activation and loss function

Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

Multi-task learning

- In multi-task learning, multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks
- This can result in improved learning efficiency and prediction accuracy for the task-specific models, when compared to training the models separately

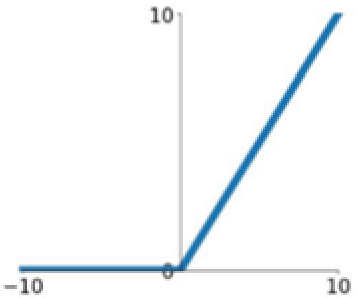


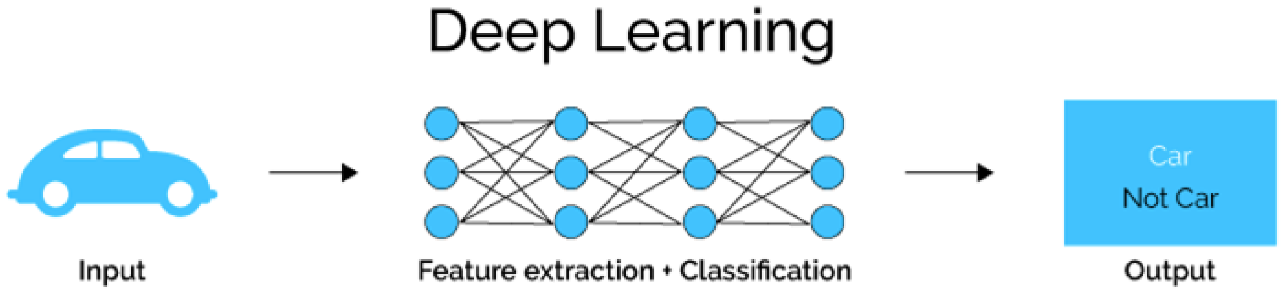
Hidden Units

- The design of hidden units is an extremely active area of research and does not yet have many definitive guiding theoretical principles
- Most hidden units can be described as accepting a vector of inputs x, computing an affine transformation $\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}$, and then applying an element-wise nonlinear function $g(\boldsymbol{z})$
- Rectified linear units are an excellent default choice for hidden units

ReLU

$\max(0, x)$



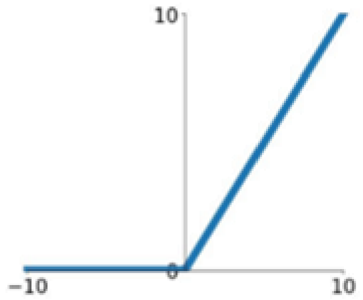


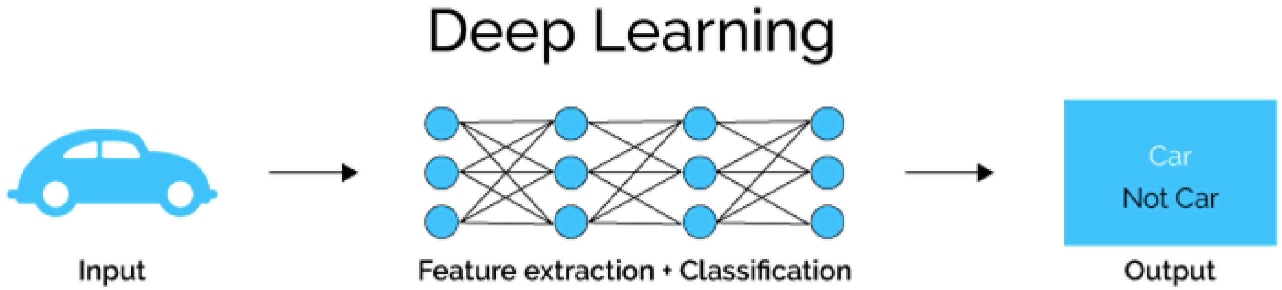
Rectified Linear Units

- ReLUs are easy to optimize because they are so similar to linear units
- Derivatives through a ReLU remain large whenever the unit is active
- Several generalizations of ReLU exist
- One drawback to rectified linear units is that they cannot learn via gradient-based methods on examples for which their activation is zero

ReLU

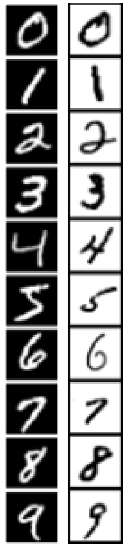
$\max(0, x)$

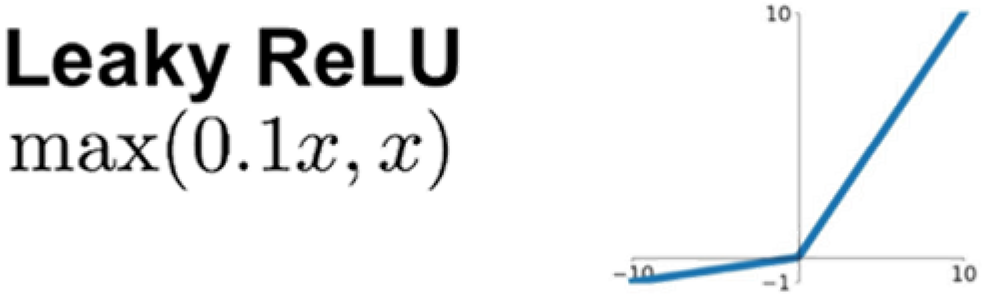




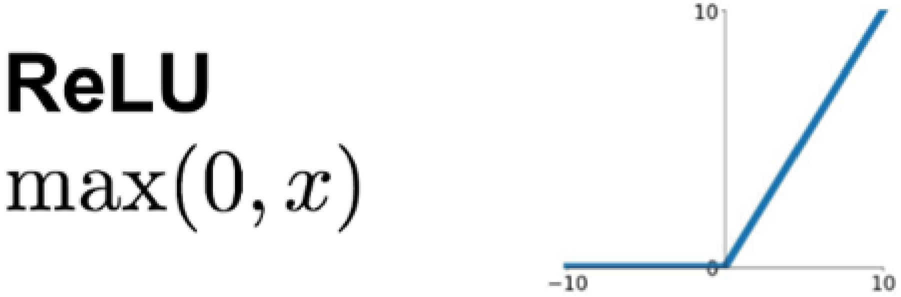
ReLU with non-zero slope

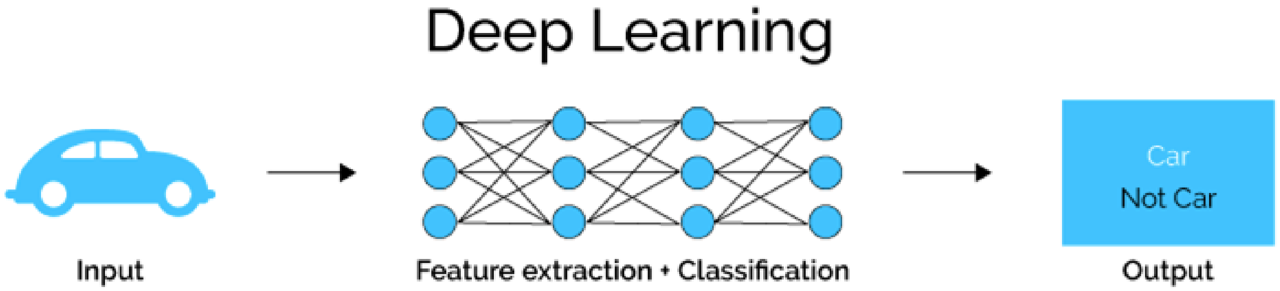
- Absolute value rectification fixes $\alpha_i = -1$ to obtain $g(z) = |z|$
- A leaky ReLU fixes α_i to a small value like 0.01
- A parametric ReLU (PReLU) treats α_i as a learnable parameter





$$h_i = g(\boldsymbol{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \min(0, z_i)$$





Logistic Sigmoid and Hyperbolic Tangent

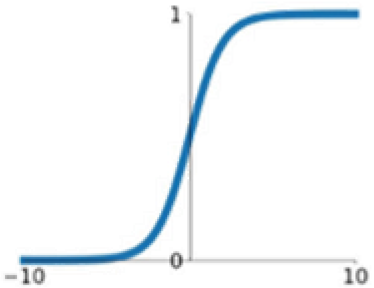
- Prior to the introduction of ReLU, most neural networks used the logistic sigmoid activation function or the hyperbolic tangent activation function

$$\tanh(z) = 2\sigma(2z) - 1$$

- Sigmoidal units saturate across most of their domain
- The widespread saturation of sigmoidal units can make gradient-based learning very difficult

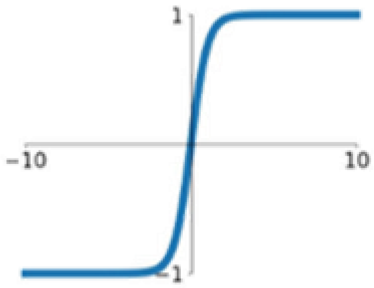
Sigmoid

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



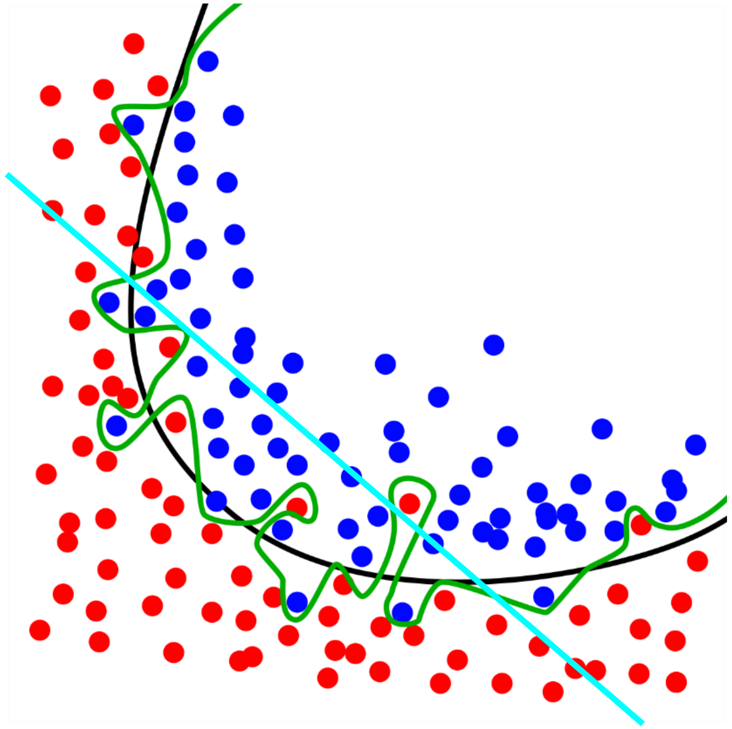
tanh

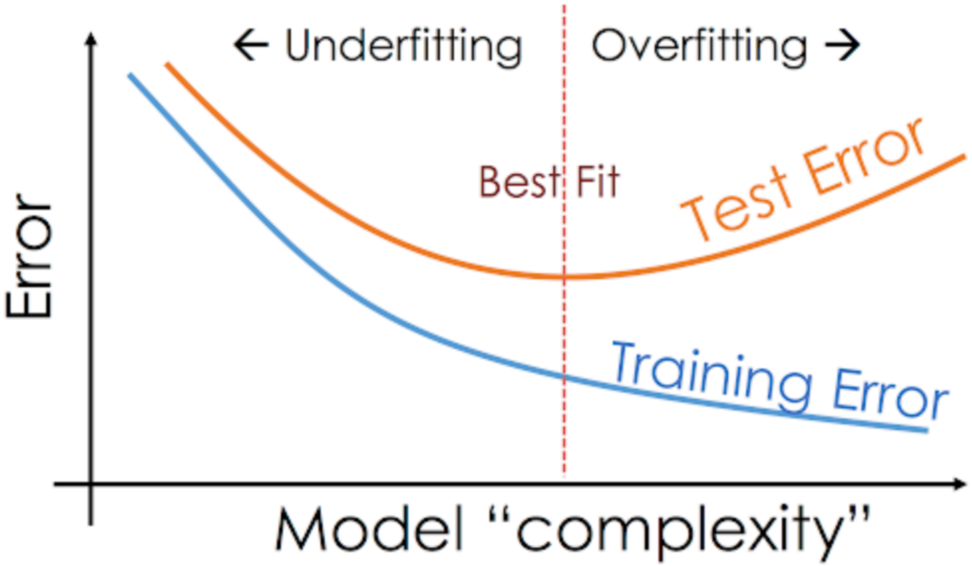
$\tanh(x)$



Regularization for Deep Learning

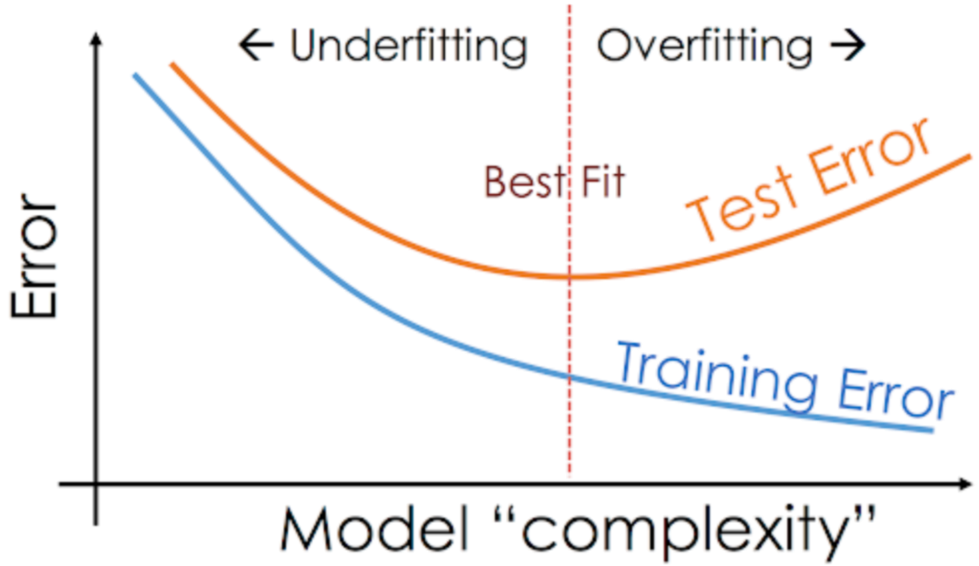
- A central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also on new inputs





Regularization for Deep Learning

- A central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also on new inputs
- Many strategies used in ML are explicitly designed to reduce the test error, possibly at the expense of increased training error
- These strategies are known as regularization



Bias and Variance

- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict
- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data