# Deep Learning

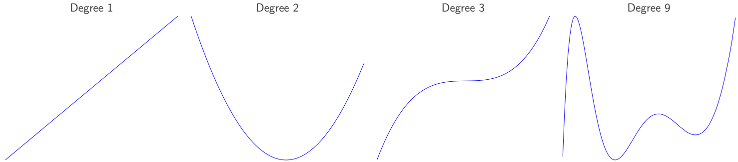
## Regularization in Neural Networks

Syed Irtaza Muzaffar

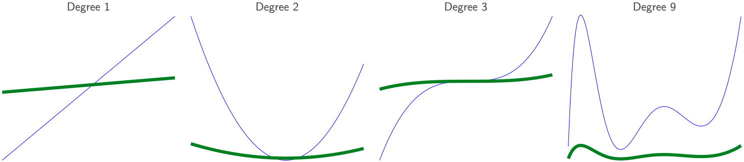
# Before we start

## *A primer on ML*

1. Capabilities of polynomials (lines, quadratics, cubics, . . . , degree $M$).




2. Capability can be reduced by restricting coefficients.



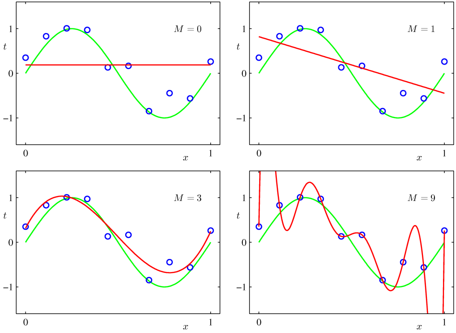

green curve = blue curve divided by 10

# Before we start
*A primer on ML*

3. Everything is noisy.

$$\text{Observation} = \text{Reality} + \text{Noise}$$

4. Therefore, zero *training* error is bad. Over-fitting vs generalisation.




5. Over-fitting can be reduced via regularization.

# Before we start

*A primer on ML*

3. Everything is noisy.

$$\text{Observation} = \text{Reality} + \text{Noise}$$

4. Therefore, zero *training* error is bad. Over-fitting vs generalisation.

$y = \sin(x) + \epsilon$

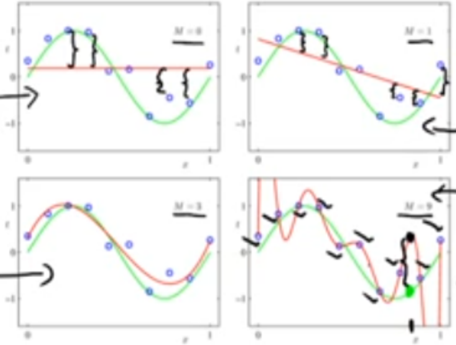
Training Data $\begin{cases} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_N, y_N \end{cases}$

$L = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2$

Under fitting

High training error

Good fitting



Underfitting

zero training error

High testing error

5. $\boxed{\text{Over-fitting}}$ can be reduced via regularization.

# Weight Penalties

- Similar to polynomials, networks with large weights are more powerful.
- Therefore, more prone to overfitting.
- So penalise magnitudes of weights to restrict capability.

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- *Hyperparameter*[1] $\lambda$ controls the level of overfitting.
- Alternative: separately penalise each layer

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \sum_{l=1}^{L} \frac{\lambda_l}{2}\|\mathbf{w}^{(l)}\|^2$$

Not used often due to increased number of hyperparameters.

[1] Something that is not a parameter but influences what the parameters will be.

# Weight Penalties

- Similar to polynomials, networks with large weights are more powerful.
- Therefore, more prone to overfitting.
- So penalise magnitudes of weights to restrict capability.

$\vec{w}^* = \arg\min_{\vec{w}} \tilde{L}(\vec{w})$

$$\tilde{L}(\mathbf{w}) = \underbrace{L(\mathbf{w})}_{y \leftrightarrow t} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{weights penalty}}$$

$\frac{\lambda}{2}(w_1^2 + w_2^2 + \dots + w_k^2)$

$\lambda > 0$

- *Hyperparameter*[1] $\lambda$ controls the level of overfitting.
- Alternative: separately penalise each layer

$\tilde{L} = L(w) + \frac{\lambda}{2}\|w\|^2$

$\tilde{L} = \underbrace{\frac{1}{2} SSE}_{1-100} + \underbrace{\frac{\lambda}{2}\|w\|^2}_{0-10000}$

$\lambda$ controls the tradeoff between fitting and regularization.

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \sum_{l=1}^{L} \frac{\lambda_l}{2}\|\mathbf{w}^{(l)}\|^2$$

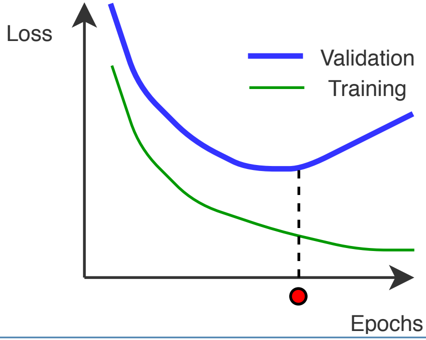
Not used often due to increased number of hyperparameters.

- If $\lambda = 0$, model can possibly overfit.
- If $\lambda \to \infty$, model will move towards underfitting.

[1] Something that is not a parameter but influences what the parameters will be.

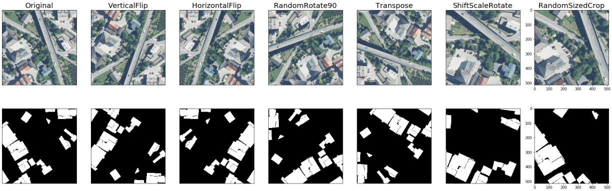
# Early Stopping

- Split some part of the training set into a validation set that will not be used for training.
- During training, record loss on training as well as validation set.
- When validation loss starts increasing while training loss is still going down, the model has started overfitting.
- So stop training at that point.



# Data Augmentation

- Augment training set with transformed versions of training samples.
- Domain specific data augmentations
  - Images: Color, Geometry
  - Text: Synonyms, Tense, Order
  - Speech: Speed, Sound effects


https://github.com/albumentations-team/albumentations

# Data Augmentation

https://github.com/aleju/imgaug

# Label Smoothing

- Training adjusts the model to make outputs as close as possible to the targets/labels.
- So if labels are smoothed a little, overfitting will be reduced.
- For example, if label 0 is mapped to 0.1 and 1 is mapped to 0.9, training will converge early.
- Training procedure will not try as hard as before to output as close as possible to 0 or 1.

# Summary

- All data contains noise.
- Given enough power, a neural network will model noise as well.
- Restricting the network's power allows it to model the underlying behaviour of data instead of noise.
- This reduces over-fitting on training data and improves generalisation of the network on unseen data.