06-20416 and 06-12412
(Intro to) Neural Computation

11 – Regularisation

Per Kristian Lehre

Last lecture

- Universal Approximation Theorem
- Evolution and Learning (guest lecture)

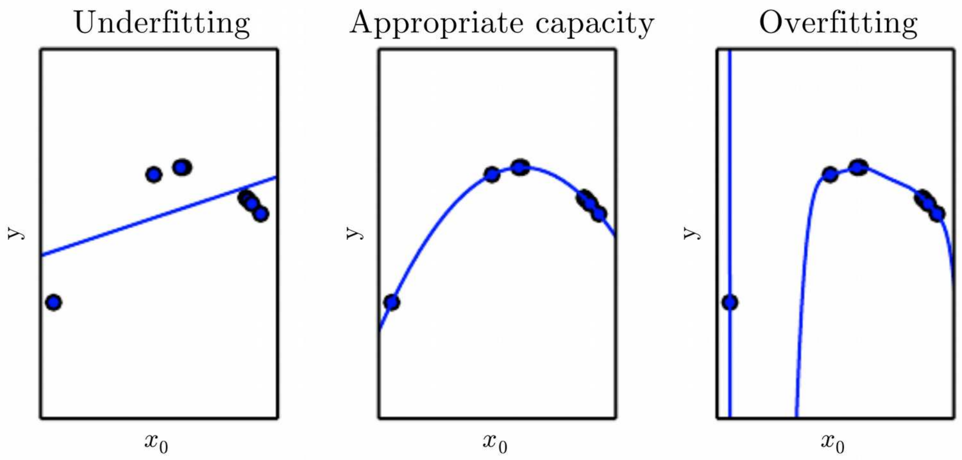
Outline

- Model capacity
- Underfitting, overfitting
- Regularisation techniques
  - Data augmentation
  - Early stopping
  - Parameter norm penalties
  - Dropout

Model Capacity

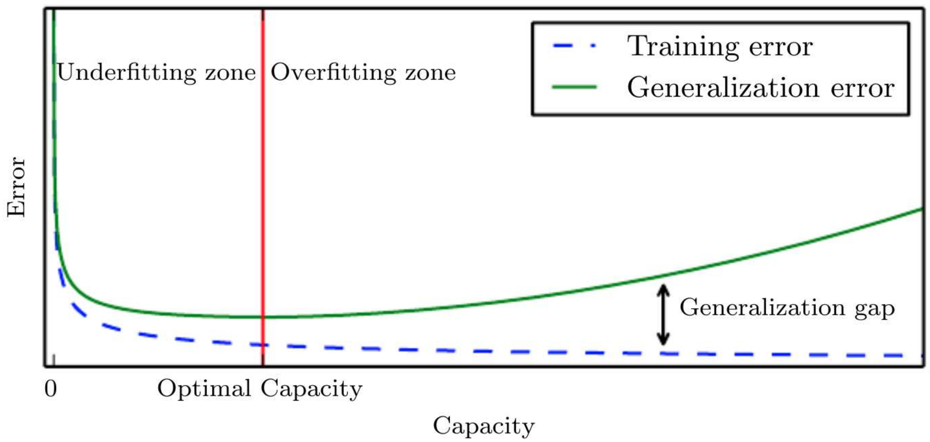
- Informally, a model's capacity to fit a wide variety of functions.
- In statistical learning theory, model capacity is quantified by **VC-dimension**: largest training set for which the model can classify the labels arbitrary into two classes
- By the universal approximation theorem, neural networks can have very high capacity.

Underfitting, Overfitting



- Underfitting
  - Too high training error
- Overfitting
  - Too large gap between training error and test error

Model Capacity vs Error



- Training and test error behave differently
- Training error often decreases with capacity
- Test error can increase beyond a certain capacity
- Capacity is optimal when model matches data generating process

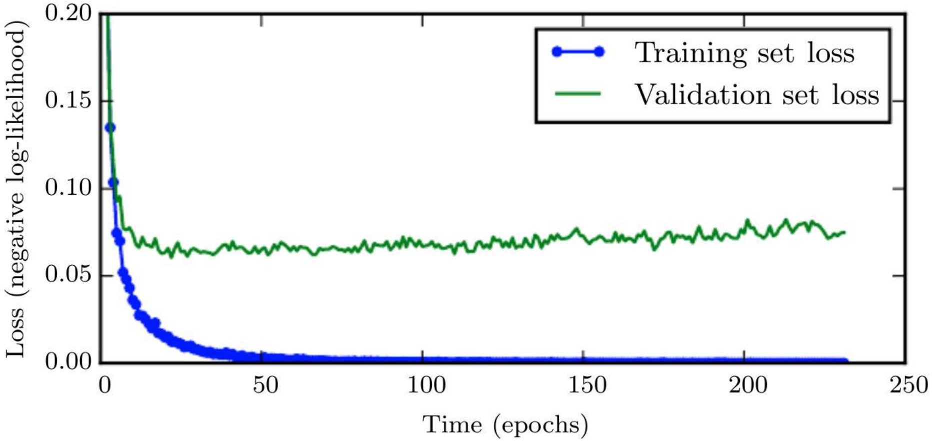
Regularisation

- Three model regimes
  1) Model family excludes data-generating process (underfitting)
  2) Model family matches data-generating process
  3) Model family matches data-generating process, and possibly many other models (possible overfitting)
- Regularisation attempts to move a model from regime 3 to regime 2

Regularisation via data augmentation

- Many data sets can be augmented via transformations
- E.g, a data set for image classification can be augmented via image transformations
  - mirroring
  - translation
  - scaling
  - rotation
  - noise

Regularisation by Early Stopping



- Split data into a training, a validation, and a test set
- Train model on training set, evaluate with fixed intervals on validation set
- Stop training when validation error has increased
- Return model parameters when validation loss was the lowest, rather than the latest parameters

Parameter Norm Penalties

Replace cost function by

$$\tilde{C}(\theta; X, y) = C(\theta; X, y) + \alpha \Omega(\theta),$$

where

C original cost function

θ model parameters

X, y training data

Ω is a regulizer, i.e.,
a function which penalises complex models

α hyperparameter controlling degree
of regularisation

L^2 parameter regularisation

Assuming parameters $\theta = (w, b)$ (i.e., weights and biases)

$$\Omega(\theta) := \frac{1}{2} \| w \|_2^2$$

Penalises large weights

Ensemble Methods

Combining different models often reduces generalisation error.

Idea

Train k neural networks on k subsets of the training data.
Output the average (or majority) of the networks.

Disadvantages
- usually requires more training data
- k times increase in training time (if sequential training)
- only feasible for small k.

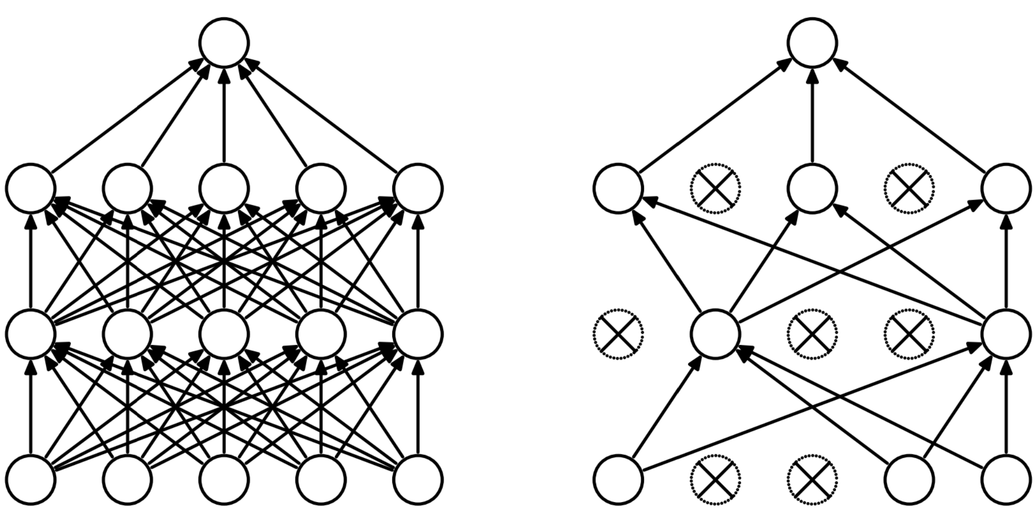
Idea 2: Dropout

In each mini-batch, "deactivate" some randomly selected activation units (not in output layer).

Each selection of units corresponds to a subnetwork. With n input and hidden layer activation units, there are 2^n subnetworks.

The subnetworks share the weights.

No dropout during testing, i.e. implicit average output from all subnetworks.



(a) Standard Neural Net (b) After applying dropout.

Implementation of Dropout

Replace each activation unit $a_j^\ell = \phi(z_j^\ell)$ in a hidden layer with a "dropout activation" unit

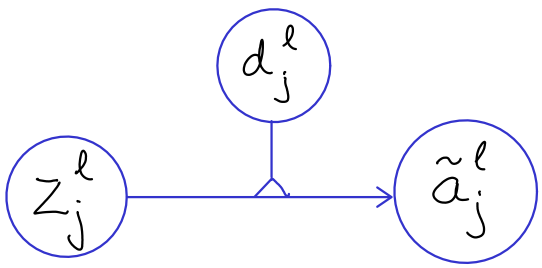
$$\tilde{a}_j^\ell = \frac{1}{1-p} \cdot d_j^\ell \cdot \phi(z_j^\ell)$$

where

$$d_j^\ell \sim \text{Bernoulli}(1-p).$$

Remark

d_j^ℓ is 0 with probability p, and 1 otherwise.



Where does the factor $\frac{1}{1-p}$ come from?

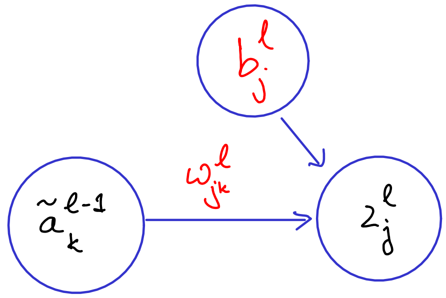
Note that $\tilde{a}_j^\ell$ is a random variable with expectation

$$\mathbb{E}[\tilde{a}_j^\ell] = p \cdot \frac{1}{1-p} \cdot 0 \cdot \phi(z_j^\ell) + (1-p) \cdot \frac{1}{1-p} \cdot 1 \cdot \phi(z_j)$$

$$= \phi(z_j^\ell) = a_j^\ell$$

Hence, choosing the factor $\frac{1}{1-p}$ makes the expected activation identical to the activation without dropout.

Backpropagation with Dropout



Given local gradient δ_j^ℓ, partial derivates are

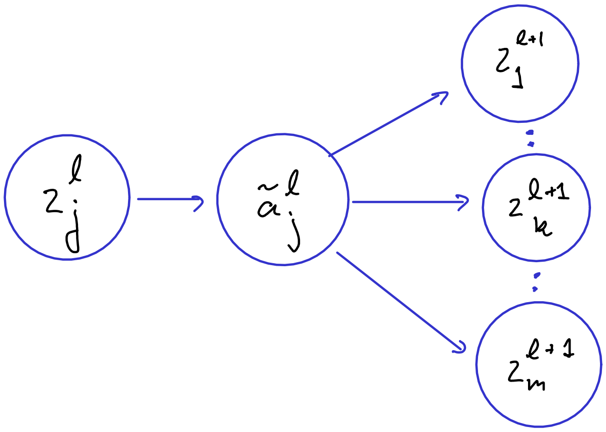
$$\frac{\partial C}{\partial w_{jk}^\ell} = \frac{\partial C}{\partial z_j^\ell} \cdot \frac{\partial z_j^\ell}{\partial w_{jk}^\ell}$$

$$= \delta_j^\ell \cdot \tilde{a}_k^{\ell-1}$$

$$\frac{\partial C}{\partial b_j^\ell} = \frac{\partial C}{\partial z_j^\ell} \cdot \frac{\partial z_j^\ell}{\partial b_j^\ell}$$

$$= \delta_j^\ell$$

Local Gradient for Hidden Layer



$$\tilde{a}_j^\ell = \frac{1}{1-p} \cdot d_j^\ell \cdot \phi(z_j^\ell)$$

$$\delta_j^\ell = \frac{\partial C}{\partial z_j^\ell} \qquad \text{by def.}$$

$$= \frac{\partial C}{\partial \tilde{a}_j^\ell} \cdot \frac{\partial \tilde{a}_j^\ell}{\partial z_j^\ell} \qquad \text{chain rule}$$

$$= \left(\sum_{k=1}^{m} \frac{\partial C}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial \tilde{a}_j^\ell} \right) \cdot \left(\frac{1}{1-p} \right) d_j^\ell \cdot \phi'(z_j^\ell) \qquad \text{chain rule}$$

$$= \left(\sum_{k=1}^{m} \delta_k^{\ell+1} \cdot w_{kj}^{\ell+1} \right) \cdot \left(\frac{1}{1-p} \right) \cdot d_j^\ell \cdot \phi'(z_j^\ell) \qquad \text{by def. of } \delta_j^{\ell+1}$$

Matrix Form

$$\delta^\ell = \left(\frac{1}{1-p} \right) \left(\left(w^{\ell+1} \right)^T \delta^{\ell+1} \right) \odot d^\ell \odot \phi'(z^\ell)$$

Backpropagation Algorithm with Dropouts

Input: A training example $(x, y) \in \mathbb{R}^m \times \mathbb{R}^{m'}$

1. Set the activation in the input layer

$$d_j^1 \sim \text{Bernoulli}(p) \quad \text{for } j = 1, \dots, m$$

$$\tilde{a}^1 = \left(\frac{1}{1-p}\right) \cdot d^1 \odot x$$

2. for each $\ell = 2$ to $L-1$ feed forward

$$d_j^\ell \sim \text{Bernoulli}(p) \quad \text{for } j = 1, \dots, m$$

$$z^\ell = w^\ell \tilde{a}^{\ell-1} + b^\ell$$

$$\tilde{a}^\ell = \left(\frac{1}{1-p}\right) d^\ell \odot \phi(z^\ell)$$

3. Set activations in output layer

Remark: No dropout in output layer

$$z^L = w^L \tilde{a}^{\ell-1} + b^\ell$$

$$a^L = \phi(z^L)$$

4. compute local gradient for output layer

$$\delta^L := \nabla_a C \odot \phi'(z^L)$$

5. backpropagate local gradients for hidden layers, i.e.

for each $\ell = L-1$ to 2

$$\delta^\ell := \frac{1}{1-p}\left(\left(w^{\ell+1}\right)^T \delta^{\ell+1}\right) \odot d^\ell \odot \phi'(z^\ell)$$

6. <u>return</u> the partial derivatives

$$\frac{\partial C}{\partial w_{jk}^\ell} = \delta_j^\ell \, \tilde{a}_k^{\ell-1}$$

$$\frac{\partial C}{\partial b_j^\ell} = \delta_j^\ell$$

Summary

- Model capacity
- Underfitting, overfitting
- Regularisation techniques
  - Data augmentation
  - Early stopping
    - Choose model parameters when validation error was lowest
  - Parameter norm penalties
    - L^2-parameter regularisation
  - Dropout
    - Often used in conjunction with L^2-regularisation.
  - See also Ch 7 in Goodfellow et al. (2016)

Next week

- Convolutional Networks