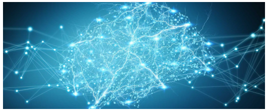


ARTIFICIAL NEURAL NETWORKS - WEEK 4

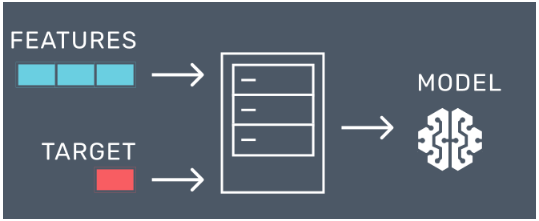
Training Neural Networks

Dr. Aamir Akbar

Director of both AWKUM AI Lab and AWKUM Robotics, Final Year Projects (FYPs) coordinator, and lecturer at the department of Computer Science
Abdul Wali Khan University, Mardan (AWKUM)

CONTENTS

TRAINING A NEURAL NETWORK MODEL

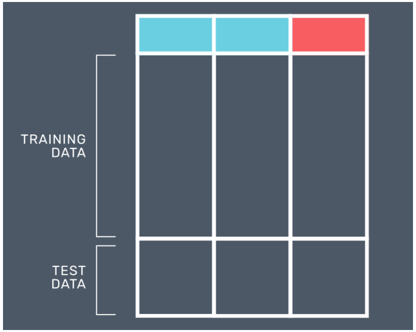


Given a dataset containing features and target, we get a `model`.

During training, the neural network learns from the data and updates its parameters. By this point, we'll have a `trained model`.

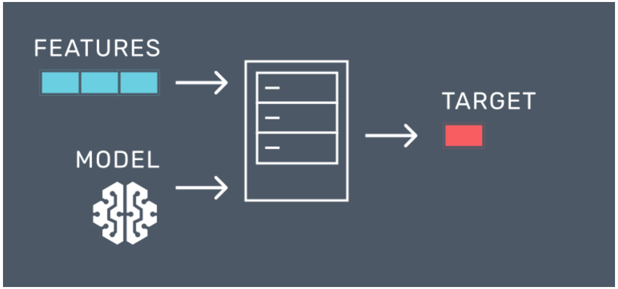
The training process is sometimes also called `fitting the model to the data`.

TRAIN/TEST DATA SPLIT



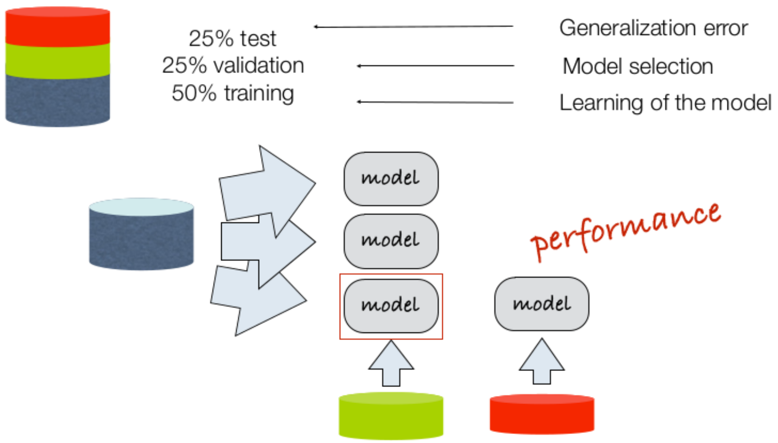
Once the training is complete, we need a way to know `how the model is performing`. For this, we'll keep a portion of the dataset for `testing`.

TESTING A NEURAL NETWORK MODEL

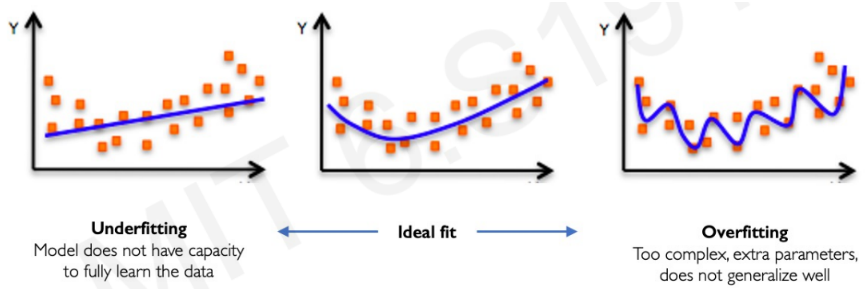


During testing, we'll provide the neural network with just the features, without the target. Now that it's already trained, it's job is to `predict the target values`.

IDEAL MODELING SETUP IN PRACTICE



WHAT IS IDEAL FIT?



BIAS AND VARIANCE

1. If your model is `underfitting` it has a `high bias`.
2. If your model is `overfitting` then it has a `high variance`
3. Your model will be alright if you balance the Bias / Variance.

How to get the ideal fit?

If your algorithm has a `high bias`:

1. Try to make your Neural Network bigger (size of hidden units, number of layers)
2. Try to use a different model that is suitable for your data.
3. Longer training.
4. Try different optimization algorithms.

If your algorithm has a `high variance`:

1. More data.
2. Try a different model that is suitable for your data.
3. Use regularization.

REGULARIZATION

what is Regularization?

Regularization is a method to prevent the model from memorizing the training data too closely. It helps by adding constraints that encourage simpler patterns, reducing the risk of `overfitting` and improving the model's ability to `generalize` to new, unseen data.

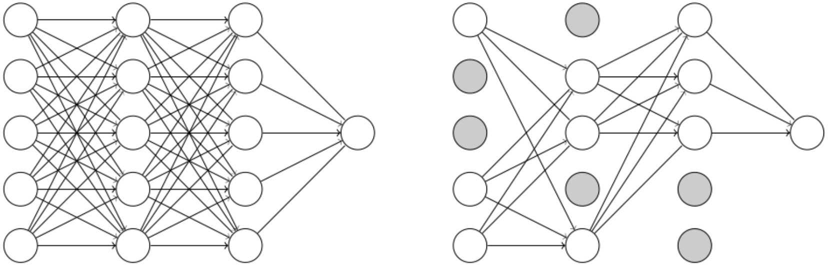
When training a model, we aim to find the best values for its parameters (weights and biases) so that it performs well on the training data, such that:

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i; W))$$

This process is about minimizing what's called `empirical risk`, which is essentially the average error on the training data.

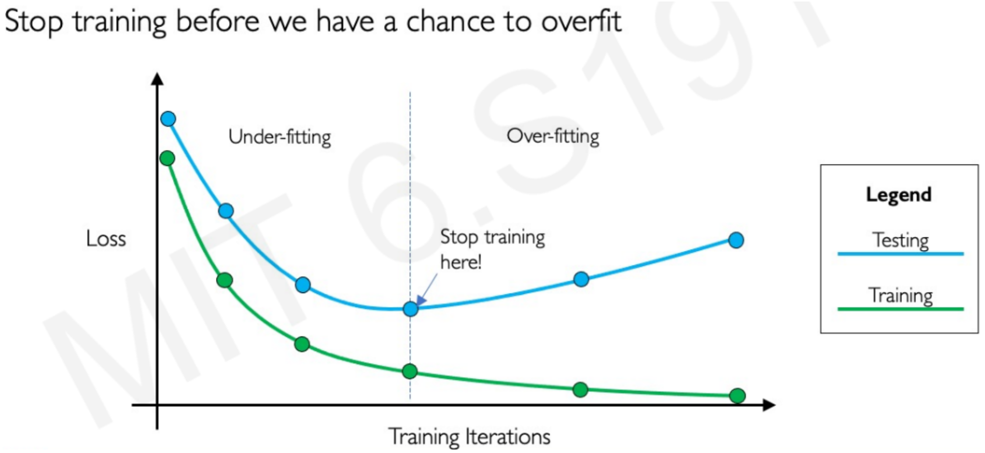
However, our ultimate goal is to make the model perform well on any new, unseen data as well. This is called `true risk`, and it's what we also need to minimize.

REGULARIZATION: DROPOUT



The idea behind DropOut is to `switch off` some of the neurons during training. The switched off neurons change at each mini-batch such that, overall, all neurons are trained during the whole process.

REGULARIZATION: EARLY STOPPING



CASE STUDY: COUNTERFEIT BANKNOTE DETECTION

How to classify whether a banknote is counterfeit?

Features such as `variance`, `skewness`, `curtosis and entropy` capture different aspects of the characteristics of banknotes, such as texture, distribution, and randomness, which can be useful for distinguishing between genuine and counterfeit banknotes in a classification task.

`Varience` measures the spread or dispersion of data points in a dataset. In the context of banknotes, the variance of certain features (such as pixel intensities if using image data) might capture information about the texture or patterns on the banknote. For example, genuine banknotes might have more consistent texture patterns, resulting in lower variance, while counterfeit banknotes might exhibit more irregular patterns, leading to higher variance.

`Skewness` measures the asymmetry of the distribution of data points. It indicates whether the data is skewed towards larger or smaller values relative to the mean. In the context of banknotes, skewness could capture asymmetries in certain features' distributions, such as the intensity distribution of certain regions on the banknote. For instance, counterfeit banknotes might exhibit skewed intensity distributions due to irregular printing or ink patterns.

CASE STUDY: COUNTERFEIT BANKNOTE DETECTION

`Curtosis` measures the peakedness or flatness of the distribution of data points. It indicates whether the data has heavy tails or is more concentrated around the mean. In the context of banknotes, curtosis might capture the sharpness or flatness of certain features' distributions, such as the sharpness of edge patterns or the concentration of ink marks. Counterfeit banknotes might exhibit different curtosis values compared to genuine banknotes due to differences in printing techniques or materials.

`Entropy` is a measure of randomness or uncertainty in a dataset. In the context of banknotes, entropy could capture the randomness or complexity of certain features, such as the texture or patterns present on the banknote surface. For example, genuine banknotes might exhibit lower entropy values if they have more structured and predictable patterns, while counterfeit banknotes might have higher entropy values due to irregular or random patterns.

CASE STUDY: COUNTERFEIT BANKNOTE DETECTION

```
   variance  skewness  curtosis  entropy  class
0  -0.89569   3.00250 -3.606700 -3.44570      1
1   3.47690  -0.15314  2.530000  2.44950      0
2   3.91020   6.06500 -2.453400 -0.68234      0
3   0.60731   3.95440 -4.772000 -4.48530      1
4   2.37180   7.49080  0.015989 -1.74140      0
5  -2.21530  11.96250  0.078538 -7.78530      0
6   3.94330   2.50170  1.521500  0.90300      0
7   3.93100   1.85410 -0.023425  1.23140      0
8   3.97190   1.03670  0.759730  1.00130      0
9   0.55298  -3.46190  1.704800  1.10080      1
```

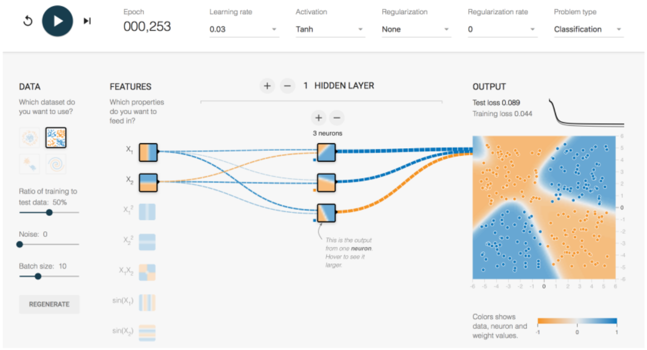
CASE STUDY: COUNTERFEIT BANKNOTE DETECTION

```
Model: "bank_note_classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 4)]               0
 dense (Dense)               multiple                  35
 dropout (Dropout)           multiple                  0
 dense_1 (Dense)             multiple                  56
 dropout_1 (Dropout)         multiple                  0
 dense_2 (Dense)             multiple                  8
=================================================================
Total params: 99
Trainable params: 99
Non-trainable params: 0
_________________________________________________________________
```

TENSORFLOW PLAYGROUND

Try both classification and regression tasks with different combinations of parameters, features selection, layers, number of neurons, etc.

http://playground.tensorflow.org



Epoch 000,253

Learning rate 0.03

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

Test loss 0.089

Training loss 0.044

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

Colors shows data, neuron and weight values.

RESOURCES

To download the source codes used in the previous slides, follow the link:

▸ Download Source Codes

Import the codes into your preferred development environment, such as Visual Studio Code (VS Code), to practice and explore further.

To learn programming in Python, follow my comprehensive 15-week Programming in Python course at:

▸ Programming in Python