Week 1 Note

Introduction

Class Plan

- **Introduction/definition of learning**
- **Learning approaches for classification**
  - Logistic Regression
  - Nonlinear Transformations
  - Support Vector Machines
- **Learning approaches for regression:**
  - Linear regression and its closed form solution
  - Support Vector Regression
- **Kernels as similarity functions**
  - Other kernel-based methods
- **Optimisation algorithms:**
  - Gradient Descent and its weaknesses
  - Newton Raphson and Iterative Reweighted Least Squares
  - Sequential Minimal Optimisation
- **Fundamentals:**
  - Is learning feasible? Learning theory
  - Bias and variance
  - Generalisation, overfitting and regularisation
  - Validation and model selection

Assessment

- Graded quizzes(**20%**)
  - Two Summative canvas quizzes worth 10% each
  - Timed for 1h max
  - Can start at any time between the release and due times
  - Your assignment will be automatically submitted at the due time, even if you started less than 1h before
  - deadline is **strict**
  - Marks and feedback released in the week after the quiz
- Exam(**80%**)
  - Main summer assessment period(May maybe)

Office Hours

- Room UG39
- Friday 14:00
- Friday 15:00

Supervised Learning

- Learns a mapping from inputs $\vec{x} = (x_1, ..., x_d)^T \in X$ to outputs $y \in Y$, given a `training set` of input-output pairs $J = \{(\vec{x}^1, y^1), (\vec{x}^2, y^2), ..., (\vec{x}^n, y^n)\}$

$$J = \{(\vec{x}^i, y^i)\}_{i=1}^{N}$$

  - The Output Space y
    - **Regression**: $y = R$
    - **Classification**: Y is a set of categories
      - 2 categories: binary classification
      - 2 categories: multi-class classification
  - The Input Space X
    - d-dimensional space, where each dimension can be:
      - `Numeric`
      - `Ordinal`
      - `Categorical`

> Use `Training Examples` $J = \{(\vec{x}^i, y^i)\}_{i=1}^{N}$ and `Hypothesis Set` $h(\vec{x}) = \vec{a}^T\vec{x} + b, \forall\vec{a} \in \mathbb{R}^d, b \in \mathbb{R}$ to `Learn Algorithm` and get `Final Hypothesis` $g \approx f$

- Problem Setting
  - `Given` a set of training examples $J = \{(\vec{x}^1, y^1), (\vec{x}^2, y^2), ..., (\vec{x}^n, y^n)\}$ where $(\vec{x}^i, y^i) \in X \times Y$ are drawn from a fixed albeit unknown joint probability distributin $p(\vec{x}, y) = p(y|\vec{x})p(\vec{x})$
  - `Goal`: to learn a function $g : X \to Y$ able to generalise to unseen (test) examples of the same probability distribution $p(\vec{x}, y)$
    - $g : X \to Y$, mapping input space to output space
    - g as a probability distribution approximating $P(y|\vec{x})$
  - `Generalisation`: minimise $E(g) = P_{(\vec{x},y)\sim p[g(\vec{x})\neq y]}$ or $E(g) = \mathrm{E}[(g(\vec{x}) - y)^2]$

Logistic Regression

- In `Logistic Regression`, we will model the probability (actually the odds) of an instance to belong to a given class as a `linear combination of the inputs.`
- Odds and Logit

- **Odds**: ratio of probabilities of two possible outcomes:

$$o_1 = \frac{p_1}{p_0} = \frac{p_1}{1-p_1}$$

> If $o_1 \geq 1$, predict `class 1`
> If $o_1 < 1$, predict `class 0`

- **Logit(aka. log-odds)**: the logarithm of the odds:

$$Logit(p_1) = \vec{w}^T \vec{x}$$

where $logit(p_1) = \ln(\frac{p_1}{1-P_1})$

> Logit enables us to map from **[0,1]** to **[$-\infty, \infty$]**
> If $logit(p_1) \geq 0$, predict `class 1`
> If $logit(p_1) < 0$, predict `class 0`

- In the case above, we know:

$$p_1 = \frac{e^{\vec{w}^T \vec{x}}}{1 + e^{\vec{w}^T \vec{x}}}$$

$$p_0 = 1 - p_1 = \frac{1}{1 + e^{\vec{w}^T \vec{x}}}$$

$$h(\vec{x}) = p_1 = p(1|\vec{x}, \vec{w}), \forall \vec{w} \in \mathbb{R}^{d+1}$$

Likelihood

- **Likelihood function**

$$\begin{align}\prod_{i=1}^{N} P_{y^i} &= \prod_{i=1}^{N} p(y^i|x^i; \vec{w}) = p(\vec{y}|\vec{X}, \vec{w}) = L(\vec{w}) \tag{1} \\ &= \underbrace{\prod_{i=1}^{N} p(1|\vec{x}^i, \vec{w})^{(y_i)}(1 - p(1|\vec{x}^i, \vec{w}))^{(1-y_i)}}_{\text{这一段使用了Bernoulli distribution进行转换}} \tag{2}\end{align}$$

> 把所有例子$\vec{X}$与对应的结果$\vec{y}$, 根据代求的参数$\vec{w}$全部汇总为一个式子$L(\vec{w})$
> $\vec{x}$代表一个例子所需要的输入, $\vec{X}$代表所有例子的输入(汇总)
>
> $$\vec{X} = \begin{pmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \dots & \dots & \dots & \dots \\ x_1^N & x_2^N & \dots & x_d^N \end{pmatrix}$$
>
> $$\vec{y} = \begin{pmatrix} y^1 \\ y^2 \\ \dots \\ y^N \end{pmatrix}$$

- **Log-Likelihood**

$$\ln(L(\vec{w})) = \ln\prod_{i=1}^{N} P_{y^i} = \sum_{i=1}^{N} \ln P_{y^i}$$

> 把求$\arg\max_w L(\vec{w})$转换为$\arg\max_w \ln L(\vec{w})$
> 作用: ln函数单调递增, 使数值更为稳定(stable)

- **Loss Function**

$$\begin{aligned} E(\vec{w}) &= -\ln(L(\vec{w})) = -\sum_{i=1}^{N} \ln P_{y^i} &(3) \\ &= -\sum_{i=1}^{N} y^i \ln p(1|\vec{x}^i, \vec{w}) + (1-y^i)\ln(1-p(1|\vec{x}^i, \vec{w})) &(4) \end{aligned}$$

> 把求$\arg\max_w \ln L(\vec{w})$转换为$\arg\min_w E(\vec{w})$
> 作用: 把问题转换为求最小值问题(而非最大值), 符合尝试

Gradient Descent

- Gradient descent adjusts $\vec{w}$ iteratively in the direction that leads to the biggest decrease (steepest descent) in $E(\vec{w})$.

$$x := x - \eta\frac{df}{dx}$$

即

$$\vec{w} = \vec{w} - \eta\bigtriangledown E(\vec{w})$$

where $\eta > 0$ and $\bigtriangledown E(\vec{w}) = \sum_{i=1}^{N}(p(1|\vec{x}^i, \vec{w}) - y^i)\vec{x}^i$

> η是学习率(Learning rate), 是超参数(Hyperparameter)，由使用者决定
> 过高的η可能导致无法找到最低点 过低的η可能导致找的效率变低

- Gradient descent is a general purpose optimisation algorithm. But is likely to get stuck in `local minima`

> For **logistic regression** using **cross-entropy loss**, this is **not a problem**, as its $E(\vec{w})$ is `strictly` convex with respect to $\vec{w}$, having a single unique minimum.

Addition

Equivalent Terms

- $\vec{x}$: input attribute, input feature, independent variable, input variable.
- y: output attribute, output variable, dependent variable, label (for classification).
- mapping: learned function, predictive model, classifier (for classification).
- Learning a model, training a model, building a model.
- J: set of training examples, training data.
- $(\vec{x}, y)$: example, observation, data point, instance (more frequently used for examples with unknown outputs).
- Different people and books will use different notations!

Notation

- Scalar: lower case, e.g., b
- Column Vector: lower case, bold, e.g., $\vec{x}$
- Vector element: lower case with subscript, e.g., x_i
- Matrix: upper case, bold, e.g., X
- Matrix element: upper case with subscripts,e.g. $X_{i,j}$
- If enumerating these (e.g., having multiple vectors), superscript will be used to differentiate this from indices, e.g., $\vec{x}^i$