UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

SCIENCE

COMPSCI 760
Advanced Neural Networks

Deep Learning: Lecture 1

Learning outcomes

Lecture 1: Deep Neural Networks Review

- Review what a deep neural network is and the differences classical ML approaches.
- Review the different steps involved in training a deep NN.
- Review network initialisation and the different activation functions.
- Review what the hyperparameters of a DNN are.
- Review the different strategies to improve the performance of a deep NN.
- Review the different strategies to tune a deep NN.

Lectures 2 & 3: Learning with sequences (RNNs, Transformers, LLMs)

- Understand how recurrent neural networks work.
- Recognise commonly used neural network architectures based on RNN (LSTM, GRU).
- Understand how transformers work.
- Understand the principles of Large Language models.

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

Disclaimer/Acknowledgment:

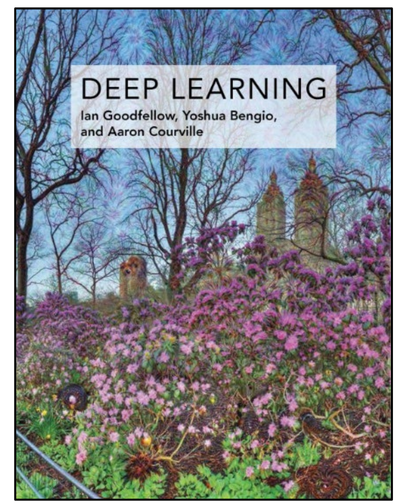
The following slides are reusing some of the content created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.

All CS188 materials are available at http://ai.berkeley.edu.

Some content is also reused from the CS230 Stanford course slides created by Andrew Ng, available at https://cs230.stanford.edu/syllabus/.

Lecture 1: Deep Neural Networks Review

- Review what a deep neural network is and the differences classical ML approaches.
- Review the different steps involved in training a deep NN.
- Review network initialisation and the different activation functions.
- Review what the hyperparameters of a DNN are.
- Review the different strategies to improve the performance of a deep NN.
- Review the different strategies to tune a deep NN.



Recommended read:

https://www.deeplearningbook.org/

Part II Deep Networks

Part I Applied Math and Machine Learning Basics

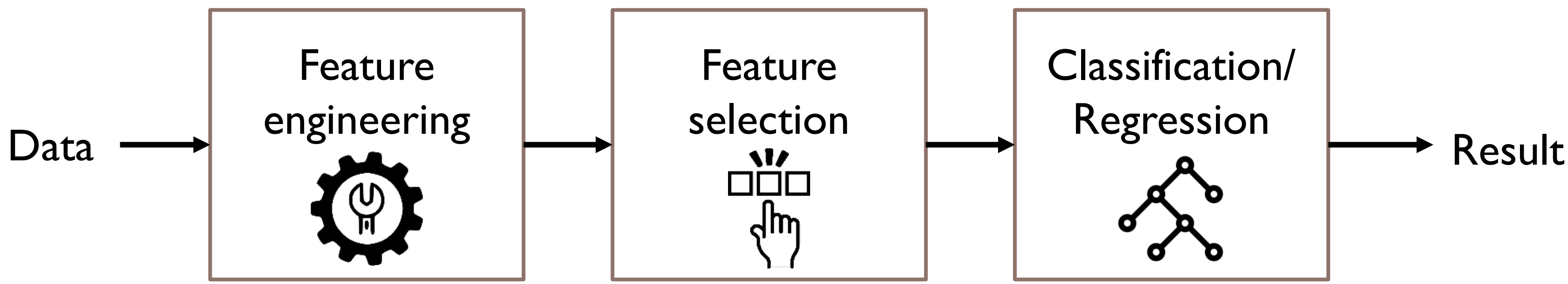
Subscribe to keep updated of ML/AI news:

https://read.deeplearning.ai/the-batch/

DeepLearning.AI
THE BATCH

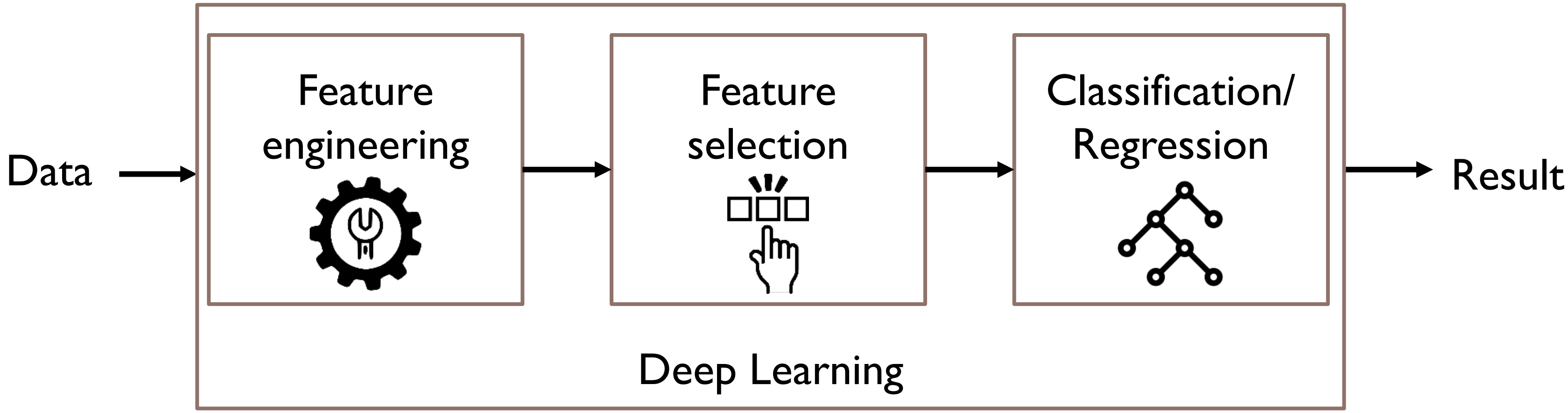
Classical Machine Learning

- The classical approach to ML:



- Need expert knowledge about the data to design features
- Can be complex to design or/and select good features
- Not best use of the huge amount of data now available
- Hard to transfer

Why Deep Learning?

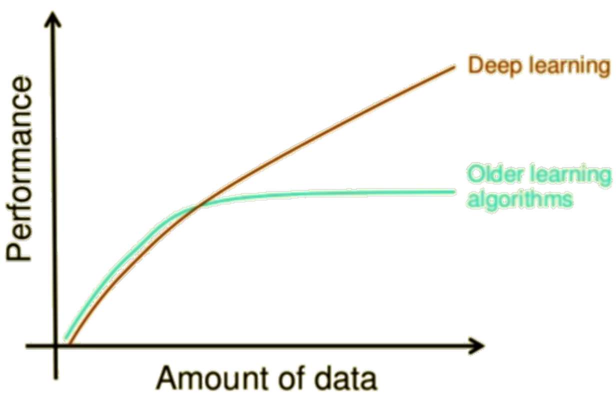


- No need for feature engineering and selection anymore.
- Scale with the amount of training data.
- Easier to transfer to other tasks/domains.
- Better performances on a lot of tasks/domains!

https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa

The death of classical ML?

- Can still be useful in specific situations!



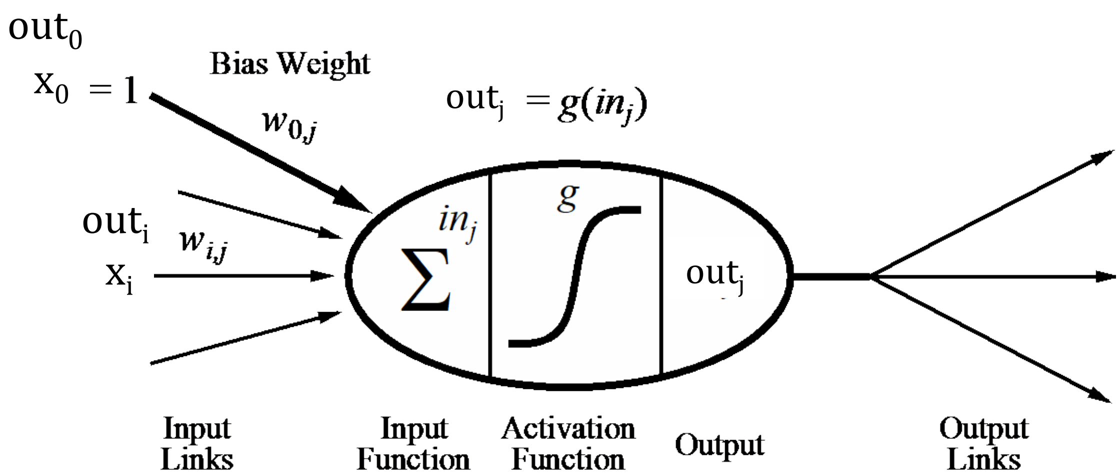
Small datasets

Ease of interpretation

Low computational resources

https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa

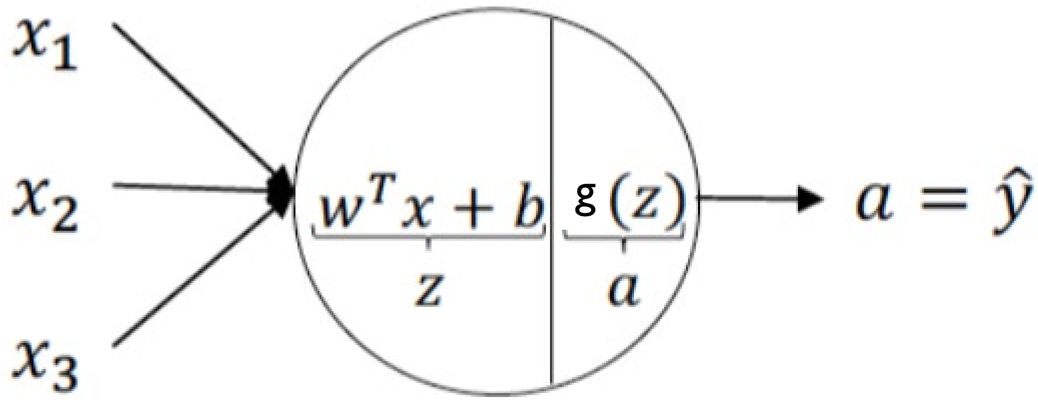
Simple model of a neuron (McCulloch & Pitts, 1943)



- Inputs x_i come from the output of node i to this node j (or from "outside")
- Each input link has a ***weight*** $w_{i,j}$
- There is an additional fixed input x_0 with ***bias*** weight $w_{0,j}$ (or b_j)
- The total input is $in_j = \Sigma_i w_{i,j} x_i$ (or $\Sigma_i w_{i,j} x_i + b_j$)
- The output is $out_j = g(in_j) = g(\Sigma_i w_{i,j} x_i)$ (or $g(\Sigma_i w_{i,j} x_i + b_j)$)

 Dan Klein and Pieter Abbeel

Linear algebra representation

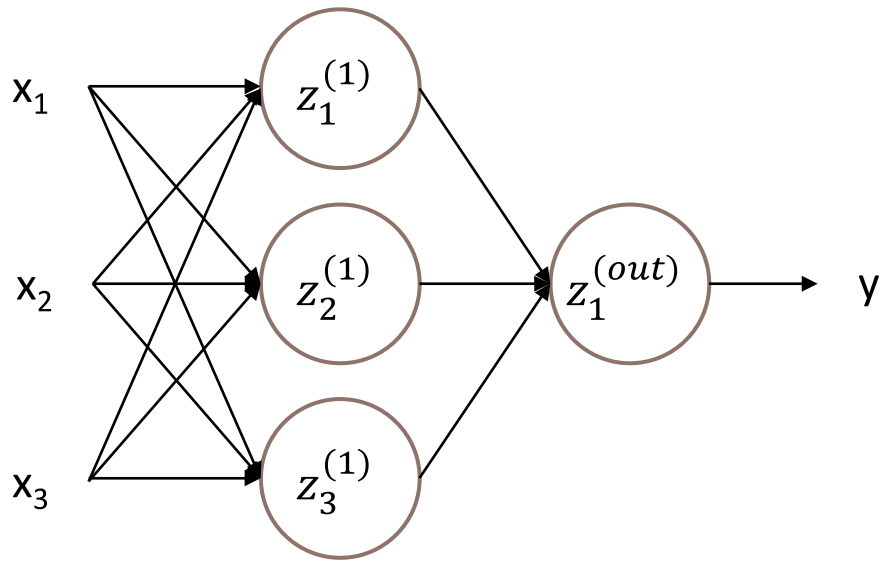


$$z = w^T x + b$$

$$a = g(z)$$

Andrew Ng

Neural network



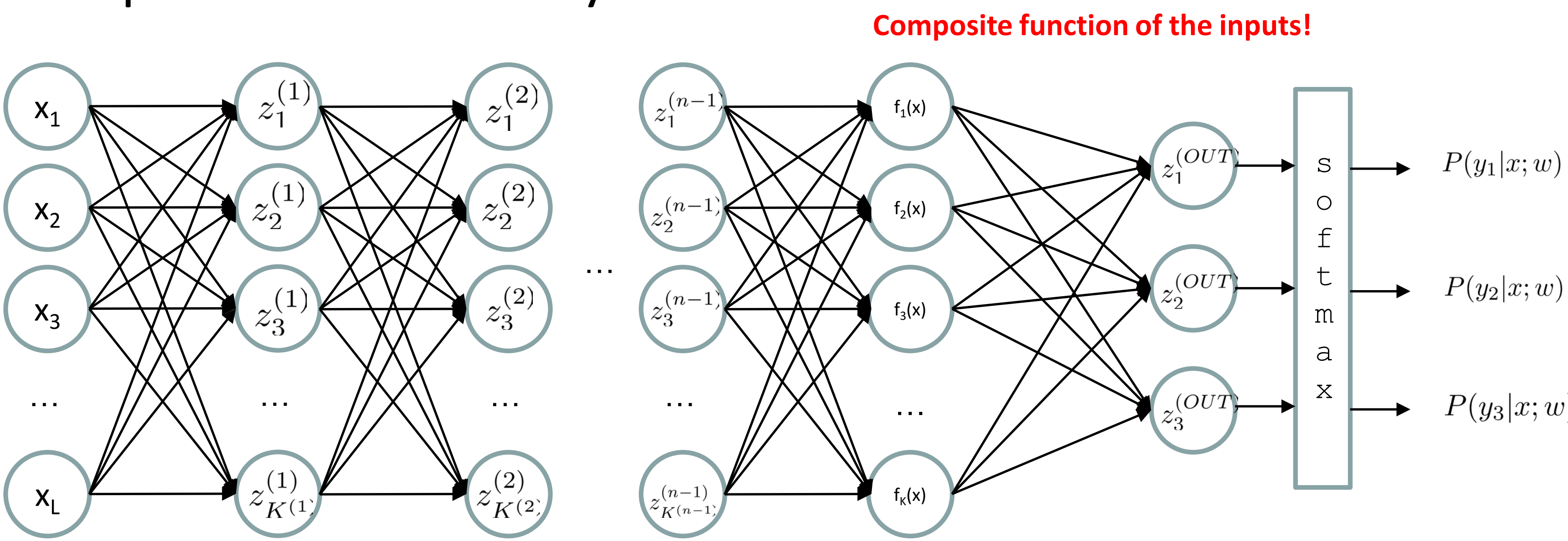
Input layer Hidden layer Output layer

$in\_z_i^{(k)}$ is the input of neuron i in layer k (after input function)

$out\_z_i^{(k)}$ is the output of neuron i in layer k (after activation)

Deep Neural Network = Also learn the features!

- Deep NN = at least 2 layers



$$out\_z_i^{(k)} = g\left(in\_z_i^{(k)}\right) = g\left(\sum_j w_{i,j}^{(k-1,k)} out\_z_j^{(k-1)}\right)$$

g = nonlinear activation function

Dan Klein and Pieter Abbeel

Why non-linear activation functions?

- If no activation function, output of a neuron is:

$$\Sigma_i \, w_{i,j} \, x_i + b_j$$

→ Linear classifier (whatever number of neurons and layers)

- Activation functions introduce the non-linearity needed to model non-linear patterns.

What characteristics do we want for the activation function?

- **Vanishing gradient problem**: Not shifting the activation value towards 0.
- **Zero-centred**: Symmetrical to 0 so gradient does not go in a particular direction.
- **Computational inexpensive**: Activation function computed a lot of times (especially in large DNN!).
- **Differentiable**: To be able to calculate the gradient for the backpropagation in the gradient descent process.

Existing activation functions

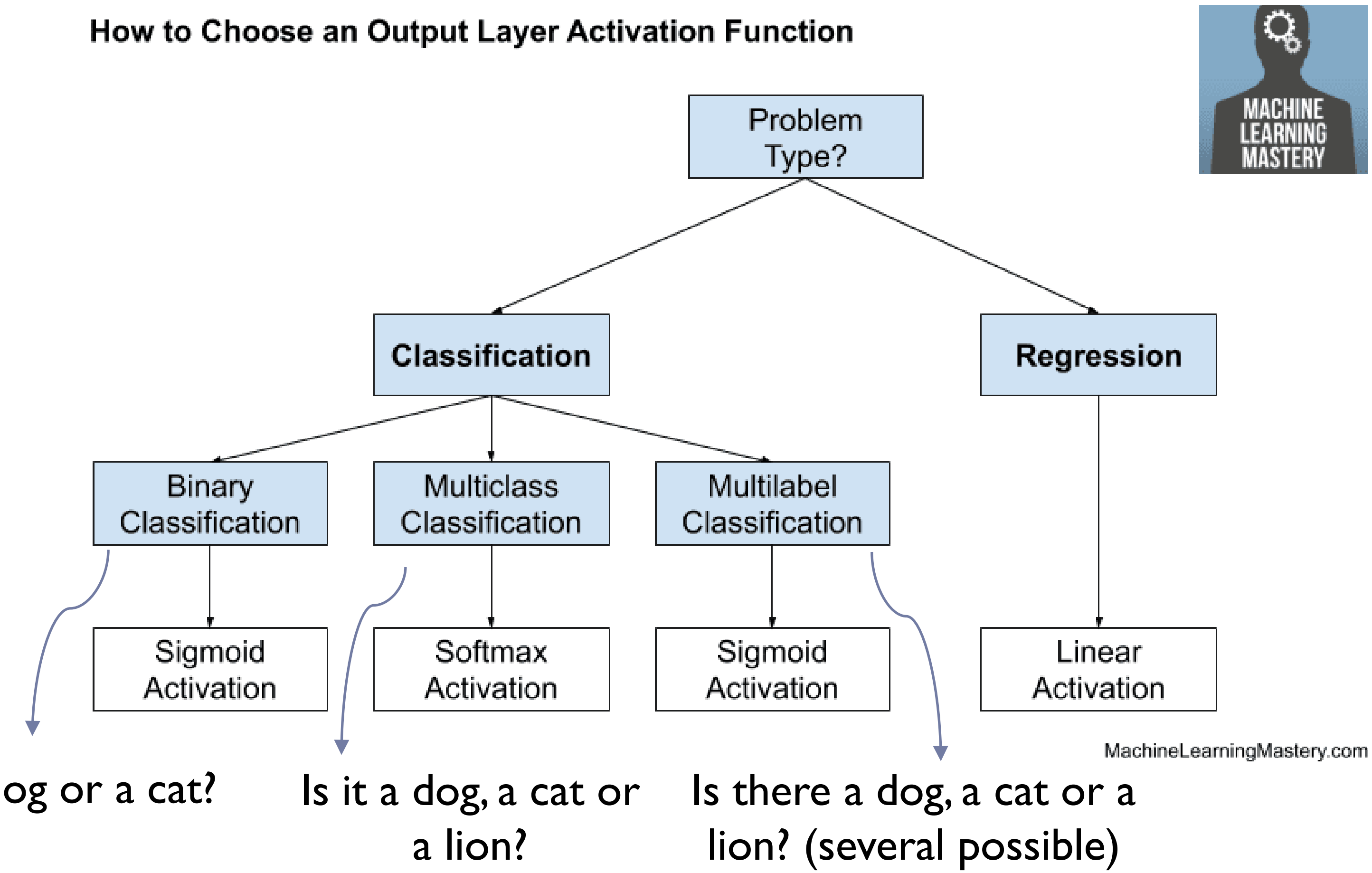
Summary of existing activation functions:

https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

In practice:

- ReLu commonly used in hidden layers.
- Sigmoid and softmax commonly used in output layer.

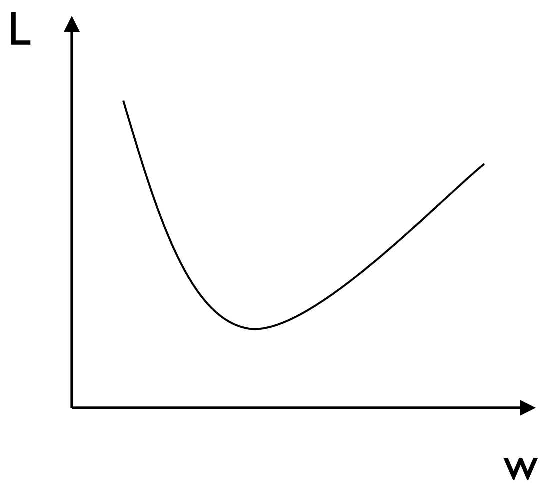
Output layer activation functions



https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=The%20output%20layer%20will%20typically,for%20a%20given%20input%20value.

Learning a function is an optimisation problem

- Optimisation of the parameters (weights and biases) to minimise a cost/loss/error function (i.e., the difference between actual value and predicted value).



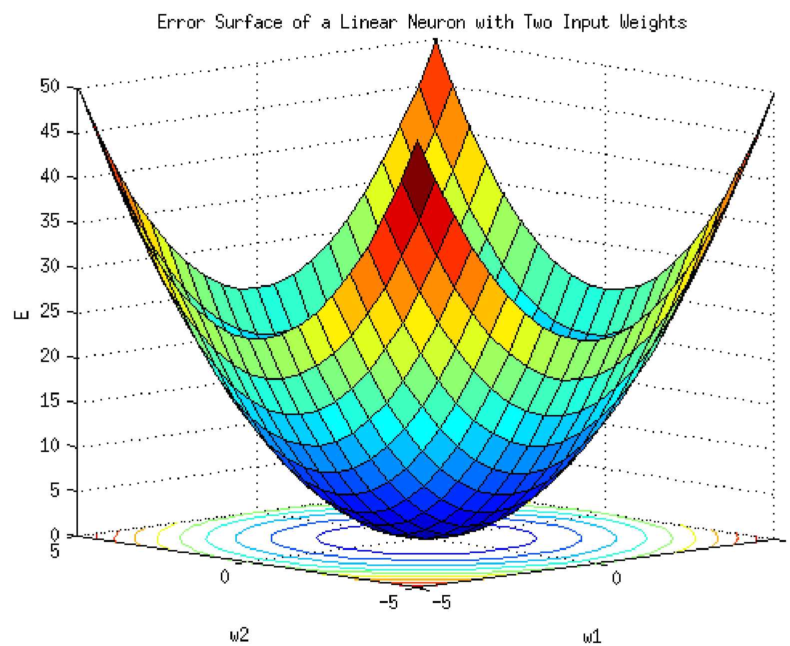


Figure: https://srinivas-yeeda.medium.com/loss-functions-in-deep-learning-models-129866be93e

Gradient Descent

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- Perform update in downhill direction for each coordinate.
- The steeper the slope (i.e. the higher the derivative) the bigger the step for that coordinate.
- E.g., consider: $g(w_1, w_2)$
  - Updates:

$$w_1 \leftarrow w_1 - \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 - \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

- Updates in vector notation:

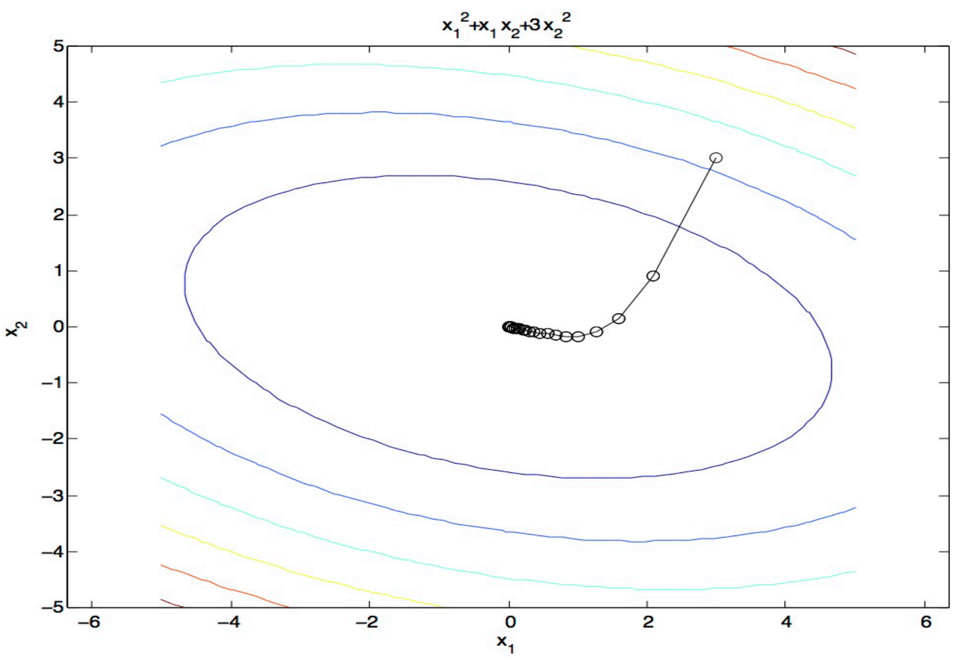
$$w \leftarrow w - \alpha * \nabla_w g(w)$$

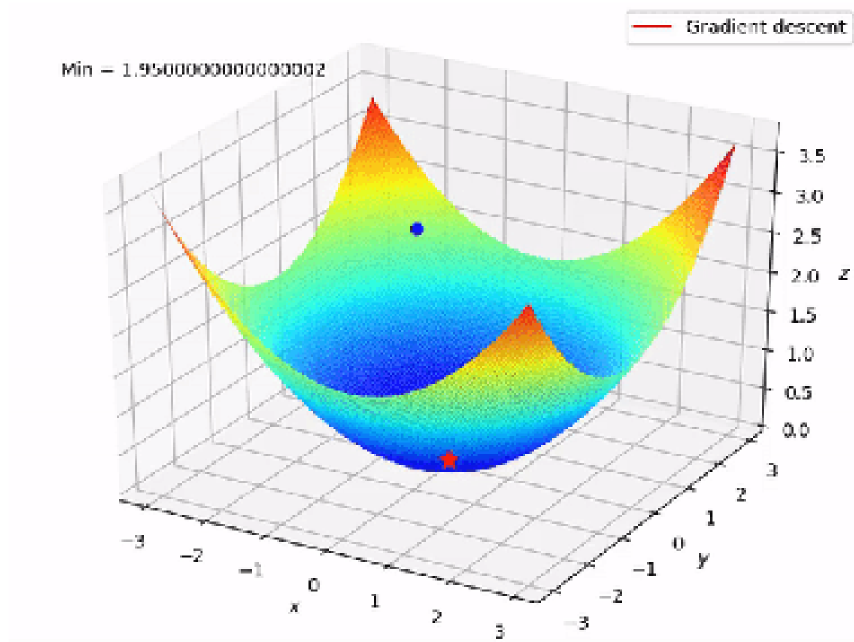
with: $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$ **= gradient**

Steepest Descent

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- Idea:
  - Start somewhere
  - Repeat: Take a step in the steepest descent direction





Figures source: Mathworks and https://suniljangirblog.wordpress.com/2018/12/03/the-outline-of-gradient-descent/

Steepest Direction

- Steepest Direction = direction of the gradient

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \cdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Dan Klein and Pieter Abbeel

Optimization Procedure: Gradient Descent

- init w
- for iter = 1, 2, …

$$w \leftarrow w - \alpha * \nabla g(w)$$

- α : learning rate --- hyperparameter that needs to be chosen carefully

- If too high → chance to miss the optimum
- If too low → very long time

Gradient Descent / Stochastic GD, Mini-batch GD

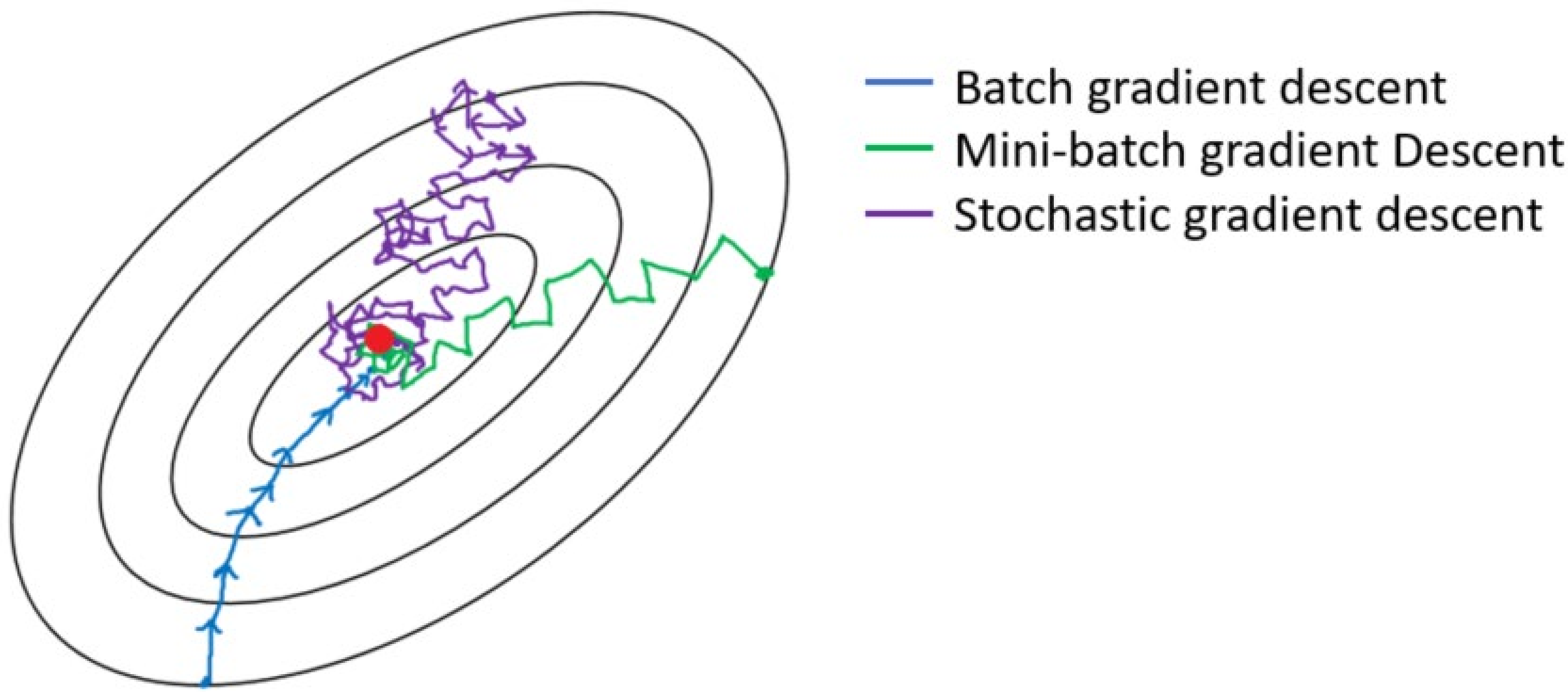
- Gradient Descent (GD)/ Batch GD: updates weights based on loss over the full training data (batch size = dataset size)
  - → Smooth descent as using average gradient over training set (true gradient)
  - → Can be used for small datasets (max 2000 instances)

- Mini-batch GD: updates weights based on loss over a randomly chosen subset of data (>1 datapoint / iteration)
  - → More efficient than GD but introduce fluctuations
  - → Typical batch sizes: 64, 128, 256, 512, … (fit CPU/GPU memory)

- Stochastic GD (SGD): updates weights based on loss over a randomly chosen instance of data (1 datapoint / iteration)
  - → More efficient than GD but fluctuates a lot

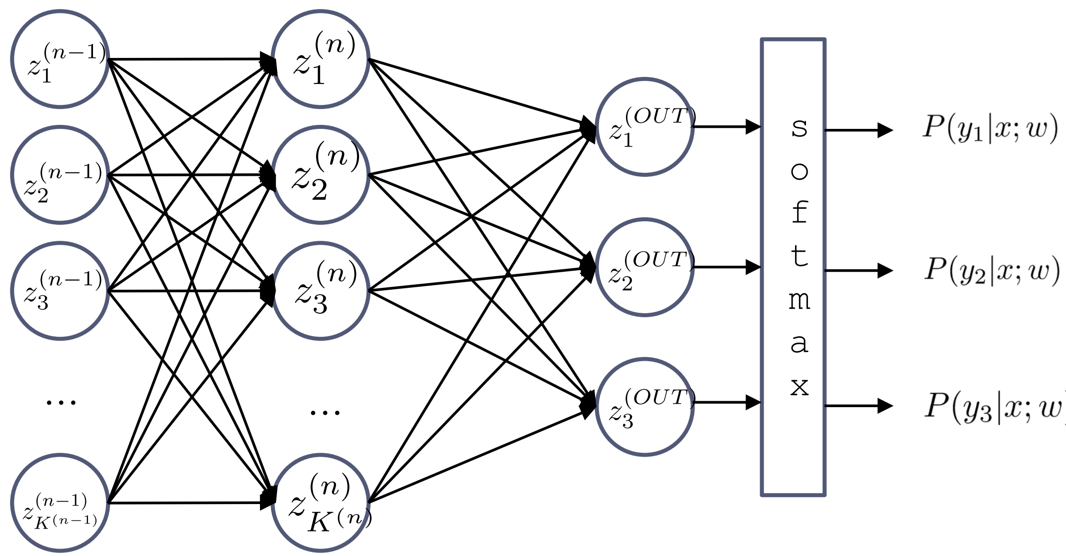
https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Gradient Descent / Stochastic GD, Mini-batch GD



https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Training a Network



Keywords:
- Forward pass
- Backward pass
- Gradient
- Backpropagation

Dan Klein and Pieter Abbeel

Exploding and vanishing gradients

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- Gradients are calculated in the backpropagation process to update the weights in the desired direction.

- **Vanishing gradients:**
  - Gradients become smaller and smaller and can become close to 0.
  - Can slow down or stop the learning process (very small weights' update).

$out\_z_j = g(\Sigma_i \, w_{i,j} \, x_i + b_j)$

$out\_z_j' = g'(\Sigma_i \, w_{i,j} \, x_i + b_j) * x_i$ (chain rule)

If g'() is close to 0, then the value of the gradient becomes smaller and smaller as backpropagation processes back to the initial layers (significant for large NN).

Exploding and vanishing gradients

- Gradients are calculated in the backpropagation process to update the weights in the desired direction.

- **Vanishing gradients:**
  - Gradients become smaller and smaller and can become close to 0.
  - Can slow down or stop the learning process (very small weights' update).

- **Exploding gradients:**
  - Gradients become larger and larger as backpropagation progresses.
  - Learning can become unstable (large weights update) and diverge.

How to improve learning? Learning rate

$$w \leftarrow w - \alpha * \nabla_w g(w)$$

- "One hyperparameter to rule them all."
- Often seen as the most important hyperparameter to tune.

- Hard to know in advance what will be the best value.
- Trial/error (common range: $10^{-6} < \alpha < 1.0$)

- Grid/random search, sensitivity analysis, optimisation technics.

https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Learning rate

- **Learning rate decay schedule**
  - Common practice: decrease learning rate over time (learning rate decay).
  - E.g., linear decay.
  - Linear decay for set number of iterations and then constant.

- **Adaptive learning rates strategies**
  - Monitors the model's performance and adapt the learning rate in response.
  - Reduces learning rate when performance plateaus.
  - Increases learning rate when performance does not improve for a number of iterations.

https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

Learning rate decay schedule

- A common practice: decaying learning rate as learning progresses.

- Why?
  - → Cost function less steep as you come close to an optimum.
  - → Decaying the learning rate allows to take smaller steps when approaching the optimum.

- Finding the right decay schedule is non-trivial.
  - Time-based: Linear decay, exponential decay?
  - Step-based: how much to drop every how many epochs?

https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1
https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

Algorithms with adaptive learning rate

- SGD uses a fix learning rate value.
- Other gradient optimisation algorithms implements adaptive learning rates.
- Adagrad, RMSProp, AdaDelta, ADAM.
- Adaptive learning rate strategies generally outperform fixed and not well tuned learning rates.

https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827

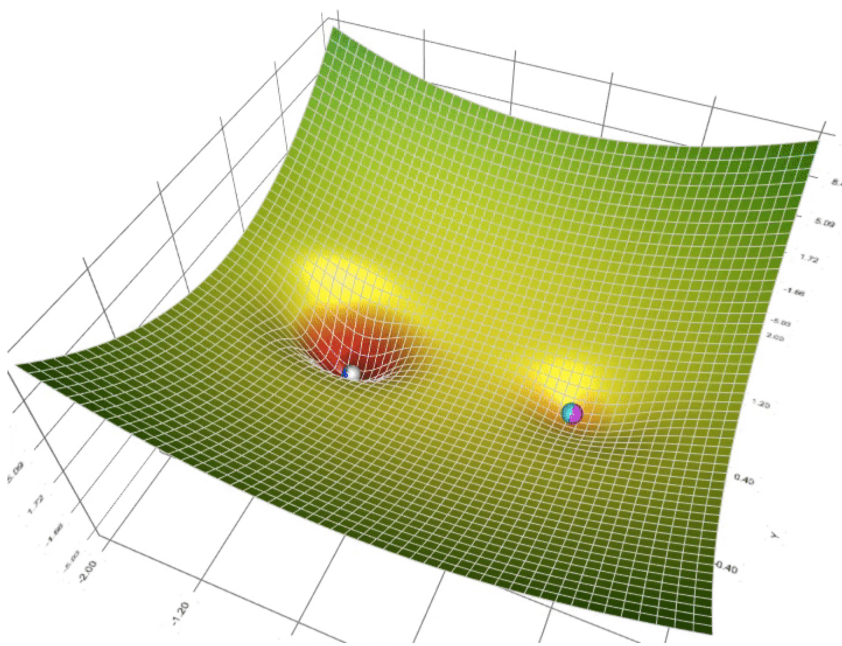
https://ruder.io/optimizing-gradient-descent/

Deep Learning, 8.5, p.298

Visualisation tool

- A visual explanation of optimisation algorithms with a visualisation tool:

https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c



Optimisation tutorial

- To better understand considerations around initialisation.

- Tutorial about NN optimisation:

https://www.deeplearning.ai/ai-notes/optimization/

(They use linear algebra notations)

Notations can be found here:

https://cs230.stanford.edu/files/Notation.pdf

Katanforoosh & Kunin, "Parameter optimization in neural networks", deeplearning.ai, 2019.

Empirical and iterative process

Lots of choices to make!

- Evaluation:

Cost function

Evaluation strategy

- NN hyperparameters:

Number of layers

Number of neurons per layer

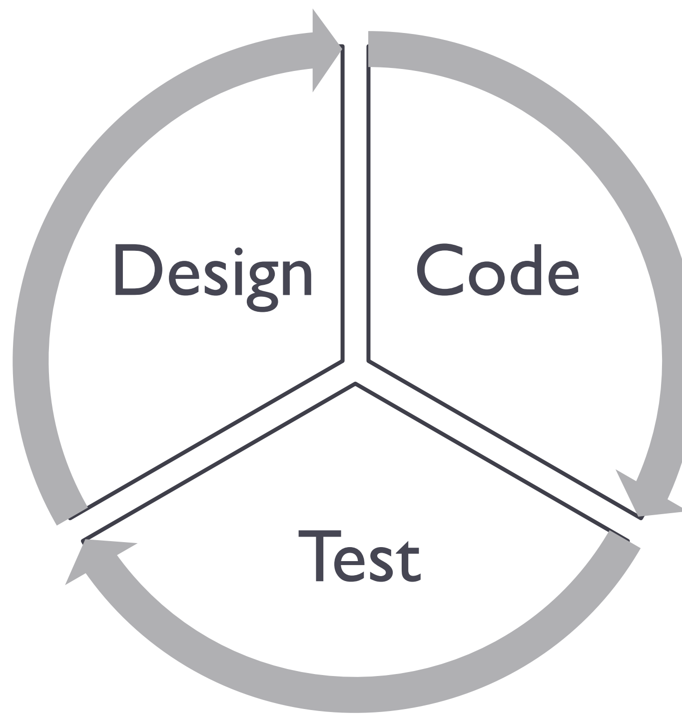
Activation functions

Learning rate

Weight initialisation

Mini-batch size

…



→ Eventually dealt with by AutoML with neural architecture search (NAS)?

https://dl.acm.org/doi/pdf/10.1145/3447582

Evaluation strategy

- Train/dev/test sets

- If small dataset (100, 1000, 10 000 samples)
- ↳ 60%/20%/20%
- If large dataset (> 1 000 000)
- ↳ 98%/1%/1%

- Training set and dev/test set usually need to come from same distribution (but it is ok if it varies a bit when gathering a lot of training data).
- Make sure dev and test sets come from the same distribution.

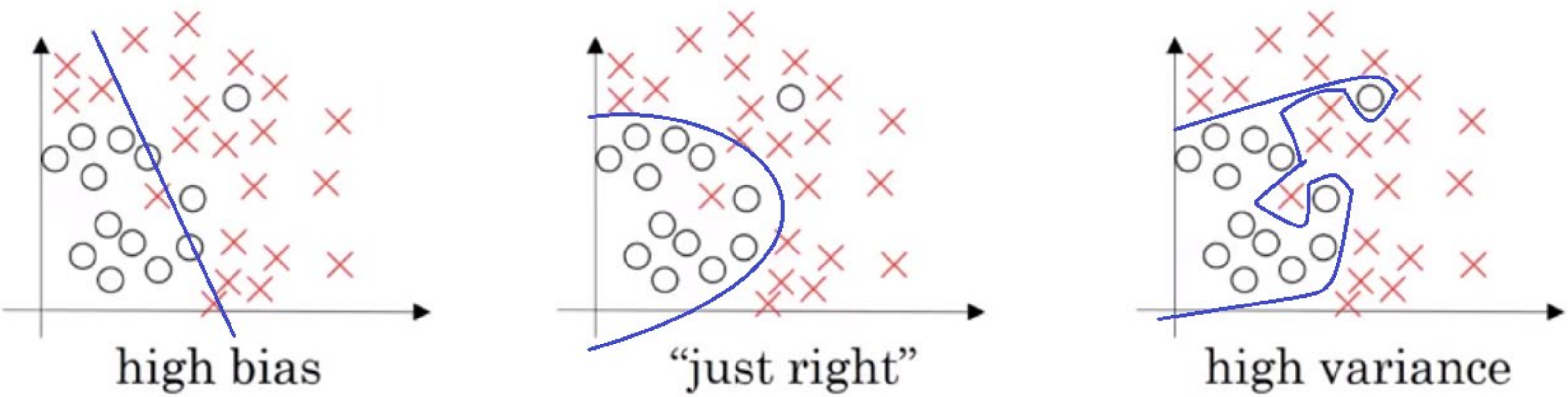
https://snji-khjuria.medium.com/everything-you-need-to-know-about-train-dev-test-split-what-how-and-why-6ca17ea6f35

How does you model do?

UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

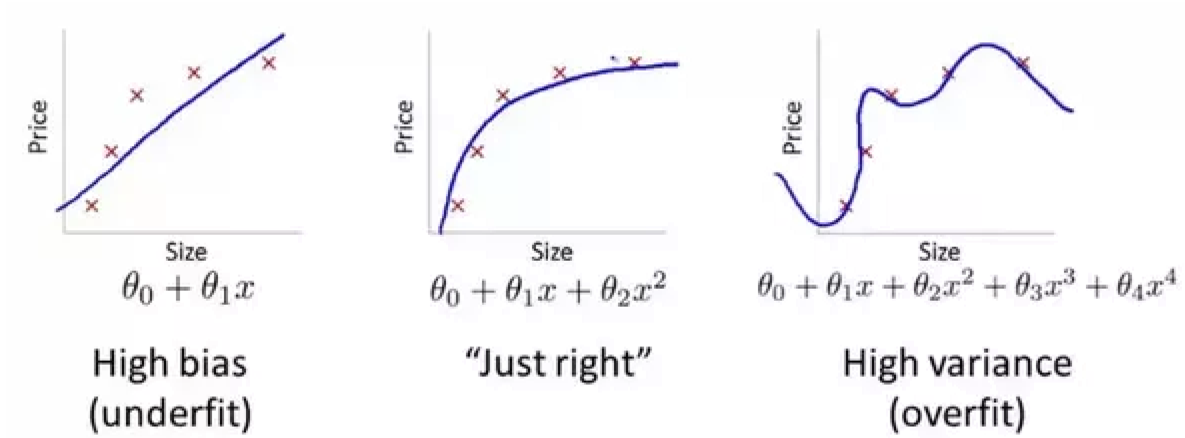
SCIENCE

- Bias vs variance / underfitting vs overfitting

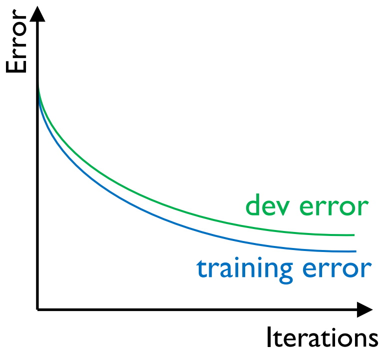


Underfit

Overfit

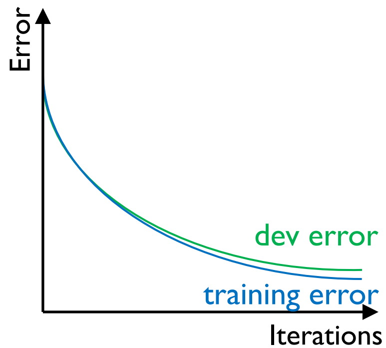


How does you model do?



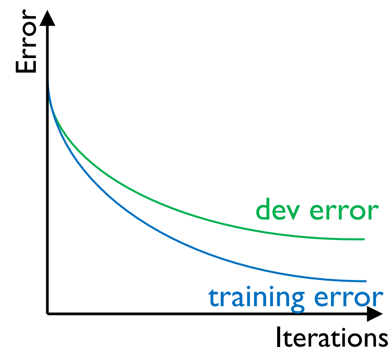
training error = 15%
dev error = 16%

High bias
Underfitting



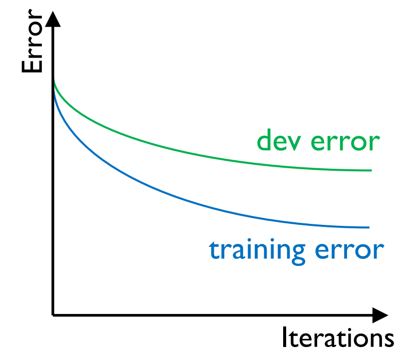
training error = 3%
dev error = 4%

Low bias
Low variance
Good performance



training error = 3%
dev error = 15%

High variance
Overfitting



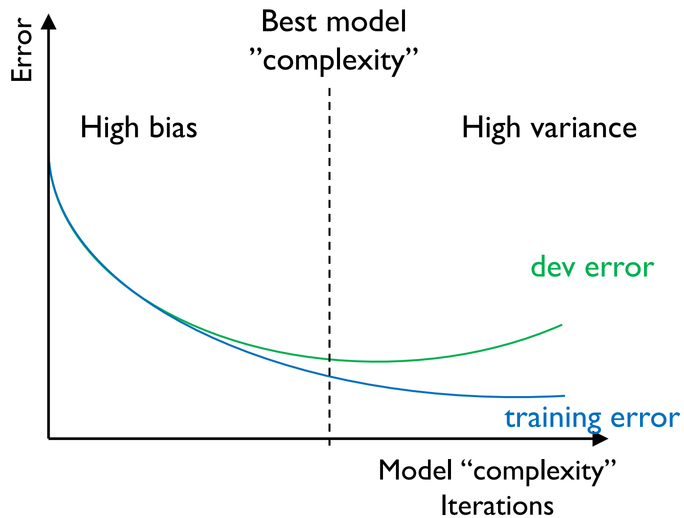
training error = 15%
dev error = 35%

High bias
High variance

https://towardsdatascience.com/how-to-efficiently-design-machine-learning-system-caef9e05d5f

How to improve learning? Overfitting

- Very common problem with Deep Learning: overfitting



- Regularisation = discouraging learning a more complex model
  - Reduces the variance, but increases the bias

Regularisation

“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

Deep learning, 5.2.2, p.117

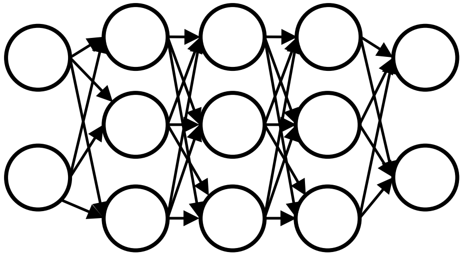
- Main regularisation technics :
  - L1 & L2 regularisation
  - Dropout
  - Early stopping
  - Data augmentation
  - Batch normalisation (not a regularisation technic originally, but has some regularisation effect)

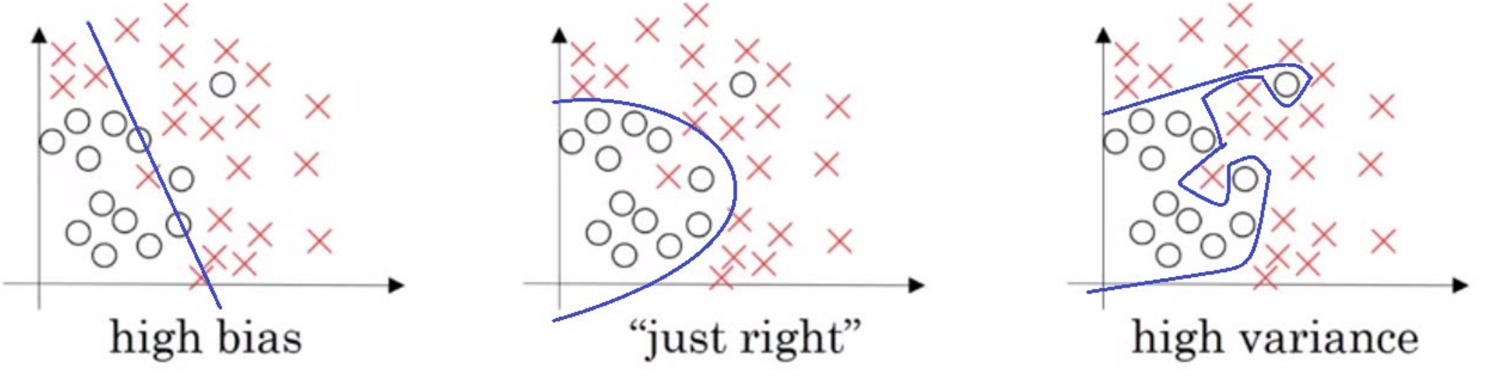
https://theaisummer.com/regularization/

How does regularisation help to avoid overfitting?

First intuition:

- E.g., L1 & L2 regularisation penalises large weights values.
- Keeping weights close to zero for some neurons.

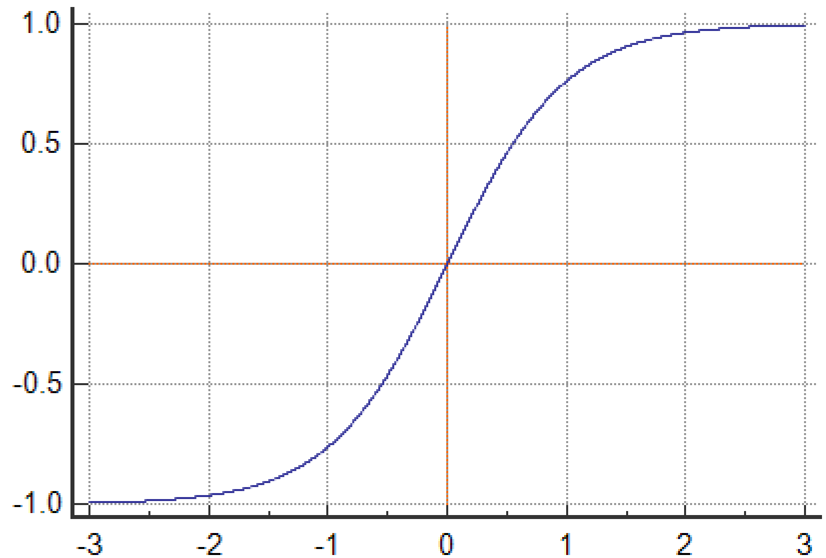




How does regularisation help to avoid overfitting?

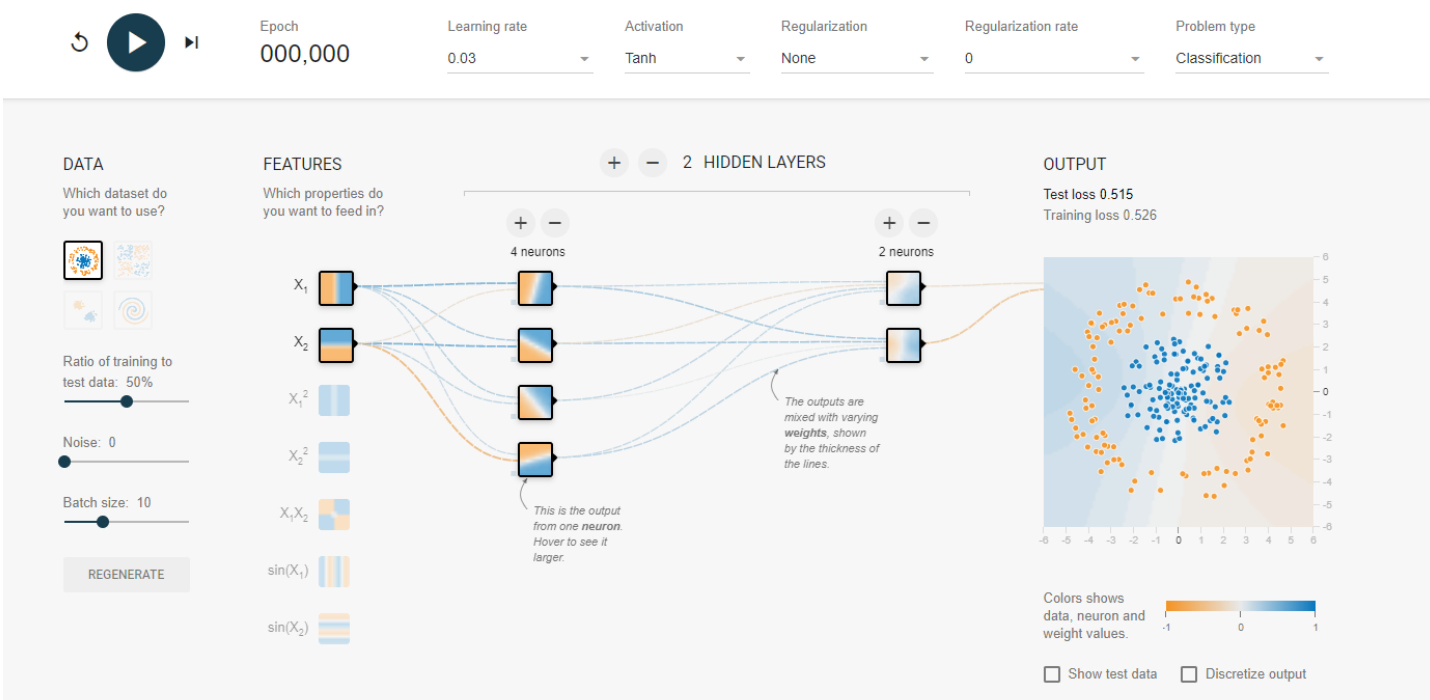
Second intuition:

- Limiting the weights values will bring the output of a neuron in the linear zone of activation function (e.g. tanh).
- It will limit the NN power to model non-linearities.



Tensorflow playground platform

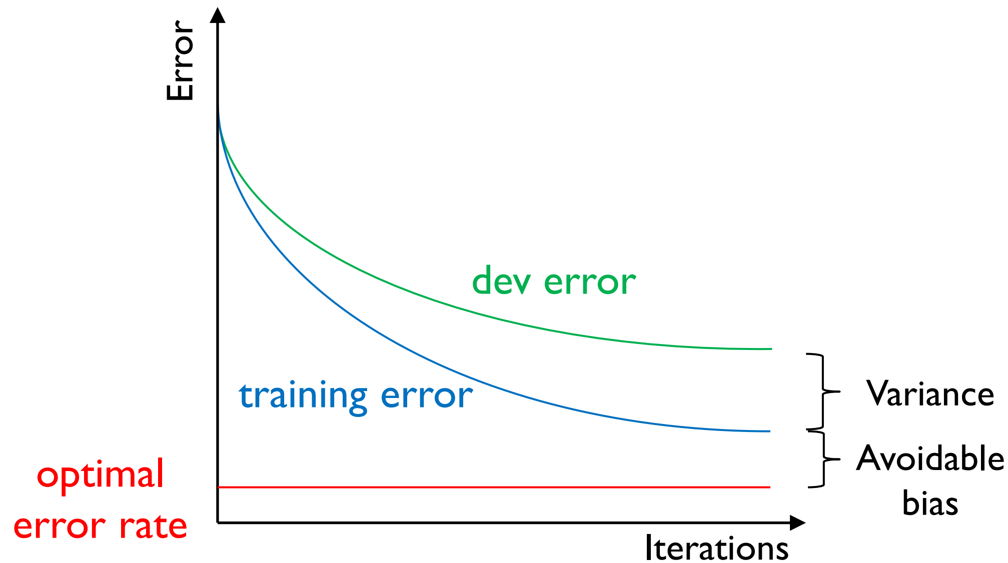
- Interactive platform to test and visualise the effects of varying hyperparameters: https://playground.tensorflow.org/



Optimal error rate / avoidable bias

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- Optimal error rate
  - Error rate of an optimal classifier (e.g., human performance)
  - Can be hard to estimate
- Avoidable bias
  - Training error – optimal error rate
- Variance
  - Dev error – training error



Andrew Ng, "Machine Learning Yearning", Chap. 22.

Simplest formula to address variance/bias issues

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- High avoidable bias
  - Intuition: model not complex enough to map inputs and outputs.
  - Simple fix: Increase model size (e.g., increase layers or neurons per layer).
  - Might increase variance and risk of overfitting (if no regularisation).
  - Will slow the learning.

- High variance
  - Intuition: training data not sufficient to generalise on dev data.
  - Simple fix: Add data to the training set.
  - More data might not be available.
  - Try data augmentation.

Andrew Ng, "Machine Learning Yearning", Section 23.

Bias vs variance tradeoff

- Some choices reduce bias but increase variance.
  - E.g., increasing size of the network.

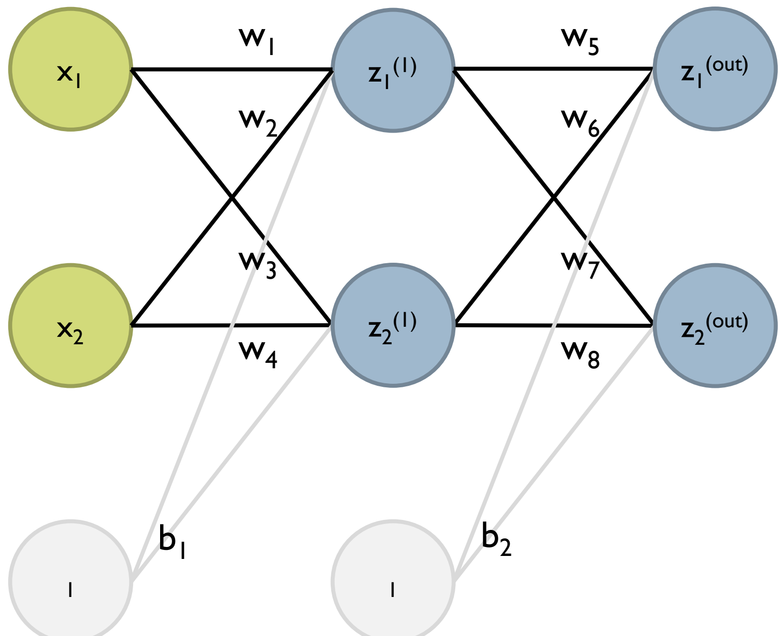
- Some choices reduce variance but increase bias.
  - E.g., adding regularisation (early stopping might stop learning before reaching low bias, penalizing high weights might prevent the model to reach low bias, etc).
  - Effect of regularisation on bias can be reduced with a good hyperparameter tuning.
  - Data augmentation does not increase bias if relevant augmentation.

- More useful advice in Sections 25 to 27 of Andrew Ng's "Machine Learning Yearning" book, to reduce variance and bias.

Andrew Ng, "Machine Learning Yearning", Sections 24 to 27.

Parameters vs Hyperparameters

- What are the parameters of a NN?



→ The weights (w_1, w_2, …) and the biases (b_1, b_2, …).

Parameters vs Hyperparameters

- What are the hyperparameters of a NN?

  - Learning rate (α)
  - Number of hidden layers
  - Number of neurons per layer
  - Number of iterations
  - Choice of the activation function
  - Choice of the optimisation method
  - Choice of regularisation
  - …

- They control the training process but are external to the model.
- They need to be tuned manually or automatically (e.g., meta-learning).

Next Lectures

Lectures 2 & 3: Learning with sequences (RNNs, Transformers, LLMs)

- Understand how recurrent neural networks work.
- Recognise commonly used neural network architectures based on RNN (LSTM, GRU).
- Understand how transformers work.
- Understand the principles of Large Language models.