Neural Computation

Summary and Example Questions (Week 1-6)

School of Computer Science, University of Birmingham

W1: Machine Learning

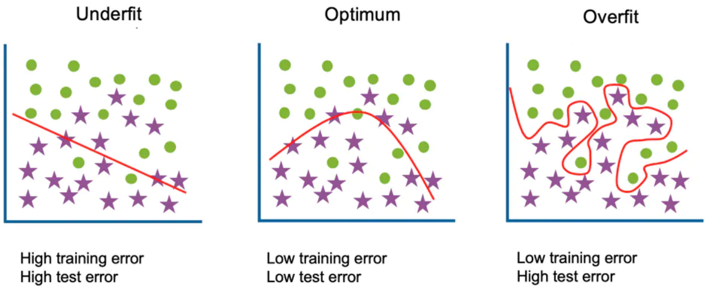
Definition by Tom Mitchell (1997)

A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- Learning task T: regression, classification, transcription, translation, synthesis and sampling, ...
- Performance measure P: depends on learning tasks, e.g., accuracy for classification
- Experience E: $S = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$, $S = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$

W1: Underfitting and Overfitting

- Loosely speaking, we say a model underfits when
  - training performance is poor
- We say a model overfits when
  - training performance is good but
  - test performance is poor



W1: Example Questions

- Categorise a given list of machine learning problem as supervised, unsupervised, or reinforcement learning
- Explain the connection between underfitting, overfitting and complexity

W2: Linear Regression

Dataset: n input/output pairs

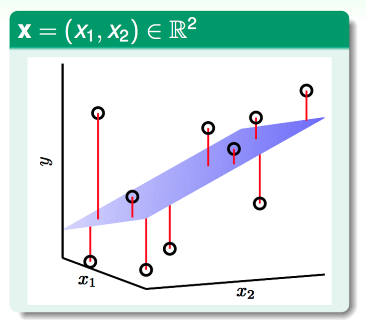
$$S = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

- $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is the "input" for the i-th data point as a feature vector with d elements.
- $y^{(i)} \in \mathbb{R}$ is the "output" for the i-th data point.

Regression Task: find a model, i.e., a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ such that $f(X) \approx Y$

Linear Model: a linear regression model has the form

$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$



W2: Linear Regression

Performance measure is of a model $\mathbf{w}$ is Mean square error (MSR)

$$C(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\big(\underbrace{y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}}_{\text{residual}}\big)^2$$

We can represent the MSR in terms of a matrix

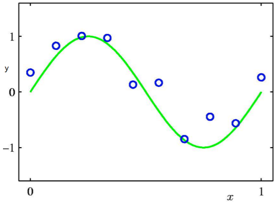
$$C(\mathbf{w}) = \frac{1}{2n}\Big(\underbrace{\mathbf{w}^\top X^\top X\mathbf{w}}_{\text{quadratic}} - 2\underbrace{\mathbf{w}^\top X^\top \mathbf{y}}_{\text{linear}} + \underbrace{\mathbf{y}^\top \mathbf{y}}_{\text{constant}}\Big) \quad X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \vdots & \\ x_1^{(n)} & x_2^{(n)} & \dots & x_d^{(n)} \end{pmatrix}$$

Closed-form solution

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} C(\mathbf{w}) \Longrightarrow \mathbf{w}^* = (X^\top X)^{-1} X^\top \mathbf{y}$$

Polynomial regression

$$f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$
$$x \mapsto \phi(x) = \left(1, x, x^2, \dots, x^M\right)^\top$$
$$f(x) = \langle \mathbf{w}, \phi(x) \rangle$$



W2: Linear Regression

The derivative of a function $f : \mathbb{R} \mapsto \mathbb{R}$ is the rate of change of f

$$f'(x) = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}.$$

Partial derivative of a **multivariate** function $f(x_1, \ldots, x_d)$ in the direction of variable x_i at $\mathbf{x} = (x_1, \ldots, x_d)$ is

$$\frac{\partial f(x_1, \ldots, x_d)}{\partial x_i} = \lim_{h \to 0} \frac{f(\ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots) - f(x_1, \ldots, x_i, \ldots, x_d)}{h}$$

Gradient of $f : \mathbb{R}^d \mapsto \mathbb{R}$

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)^\top.$$

Example

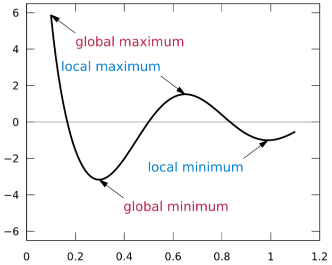
Let $f(\mathbf{w}) = \frac{1}{2}(\mathbf{w}^\top \mathbf{x} - y)^2$. Then

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{x}\mathbf{x}^\top \mathbf{w} - y\mathbf{w}^\top \mathbf{x} + \frac{1}{2}y^2 \Longrightarrow \nabla f(\mathbf{w}) = (\mathbf{w}^\top \mathbf{x} - y)\mathbf{x}.$$

W2: Linear Regression

Given an objective function $C : \mathbb{R}^d \mapsto \mathbb{R}$, we want to solve

$$\min_{\mathbf{w} \in \mathbb{R}^d} C(\mathbf{w}).$$



First-order Necessary Optimality Condition

If $\mathbf{w}^*$ is a local minimum of a differentiable function C, then

$$\nabla C(\mathbf{w}^*) = 0. \tag{1}$$

We say $\mathbf{w}^*$ satisfying Eq. (1) a stationary point.

W2: Example Questions

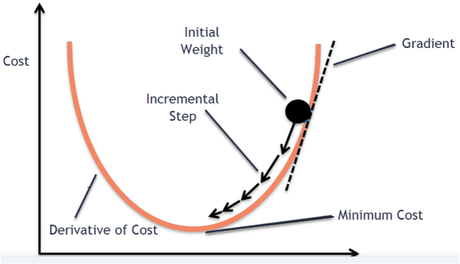
- Given a ML problem, formulate it as a linear regression problem and find the solution
- Compute gradient for simple 2D functions
- Compute a few steps of gradient descent on a simple optimization problem given starting point and learning rate
- Explain difference between local and global optima of cost functions
- Find a local optimal solution of simple cost functions via derivatives
- Explain the connection and difference between linear regression and polynomial regression

W3: Gradient Descent

Gradient Descent

Starting from $\mathbf{w}^{(0)}$ and producing a new $\mathbf{w}^{(t+1)}$ at each iteration as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla C(\mathbf{w}^{(t)}), \quad t = 0, 1, \ldots.$$



Example (Gradient Descent for Linear Regression)

The gradient of $C(\mathbf{w})$ is $\nabla C(\mathbf{w}) = \frac{1}{n}\left(X^\top X\mathbf{w} - X^\top \mathbf{y}\right)$, and then

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla C(\mathbf{w}^{(t)}) = \mathbf{w}^{(t)} - \frac{\eta}{n}\left(X^\top X\mathbf{w}^{(t)} - X^\top \mathbf{y}\right).$$

W3: Stochastic Gradient Descent

Sum Structure

$$C(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} C_i(\mathbf{w}), \quad C_i(\mathbf{w}) \text{ often corresponds to a loss with } i\text{-th example}$$

Stochastic Gradient Descent (Robbins & Monro 1951)

- Initialize the weights $\mathbf{w}^{(0)}$
- For $t = 0, 1, \ldots, T$
  - Draw i_t from $\{1, \ldots, n\}$ with equal probability
  - Compute stochastic gradient $\nabla C_{i_t}(\mathbf{w}^{(t)})$ and update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla C_{i_t}(\mathbf{w}^{(t)})$$

W3: SGD for Linear Classification

The performance of a model $\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$ on a training dataset can be measured by

$$C(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\max\{0, 1 - \underbrace{y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)}}_{\text{margin}}\} \right)^2.$$

Algorithm 1: SGD for Linear Classification

for $t = 0, 1, \ldots$ **to** T **do**
- $i_t \leftarrow$ random index from $\{1, 2, \ldots, n\}$
- **if** $y^{(i_t)}(\mathbf{w}^{(t)})^\top x^{(i_t)} \geq 1$ **then**

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)}$$

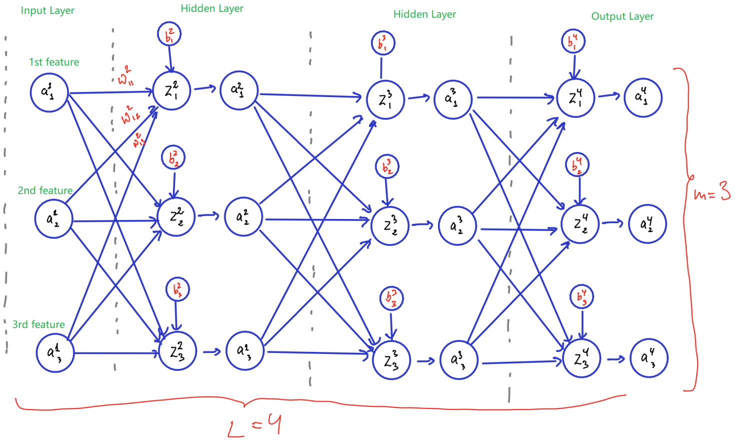
- **else**

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t (1 - y^{(i_t)} \mathbf{w}^\top \mathbf{x}^{(i_t)}) y^{(i_t)} \mathbf{x}^{(i_t)}$$

W3: Example Questions

- Compute a few steps of gradient descent on a simple optimization problem given starting point and learning rate
- Compute a few steps of stochastic gradient descent on a simple optimization problem given starting point and learning rate
- Explain the advantage/disadvantage of gradient descent and stochastic gradient descent
- Explain the intuition behind margin-based loss function

W4: Neural Network



- ω^ℓ_{jk}: "weight" of connection between k-th unit in layer $\ell-1$, to j-th unit in layer ℓ
- b^ℓ_j: "bias" of j-th unit in layer ℓ
- $z^\ell_j = \sum_k \omega^\ell_{jk} a^{\ell-1}_k + b^\ell_j$: weighted input to unit j in layer ℓ
- $a^\ell_j = \sigma(z^\ell_j)$: "activation" of unit j in layer ℓ, where σ is an "activation function"

L4: Perceptron and Neural Network

$$\mathbf{W}^\ell = \begin{pmatrix} \omega_{11}^\ell & \omega_{12}^\ell & \cdots & \omega_{1m}^\ell \\ \omega_{21}^\ell & \omega_{22}^\ell & \cdots & \omega_{2m}^\ell \\ \vdots & \vdots & \vdots & \vdots \\ \omega_{m1}^\ell & \omega_{m2}^\ell & \cdots & \omega_{mm}^\ell \end{pmatrix}, \mathbf{z}^\ell = \begin{pmatrix} z_1^\ell \\ z_2^\ell \\ \vdots \\ z_m^\ell \end{pmatrix}, \quad \mathbf{a}^\ell = \begin{pmatrix} a_1^\ell \\ a_2^\ell \\ \vdots \\ a_m^\ell \end{pmatrix}, \quad \mathbf{b}^\ell = \begin{pmatrix} b_1^\ell \\ b_2^\ell \\ \vdots \\ b_m^\ell \end{pmatrix}$$

Forward Propagation

Input: $\mathbf{x} \in \mathbb{R}^d, \mathbf{W}^\ell, \mathbf{b}^\ell, \ell = 2, \ldots, L$

Output: $\mathbf{a}^L$ ▷ We assign the original feature to the first layer

1: Set $\mathbf{a}^1 = \mathbf{x}$

2: **for** $\ell = 2, \ldots, L$ **do** ▷ from bottom to top

3: Compute the activations in ℓ-th layer via

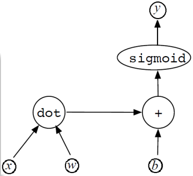
$$\mathbf{a}^\ell = \sigma\left(\mathbf{W}^\ell \mathbf{a}^{\ell-1} + \mathbf{b}^\ell\right)$$

$\mathbf{a}^\ell$ can be considered as new feature learned from data!

W5: Backpropagation

Computation Graph: a directed acyclic graph

- Node represents all the inputs and computed quantities
- Edge represents which nodes are computed directly as a function of which other (dependency)
  - from A to B: output of A is an input of B



Graph for 1-Layer NN

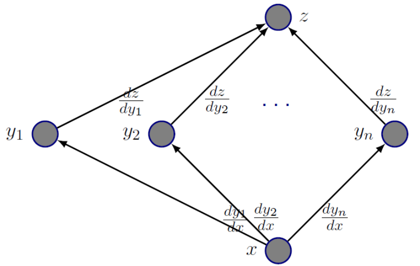
$$y = \mathrm{sigmoid}(\mathbf{w}^\top \mathbf{x} + b)$$

Chain Rule in Computation Graph

Let $\{y_1, \ldots, y_n\}$ be the successor of x. Then

$$\frac{\partial z}{\partial x} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x}$$

- **backpropagated gradient**: the gradient w.r.t. successor ($\frac{\partial z}{\partial y_j}$)
- **local gradient**: the gradient of a successor w.r.t. itself ($\frac{\partial y_j}{\partial x}$)



Backpropagation

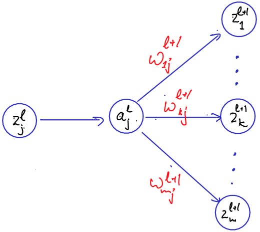
- **back-propagated** gradient: $\delta_j^\ell := \frac{\partial C}{\partial z_j^\ell}$

$$\frac{\partial C}{\partial \omega_{jk}^\ell} = \delta_j^\ell \cdot a_k^{\ell-1} \qquad \frac{\partial C}{\partial b_j^\ell} = \delta_j^\ell, \tag{2}$$

- The back-propagated gradient for the output layer is

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \sigma'(z_j^L). \tag{3}$$

- The back-propagated gradient for hidden layer is



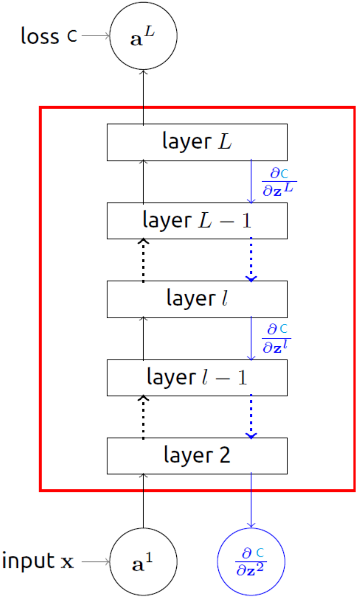
$$z_k^{\ell+1} = \sum_r \omega_{kr}^{\ell+1} a_r^\ell$$

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

$$\delta_j^\ell = \frac{\partial C}{\partial a_j^\ell} \cdot \frac{\partial a_j^\ell}{\partial z_j^\ell} = \left(\sum_k \frac{\partial C}{\partial z_k^{\ell+1}} \cdot \frac{\partial z_k^{\ell+1}}{\partial a_j^\ell}\right) \cdot \sigma'(z_j^\ell)$$

$$= \sigma'(z_j^\ell) \sum_k \delta_k^{\ell+1} \cdot \omega_{kj}^{\ell+1}$$

Backpropagation



Forward Equations

1. $\mathbf{a}^1 = \mathbf{x}$
2. $\mathbf{z}^\ell = \mathbf{W}^\ell \mathbf{a}^{\ell-1} + \mathbf{b}^\ell$
3. $\mathbf{a}^\ell = \sigma(\mathbf{z}^\ell)$
4. $C(\mathbf{a}^{(L)}, y)$

Backward Equations

1. $\delta^L = \nabla_{a^L} C \odot \sigma'(\mathbf{z}^L)$
2. $\delta^\ell = \left((\mathbf{W}^{\ell+1})^\top \delta^{\ell+1}\right) \odot \sigma'(\mathbf{z}^\ell)$
3. $\frac{\partial C}{\partial \mathbf{W}^\ell} = \delta^\ell (\mathbf{a}^{\ell-1})^\top$
4. $\frac{\partial C}{\partial \mathbf{b}^\ell} = \delta^\ell$

Note δ^ℓ is a column vector

W4 and W5: Example Questions

- Find the number of trainable parameters in a given MLP
- Given a MLP, implement forwardpropagation to compute loss function
- Given a MLP, implement backwardpropagation to compute gradient
- Derive backpropagation for simple variants of the networks we have discussed (e.g., use different activation function)
- What are local gradients and backpropagated gradients? What are their roles in the implementation of backpropagated gradients

W6: Optimisation Algorithms

Gradient Descent with Momentum

- Introduce a velocity to track historic gradients
- $\mathbf{v}^{(0)} = 0$ and update
$$\mathbf{v}^{(t+1)} = \alpha \cdot \mathbf{v}^{(t)} - \eta \nabla C(\mathbf{w}^{(t)})$$
- $\alpha \in [0, 1)$ and update
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{v}^{(t+1)}$$

Adaptive Gradient Descent: introduce accumulated gradient norm square to allow for different learning rates on different features

---

Algorithm 2: AdaGrad

Set initial $\mathbf{w}^{(0)}$ and $\mathbf{r}^{(0)} = (0, \ldots, 0)^\top$ **for** $t = 0, 1, \ldots$ **to** T **do**

$i_t \leftarrow$ random index from $\{1, 2, \ldots, n\}$ with equal probability

approximate gradient with selected example $\hat{\mathbf{g}}^{(t)} \leftarrow \nabla C_{i_t}(\mathbf{w}^{(t)})$

update the accumulated gradient norm square $\mathbf{r}^{(t+1)} \leftarrow \mathbf{r}^{(t)} + \hat{\mathbf{g}}^{(t)} \odot \hat{\mathbf{g}}^{(t)}$

update the model $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \dfrac{\eta}{\delta + \sqrt{\mathbf{r}^{(t+1)}}} \odot \hat{\mathbf{g}}^{(t)}$

W6: Example Questions

- What are some problems with plain GD/SGD?
- What is the motivation of momentum?
- What is the motivation of AdaGrad?
- Implementation of Momentum/AdaGrad on simple problems.