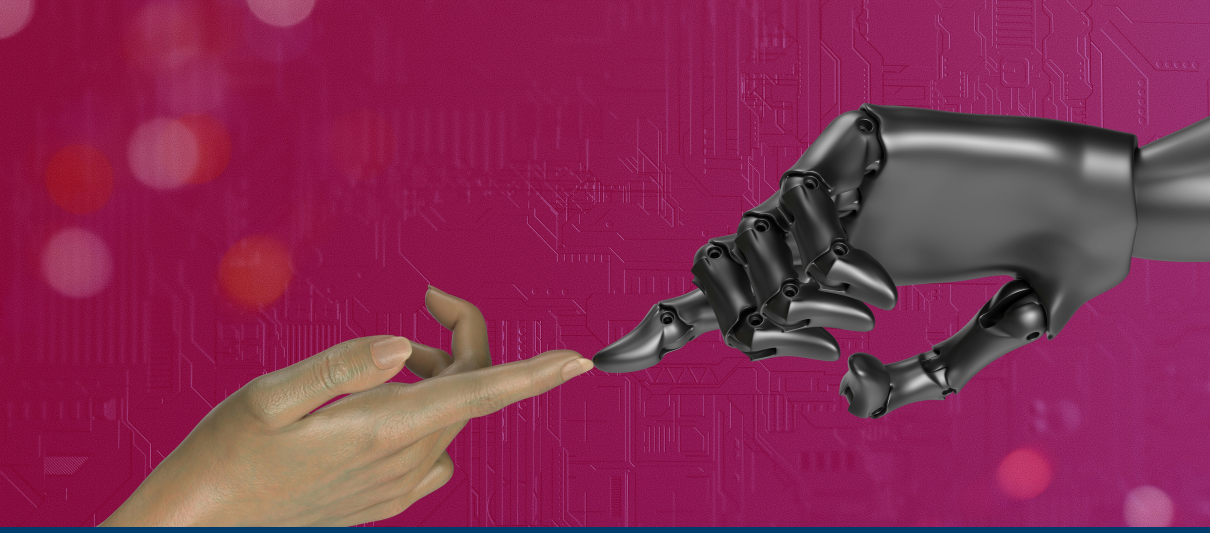


Hands-On Machine Learning

Dr. Derek Bridge | 2026/2027

UCC Coláiste na hOllscoile Corcaigh
University College Cork, Ireland

School of Computer Science and Information Technology
Scoil na Ríomheolaíochta agus na Teicneolaíochta Faisnéise

19. Overfitting in Neural Networks

In this lecture, we use the MNIST dataset again.

Overfitting

- One of the central problems of deep learning is overfitting.
- Reminder. If your model overfits, your main options are:
  - Gather more training examples;
  - Change to a less complex model;
  - Clean the training examples to remove noise.
  - Dimensionality reduction/feature selection: to reduce the number of features.
  - Parametric models: use regularization.
  - Decision Trees: use post-pruning.
  - Neural Networks: use dropout; use batch-normalisation.
  - Gradient Descent: use early-stopping.
  - Use an ensemble (although the relationship between ensembles and overfitting is not well-understood).

What we will look at

- early stopping;
- reducing the network's size to give a less complex model;
- weight regularization;
- dropout;
- data augmentation — creating synthetic training examples, rather than gathering more real training examples.

Early Stopping

- We know that a sign of overfitting is that validation error stops getting lower and starts getting larger.
- We can exploit this during Gradient Descent as a way of avoiding overfitting, known as **early stopping**.
- During Gradient Descent, monitor validation error (or loss).
- Interrupt training when the validation error has stopped improving for a certain number of epochs.

Reducing Network Size

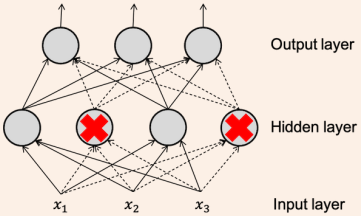
- We can make the model (neural network) less complex by reducing the number of parameters.
- Obviously enough, this is achieved by:
  - reducing the number of hidden layers, and/or
  - reducing the number of neurons within the hidden layers.

Weight Regularization

- For models that have parameters, we used regularization to ensure that the parameters took only small values by penalizing large values in the loss function.
  - Lasso: we penalized by the l_1-norm (the sum of their absolute values).
  - Ridge: we penalized by the l_2-norm (the sum of their squares).
  - A hyperparameter, λ, called the 'regularization parameter', controlled the balance between fitting the data versus shrinking the parameters.
- Weight Regularization in neural networks is the same idea, but applied to the weights in the layers of a network.
- Weight regularization can work well when the network is small but has little effect on larger networks.
- For larger networks, a better option can be dropout.

Dropout

- Imagine we have a layer that uses **dropout** with **dropout rate** p, e.g. $p = 0.5$.
- Then, in a given step of the backprop algorithm, each neuron in the layer has probability p of being ignored — treated as if it were not there.



One way of doing dropout

- Training. For any given mini-batch:
  - In the forward propagation,
    - decide which neurons will be dropped (chosen with probability p);
    - set the activations of the dropped neurons to zero;
    - multiply the activations of the kept neurons by $1/(1-p)$.
  - In the backpropagation, ignore the dropped out neurons.

  Note that different neurons will get dropped for each mini-batch.
- Inference. No change.

But why did we multiply activations by $1/(1-p)$?

- In inference, for $p = 0.5$, a neuron in the next layer will receive input from on average twice as many neurons as it did in training.
- The multiplication by $1/(1-p)$ compensates for this.

Why does dropout reduce overfitting?

- Consider a company whose employees were told to toss a coin every monring to decide whether to go to work or not.
- The organization would need to become more resilient. It could not rely on any one employee to perform critical tasks: the expertise would need to be spread across many employees. They must become more like generalists, less like specialists.
- Similarly, in dropout layers, neurons learn more robust features.

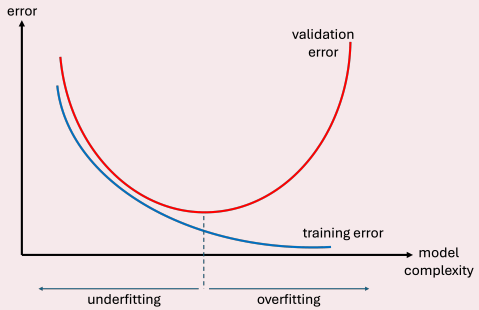
- Since a neuron can be present or absent, it's like training on a different neural network at each step.
- The final result is a bit like an ensemble of these many different virtual neural networks.
- However, it typically increases the number of epochs needed for convergence (roughly double when $p = 0.5$).

Data Augmentation

- Another way to reduce overfitting is to get more examples but here we do something similar: **data augmentation**.
- We augment the training set with examples that we synthesize from the existing training set.
- It's relatively easy when the dataset consists of images to use transformations on existing images to synthesize believable-looking new images.
- Be aware that this is not as good as additional real examples. These synthesized examples are correlated with each other and the originals from which they were generated.
- Augmentation layers in a neural network will only augment the training data, not the validation or test data.

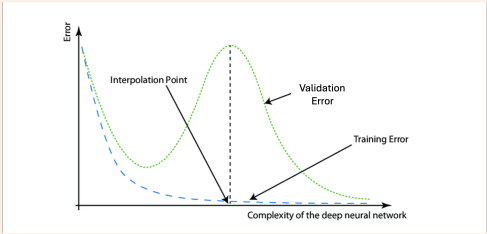
The Conventional Wisdom

- For low complexity models (e.g. small neural networks), training error and validation error are both high (underfitting).
- But as we increase complexity (e.g. more network layers), both decrease. But, after a certain point, validation error starts to rise again (overfitting).



Double-Descent

- In fact, with deep learning, we are finding something counter-intuitive. If we keep increasing the complexity, then validation error decreases again, falling to a new minimum. This is called **double descent**.



Over-Paramtereization and the Interpolation Threshold

- We are finding this happens with **over-parameterized** models — ones that are excessively complex.
- The current theory, in high-level terms, is that the over-parameterised model makes smoother interpolations between the training examples, and this is what gives the better validation error.

If you're interested

- A visualization
- An explanation

Image credits

Image based on one from A. Géron: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (2nd edn), O'Reilly, 2019. The analogy between dropout and a company whose employees are told to toss a coin to decide whether to go to work each morning comes from the same book.