Contents

1 Probability Basics **3**
1.1 Probability Rules . . . 3
1.1.1 Expectation: . . . 3
1.2 Common Distributions . . . 3

2 Vector Calculus **4**
2.1 General Formulae . . . 4
2.2 Pseudo Inverse . . . 4

3 Linear Regression **4**

4 Non-linear Regression **5**

5 MLE and Bayesian Parameter Estimation **6**
5.1 Why Maximize the Likelihood $p(\mathcal{D} \mid \theta)$? $(\theta = (\mu, \sigma^2))$ . . . 6
5.2 How is it related to $p(\theta \mid \mathcal{D})$? . . . 6
5.3 Bringing it Together . . . 6
5.4 Notation . . . 6
5.5 Bayesian Methodology . . . 7
5.6 Linear Regression via Maximum a Posteriori . . . 7
5.7 Bayesian Linear Regression . . . 8

6 Logistic Regression/ Linear Classification **8**
6.1 Gradient Descent in Logistic Regression . . . 9
6.2 Softmax Function (Tut 2) . . . 9

7 Naïve Bayes Model **12**
7.1 Introduction . . . 12
7.2 Derivation of Loss for Naïve Bayes . . . 12
7.3 Prediction . . . 14
7.4 Extension 1: Multiclass Classification . . . 15
7.5 Extension 2: Non-Gaussian Features . . . 15
7.6 Extension 3: Missing Data . . . 15

8 Connection between Naive Bayes and Logistic Regression (Tut3) **16**

9 Bias Variance Decomposition **18**
9.1 Introduction . . . 18
9.2 The Decomposition . . . 18

10 PAC **19**
10.1 Introduction . . . 19
10.2 Is Learning Probable? . . . 19
10.3 Hoeffding's inequality . . . 19
10.4 Developing a Bound for a Given Hypothesis y . . . 20
10.5 Developing a Bound for All Hypotheses y in the Entire Hypothesis Space $\mathcal{H}$ . . . 20
10.6 Limitations . . . 21

11 Neural Networks **21**
11.1 Neural Networks as Adaptive Basis Functions 21
11.2 Backpropagation 22
11.3 Gradient Descent 22
11.4 Stochastic Gradient Descent 23
11.5 Tutorial 6 (Gradient Descent and Stochastic Gradient Descent) 23
11.5.1 Gradient Descent 23
11.5.2 Stochastic Gradient Descent 24
11.6 Convolution 25
11.7 Transformers 25

12 Kernels **26**
12.1 Non-parametric Linear Regression 26
12.2 Valid Kernels 26
12.3 Lagrangians 27

13 Tutorial 9: Multivariate Gaussians **28**
13.1 SVM 29
13.2 Quadratic Program 29
13.3 Applying Lagrange multipliers 29

14 K-Means Clustering **30**

15 Mixture Models **31**

16 EM Algorithm **32**
16.1 Using it in practice 33
16.2 Finding the next iteration of $\boldsymbol{\theta}$ 34
16.3 Exponential Mixture Model 34

17 Variational Autoencoder (VAE) **35**
17.1 Data, Model and Loss 35
17.2 Optimization and the reparameterization trick 37

1 Probability Basics

1.1 Probability Rules

- **Marginalization:** $p(x) = \int_{y \in Y} p(x, y) dy$.
- **Joint Probability:** $p(x, y) = p(x \mid y)p(y) = p(y \mid x)p(x)$.
- **Bayes' Theorem:** $p(x \mid y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x)p(x)dx}$.
- **Independence:** x and y are *independent* $\iff p(x, y) = p(x)p(y)$.

1.1.1 Expectation:

Written as $\mathbb{E}_{\text{distribution}}[\text{variable}]$, subscript omitted if distribution obvious.

- **Expectation for Discrete Random Variables**: $\mathbb{E}_{p(X)}[X] = \sum_x (xp(x))$ if discrete, $\int (xp(x))\,\mathrm{d}x$ if continuous.
- **Expectation of a Function of** x: $\mathbb{E}_{p(X)}[f(X)] = \int (f(x)p(x))\,\mathrm{d}x$
- **Linearity of Expectations**:
  - $\mathbb{E}_{p(X,Y)}[X + Y] = \mathbb{E}_{p(X)}[X] + \mathbb{E}_{p(Y)}[Y]$
  - $\mathbb{E}_{p(X)}[aX] = a\mathbb{E}_{p(X)}[X]$
- **Variance:** $\mathrm{Var}_{p(X)}(X) = \mathbb{E}_{p(X)}\left[\left(X - \mathbb{E}_{p(X)}[X]\right)^2\right] = \mathbb{E}_{p(X)}\left[X^2\right] - \mathbb{E}_{p(X)}[X]^2$

1.2 Common Distributions

$X \sim$	$p(x) =$	$\mathbb{E}_{p(x)}[X] =$	$\mathrm{Var}_{p(X)}(X) =$
$\mathcal{N}(\mu, \sigma^2), \mu \in \mathbb{R}, \sigma^2 > 0$	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	μ	σ^2
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu} \in \mathbb{R}^D, \Sigma \in \mathbb{R}_+^{D \times D}$	$\frac{1}{(2\pi)^{D/2}\lvert\Sigma\rvert^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$	$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$
$\mathrm{Bern}[\pi], \pi \in [0, 1]$	$\pi^x(1-\pi)^{1-x}$, $x \in \{0, 1\}$	π	$\pi(1-\pi)$
$B(n, p)$, $n \in \mathbb{Z}^+$, $p \in [0, 1]$	$\binom{n}{x} p^x (1-p)^{n-x}$, $x \in \{0, \ldots, n\}$	np	$np(1-p)$
$\mathrm{Poisson}(\lambda)$, $\lambda > 0$	$\frac{\lambda^x e^{-\lambda}}{x!}$, $x \in \mathbb{N}$	λ	λ
$\mathrm{Gamma}(\alpha, \beta)$, $\alpha, \beta > 0$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$, $x > 0$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$
$\mathrm{Cat}(\boldsymbol{\pi})$, $\pi_i \geq 0, \sum_{i=1}^K \pi_i = 1$	$\prod_{i=1}^K \left(\pi_k^{x_i}\right)$	$\boldsymbol{\pi}$	$\mathrm{diag}(\boldsymbol{\pi}) - \boldsymbol{\pi}\boldsymbol{\pi}^\top$

Types of Covariance: $\Sigma_{\text{spher}} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}, \Sigma_{\text{diag}} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}, \Sigma_{\text{full}} = \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 \end{bmatrix}$

If we have $X_1, \ldots, X_n$ where any given $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, then

$$X = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}^\top \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \boldsymbol{\mu} = \begin{bmatrix} \mu_1 & \cdots & \mu_n \end{bmatrix}^\top, \Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$$

2 Vector Calculus

If $w \in \mathbb{R}^N$, then

$$\nabla_w E(w) = \begin{bmatrix} \frac{\partial E}{\partial w_1} & \cdots & \frac{\partial E}{\partial w_N} \end{bmatrix}^\top.$$

For $\mathbf{w}, \mathbf{x} \in \mathbb{R}^N, A \in \mathbb{R}^{N \times N}$, where $A^\top = A$,

- $\nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{x}) = \mathbf{x}$
- $\|\mathbf{w}\|_2^2 = \sum_{i=1}^n |w_i|^2$, $\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^n |w_i|^2}$
- $\nabla_{\mathbf{w}}(\|\mathbf{w}\|_2) = \frac{\mathbf{w}}{\|w\|_2}$
- $\nabla_{\mathbf{w}}\left(\|\mathbf{w}\|_2^2\right) = \nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{w}) = 2\mathbf{w}$
- $\nabla_{\mathbf{w}}(\mathbf{w}^\top A \mathbf{w}) = 2A\mathbf{w}$

2.1 General Formulae

For $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x} \in \mathbb{R}^N, A, B, C, X \in \mathbb{R}^{N \times N}$,

- $\nabla_{\mathbf{x}}(\mathbf{x}^\top \mathbf{a}) = \nabla_{\mathbf{x}}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}$
- $\nabla_X(\mathbf{a}^\top X \mathbf{a}) = \nabla_X(\mathbf{a}^\top X^\top \mathbf{a}) = \mathbf{a}\mathbf{a}^\top$
- $\nabla_{\mathbf{x}}(\mathbf{x}^\top A \mathbf{x}) = (A + A^\top)\mathbf{x}$
- $\nabla_X(\mathbf{a}^\top X^\top X \mathbf{b}) = X(\mathbf{a}\mathbf{b}^\top + \mathbf{b}\mathbf{a}^\top)$
- $\nabla_{\mathbf{x}}((A\mathbf{x} + \mathbf{a})^\top B(C\mathbf{x} + \mathbf{b})) = A^\top B(C\mathbf{x} + \mathbf{b}) + C^\top B^\top(A\mathbf{x} + \mathbf{a})$
- $\nabla_X((X\mathbf{a} + \mathbf{b})^\top C(X\mathbf{a} + \mathbf{b})) = (C + C^\top)(X\mathbf{a} + \mathbf{b})\mathbf{a}^\top$
- $\nabla_X(\mathbf{a}^\top X \mathbf{b}) = \mathbf{a}\mathbf{b}^\top, \nabla_{\mathbf{b}}(\mathbf{a}^\top X \mathbf{b}) = X^\top \mathbf{a}$
- $\nabla_X(\mathbf{a}^\top X^\top C X \mathbf{b}) = C^\top X \mathbf{a}\mathbf{b}^\top + C X \mathbf{b}\mathbf{a}^\top$

2.2 Pseudo Inverse

Assume $\mathbf{A}^+$ to be pseudo-inverse of $\mathbf{A}$ and $\mathbf{A}$ has full rank,

$$\begin{aligned}
\mathbf{A}_{n\times n} \quad \text{Square} \quad \text{rank}(\mathbf{A}) = n \quad &\implies \quad \mathbf{A}^+ = \mathbf{A}^{-1} \\
\mathbf{A}_{n\times m} \quad \text{Broad} \quad \text{rank}(\mathbf{A}) = n \quad &\implies \quad \mathbf{A}^+ = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} \\
\mathbf{A}_{n\times m} \quad \text{Tall} \quad \text{rank}(\mathbf{A}) = m \quad &\implies \quad \mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T
\end{aligned}$$

Broad version is known as right inverse and tall version is known as left inverse

3 Linear Regression

A linear regression model is formally defined by its **data**, **model**, and **loss**.

- Data $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$ is a set of all inputs $\mathbf{x}$ and their targets t.
- Model $y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ is a function parameterized on some given weights that predicts an output for a given input.
- Loss is a function that tells how "bad" a model is on a data set, $\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \left((\mathbf{w}^\top \mathbf{x}_n - t_n) \right)^2 \Big)$.

Analytical solution for the weights that minimize the loss for a given linear regression model:

$$\mathbf{w}_{\text{ML}} = \arg\min_{\mathbf{w}} (\mathcal{L}(\mathbf{w})) = (X^\top X)^{-1} X^\top t$$

Proof We can express $\mathcal{L}(w) = \frac{1}{2}(t - X\mathbf{w})^\top (t - X\mathbf{w})$ where

$$X = \begin{bmatrix} \mathbf{x}_1^\top & \mathbf{x}_2^\top & \cdots & \mathbf{x}_N^\top \end{bmatrix}^\top, \quad t = \begin{bmatrix} t_1 & t_2 & \cdots & t_N \end{bmatrix}^\top$$

we have

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2}(\mathbf{t} - X\mathbf{w})^\top(\mathbf{t} - X\mathbf{w}) \\ &= \frac{1}{2}(\mathbf{t}^\top\mathbf{t} - \mathbf{t}^\top X\mathbf{w} - \mathbf{w}^\top X^\top\mathbf{t} + \mathbf{w}^\top X^\top X\mathbf{w}) \\ &= \frac{1}{2}(\mathbf{t}^\top\mathbf{t} - 2\mathbf{w}^\top X^\top\mathbf{t} + \mathbf{w}^\top X^\top X\mathbf{w}) \qquad (\text{Since } \mathbf{t}^\top Xw = \mathbf{w}^\top X^\top\mathbf{t} \in \mathbb{R})\end{aligned}$$

Therefore, from

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = X^\top X\mathbf{w} - X^\top\mathbf{t} = 0$$

we have

$$\mathbf{w} = (X^\top X)^{-1}X^\top\mathbf{t}$$

4 Non-linear Regression

Non-linear regression is largely the same as linear regression, except that we now introduce *features* by transforming our inputs with *basis functions* ϕ_i. Then, we can treat our model as a linear model using the generated features instead of the inputs. Common basis functions include polynomials and trigonometric functions.

A potential issue with introducing more features is *overfitting*. A symptom of overfitting is weights growing too large. *Regularization* is a common solution to this. By introducing a *regularization term* $\lambda R(f)$ to our loss function, we penalize weights that are overly large, preventing overfitting. A common regularizer is to take the sum of weight magnitudes $\|\mathbf{w}\|_2^2$. $\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2$

Analytical solution for a given linear regression model after regularization:

$$\mathbf{w}_{RR} = (X^\top X + \lambda I)^{-1}X^\top t$$

Proof

$$\begin{aligned}\tilde{\mathcal{L}}(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\left(\left(\mathbf{w}^\top\phi(x_n) - t_n\right)^2\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \\ &= \frac{1}{2}\|\Phi\mathbf{w} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \\ \implies \nabla_{\mathbf{w}}\tilde{\mathcal{L}}(\mathbf{w}) &= \Phi^\top(\Phi\mathbf{w} - \mathbf{t}) + \lambda\mathbf{w} \\ &= (\Phi^\top\Phi + \lambda I)\mathbf{w} - \Phi^\top\mathbf{t} = \mathbf{0} \\ \therefore \mathbf{w} &= (\Phi^\top\Phi + \lambda I)^{-1}\Phi^\top\mathbf{t}\end{aligned}$$

In general, basis functions and regularization are two methods of providing *inductive bias*, which is anything that nudges the algorithm to learn one pattern over another.

5 MLE and Bayesian Parameter Estimation

5.1 Why Maximize the Likelihood $p(\mathcal{D} \mid \theta)$? ($\theta = (\mu, \sigma^2)$)

1. **Computational Simplicity**: Often, the likelihood function $p(\mathcal{D} \mid \theta)$ is mathematically easier to work with. It frequently leads to tractable optimization problems.
2. **Frequentist Perspective**: In the frequentist view of statistics, parameters are considered fixed but unknown quantities. Thus, it doesn't make sense to talk about the "probability" of parameters. Only data are considered to be generated from a random process.
3. **Point Estimation**: MLE provides a single "best guess" point estimate of the model parameters that makes the observed data most likely.

5.2 How is it related to $p(\theta \mid \mathcal{D})$?

1. **Bayesian Perspective**: If you're interested in $p(\theta \mid \mathcal{D})$, the Bayesian framework explicitly tackles this by using Bayes' Theorem. Bayesian methods do allow us to talk about the probability distribution over parameters.
$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta) \times p(\theta)$$
2. **Maximum A Posteriori (MAP)**: If you take the Bayesian route but still want a point estimate, you could use the MAP estimate, which maximizes $p(\theta \mid \mathcal{D})$. Note that MLE is a special case of MAP where the prior $p(\theta)$ is uniform.
3. **Intuitive Connection**: Maximizing $p(\mathcal{D} \mid \theta)$ indirectly gives us parameters that are likely, assuming the data we observed is likely to occur again. The Bayesian approach refines this by adding a prior belief $p(\theta)$ to create a more nuanced understanding.

5.3 Bringing it Together

Maximizing $p(\mathcal{D} \mid \theta)$ and $p(\theta \mid \mathcal{D})$ are two sides of the same coin, related through Bayes' Theorem. The former is easier computationally and aligns with frequentist philosophy, while the latter incorporates prior beliefs and is the cornerstone of Bayesian methods.

5.4 Notation

- $\mathbf{t} = [t_1, ..., t_N]^T$, $\mathbf{w} = [w_0, ..., w_{M-1}]^T$
- $X = \begin{bmatrix} \text{———} \mathbf{x}_1^T \text{———} \\ \text{———} \mathbf{x}_2^T \text{———} \\ \vdots \\ \text{———} \mathbf{x}_N^T \text{———} \end{bmatrix}$, $\boldsymbol{\phi} = \begin{bmatrix} \text{———} \phi(\mathbf{x}_1)^T \text{———} \\ \text{———} \phi(\mathbf{x}_2)^T \text{———} \\ \vdots \\ \text{———} \phi(\mathbf{x}_N)^T \text{———} \end{bmatrix}$
- $\mathbf{y} = [\mathbf{w}^T \phi(\mathbf{x}_1), ..., \mathbf{w}^T \phi(\mathbf{x}_N)]^T = \boldsymbol{\phi}\mathbf{w}$

The data is $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$, where $t = \mathbf{w}^\top \phi(\mathbf{x}) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ being the noise (we assume noise is Gaussian, but this may not always be the case, e.g., when data is consistantly higher/lower than underlying $\mathbf{w}^\top \phi(\mathbf{x})$). Assume variance σ^2 is known. As $\mathbf{w}^\intercal \phi(\mathbf{x})$ is not a random variable, we can write

$$t \sim \mathcal{N}(\mathbf{w}^\intercal \phi(\mathbf{x}), \sigma^2)$$

The key idea behind MLE is that we want to maximise the probability of $\mathbf{w}^\intercal \phi(\mathbf{x}) + \epsilon$ being equal to the actual t in our data. However, this is often very hard to calculate, so we simplify our problem by

maximising the likelihood[1]. This is given by

$$\begin{aligned}
\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w}) &= \max_{\mathbf{w}} \log(p(\mathcal{D}|\mathbf{w})) && \text{(log is increasing in } (0,1]) \\
&= \max_{\mathbf{w}} \log(\prod_n p(t_n|\mathbf{x}_n, \mathbf{w})) && \text{(assume i.i.d.)} \\
&= \max_{\mathbf{w}} \sum_n \log(p(t_n|\mathbf{x}_n, \mathbf{w})) && \text{(log rules)} \\
&= \max_{\mathbf{w}} \sum_n \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(t_n - \mathbf{w}^\intercal\phi(\mathbf{x}))^2}{2\sigma^2}\right)\right] && \text{(p.d.f. of Gaussian)} \\
&= \max_{\mathbf{w}} \sum_n \cancel{\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right]} + \log\left[\exp\left(-\frac{(t_n - \mathbf{w}^\intercal\phi(\mathbf{x}))^2}{2\cancel{\sigma^2}}\right)\right] && \text{(cancel constant terms)} \\
&= \max_{\mathbf{w}} \sum_n -\frac{(t_n - \mathbf{w}^\intercal\phi(\mathbf{x}))^2}{2} \\
&= \min_{\mathbf{w}} \sum_n \frac{(t_n - \mathbf{w}^\intercal\phi(\mathbf{x}))^2}{2}
\end{aligned}$$

We now have the loss function which will maximise the probability of our model being correct.

5.5 Bayesian Methodology

Bayesian Prediction: $p(H|D) = \sum_\theta p(H, \theta|D) = \sum_\theta p(H|\theta)p(\theta|D)$
MAP ("Poor Man's Bayes"): $\theta_{MAP} = argmax_\theta(p(\theta|D)), \quad p(H|D) = p(H|\theta_{MAP})$

5.6 Linear Regression via Maximum a Posteriori

From the Bayesian perspective, the weights $\mathbf{w}$ are not predetermined but follow a probability distribution. Before the Bayesian sees any data, he already has a preconceived notion on what distribution $\mathbf{w}$ should follow. After seeing some data, the Bayesian will update his beliefs about how $\mathbf{w}$ is distributed. Thus, instead of maximising $p(\mathcal{D}|\mathbf{w})$ like the frequentist, the Bayesian wants to maximise $p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$ by Bayes' theorem, which is the probability that $\mathbf{w}$ are the weights given the data. We are assuming the same data and model as section before, with the additional prior that $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$ and $\mathbf{t} \sim \mathcal{N}(\phi\mathbf{w}, \beta^{-1}\mathbf{I})$ ($\alpha^{-1} = \sigma_w^2$, $\beta^{-1} = \sigma_\epsilon^2$). What the prior tells us is that we assume that the weights are more likely to be close to 0 according to the multivariate Gaussian distribution.

$$\begin{aligned}
\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) &= \max_{\mathbf{w}} \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} && \text{(Bayes' theorem)} \\
&= \max_{\mathbf{w}} \log(p(\mathcal{D}|\mathbf{w}) + \log(p(\mathbf{w})) - \cancel{\log(p(\mathcal{D}))} && \text{(cancel constants w.r.t. } \mathbf{w}) \\
&= \max_{\mathbf{w}} \underbrace{\log(p(\mathcal{D}|\mathbf{w})}_{\text{MLE}} + \underbrace{\log(p(\mathbf{w}))}_{\text{log prior}}
\end{aligned}$$

We will continue expanding the log prior until we get the L2 regulariser.

$$\begin{aligned}
\log(p(\mathbf{w})) &= \log(\frac{1}{(2\pi)^{k/2}\det(\alpha^{-1}\mathbf{I})^{1/2}} \exp(-\frac{1}{2}\mathbf{w}^\intercal(\alpha\mathbf{I})\mathbf{w})) && \text{(p.d.f. of multivariate Gaussian)} \\
&= \log((2\pi)^{-k/2}\det(\alpha^{-1}\mathbf{I})^{-1/2}) + \log(\exp(-\frac{1}{2}\mathbf{w}^\intercal(\alpha\mathbf{I})\mathbf{w})) && \text{(log rules)} \\
&= c_1 - \frac{\alpha}{2}\mathbf{w}^\intercal\mathbf{w} && \text{(combine constant terms w.r.t. } p(\mathbf{w}|\mathcal{D}))
\end{aligned}$$

[1]Likelihood is different from probability. Consider a uniform distribution of $[0, 1]$. The likelihood of 0.5 is 1, while the probability is 0.

We get $\mathbf{w}^\intercal\mathbf{w}$, the sum of squares of each of the weights.
We perform the same steps for the other term using $p(D|\mathbf{w}) = N(t \mid \phi\mathbf{w}, \beta^{-1}I)$ to get $c_2 - \frac{\beta}{2}\|t - \phi w\|_2^2$.
From this, we can see that a prior of multivariate Gaussian distribution results in the L2 regulariser, and as seen in tutorial 3, different priors result in different regularisers, such as the Laplace distribution resulting in the L1 regulariser and the uniform distribution resulting in no regularisation.

5.7 Bayesian Linear Regression

We have seen how to get a point estimate using MAP, but we can also apply Bayesian Linear Regression to obtain the distribution over the weights.
As before, assume the prior that $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$ and $\mathbf{t} \sim \mathcal{N}(\boldsymbol{\phi}\mathbf{w}, \beta^{-1}\mathbf{I})$. We apply Bayes' rule:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} = \mathcal{N}(\mathbf{w}|m_N, S_N)$$

$$\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \min_{\mathbf{w}} \frac{1}{2}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}, \quad \lambda = \frac{\alpha}{\beta}.$$

where the analytical solution is given by

$$m_N = s_N\left(s_0^{-1}m_0 + \beta\boldsymbol{\phi}^T\mathbf{t}\right) \quad \text{and} \quad s_N = \left(s_0^{-1} + \beta\boldsymbol{\phi}^T\boldsymbol{\phi}\right)^{-1}.$$

In this case, $m_0 = \mathbf{0}$ and $s_0^{-1} = \alpha\mathbf{I}$, which simplifies to

$$m_N = \left(\boldsymbol{\phi}^T\boldsymbol{\phi} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1}\boldsymbol{\phi}^T\mathbf{t},$$

which is very similar to our ridge regression solution $\mathbf{w}_{RR} = \left(\boldsymbol{\phi}^T\boldsymbol{\phi} + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\phi}^T\mathbf{t}$.

Here, α is the precision of our prior and β is the precision of the observed noise.
Once we have our posterior $p(\mathbf{w}|\mathcal{D})$, we can make a prediction $\mathbf{t}'$ upon observing some new data $\mathbf{x}'$:

$$\begin{aligned} p(\mathbf{t}'|\mathbf{x}', \mathcal{D}) &= \int_{\mathbf{w}} p(\mathbf{t}'|\mathbf{x}', \mathbf{w})p(\mathbf{w}|\mathcal{D})\, d\mathbf{w} \\ &= N(t' \mid \underbrace{\phi(x')^T m_N}_{\text{mean prediction}}, \underbrace{\phi(x')^T S_N \phi(x')}_{\text{variance}}) \\ &= \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}\left[p(\mathbf{t}'|\mathbf{x}', \mathbf{w})\right] \end{aligned}$$

$$\min_{q \in Q} \mathcal{L}(q) = -\mathbb{E}_{q(w)}\big[\log p(\mathcal{D} \mid w)\big] + \mathbb{D}_{\mathrm{KL}}\big[q(w) \,||\, p(w)\big]$$

where Q is set of probability distributions.

6 Logistic Regression/ Linear Classification

- **Activation Function**: The activation function used in logistic regression is the sigmoid function, denoted by $\sigma(z) = \frac{1}{1+\exp(-z)}$.
- **Nice Properties of Sigmoid**:
  - $\sigma(-z) = 1 - \sigma(z)$
  - $z = \log\left(\frac{\sigma(z)}{1-\sigma(z)}\right)$ ("log odds")
  - $\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$

- **Likelihood**: The likelihood of the data given the parameters is

$$p(D|\theta) = \prod_{n=1}^{N} p(t_n|x_n, w) = \prod_{n=1}^{N} \text{Bern}_{t_n}(y_n) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{(1-t_n)}$$

  where $y_n = \sigma(w^T\phi(x_n)) = p(t_n = 1|x_n)$ and $\phi(x_n)$ is the basis function.

- **Prior**: The prior distribution over the weight vector w is a Gaussian centered at zero,

$$p(w) = \mathcal{N}(w|0, \alpha^{-1}I)$$

- **Maximum A Posteriori Estimate**:

$$w_{\text{MAP}} = \arg\min_w -\log p(w|D) = \underbrace{-\sum_{n=1}^{N}[t_n \log y_n + (1-t_n)\log(1-y_n)]}_{\text{binary cross-entropy error/BCE}} + \underbrace{\frac{\alpha}{2}w^T w}_{\text{regularizer}}$$

- **Loss Function**: The cross-entropy error term encourages y_n to match t_n.

- **Optimization**: There is no analytical solution for w in logistic regression, so we have to use gradient descent or other optimization techniques.

6.1 Gradient Descent in Logistic Regression

The update equation for gradient descent in logistic regression is given by:

$$w_{k+1} = w_k - \eta\nabla_w L(w)$$

where η is the learning rate, and $\nabla_w L(w)$ is the gradient of the loss function with respect to the weight vector w.
The gradient $\nabla_w L(w)$ can be computed as:

$$\nabla_w L(w) = \sum_{n=1}^{N}(y_n - t_n)\phi_n + \alpha\mathbf{w}$$

where $y_n = \sigma(w^T\phi_n)$ for classification, $w^T\phi_n$ for regression. Here, N is the total number of data points, y_n is the predicted label, t_n is the true label, ϕ_n is the feature vector for the n-th data point, and α is the regularization parameter.

6.2 Softmax Function (Tut 2)

- **Log Odds for class j against K classes**:

$$z_j = \ln\left(\frac{p(t=j|x)}{p(t=K|x)}\right)$$

- **Learning Weights**:

$$w_j^\top x \text{ regresses towards } z_j$$

- **Probability of class j given the features x**:

$$p(t=j|x) = p(t=K|x)\cdot\exp(z_j)$$

- **Probability of class K given x**:

$$p(t = K|x) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp(z_i)}$$

- **Softmax Function**:

$$p(t = j|x) = \frac{\exp(z_j)}{\sum_{i=1}^{K} \exp(z_i)}$$

- **Weight Matrix W**:

$$W = \begin{bmatrix} w^{(1)} & \cdots & w^{(K)} \end{bmatrix}$$

- **Vector of Probabilities via Softmax**:

$$\begin{bmatrix} p(t = 1|x, W) \\ \vdots \\ p(t = K|x, W) \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} w^{(1)\top} x \\ \vdots \\ w^{(K)\top} x \end{bmatrix} \right) = \text{softmax}(W^\top x)$$

Perform MLE:
Now that we have a reasonable choice of the likelihood function $p(t|x, W)$, we are now ready to perform MLE. Using the maximum likelihood paradigm, derive an error function $E(W)$ for the multiclass logistic regression. You may keep any logsumexp terms in your solution without further simplification.

K classes:

$$p(t) = \text{cat}(t|\mu) = \prod_{i=1}^{K} \mu_i^{t_i}, \quad \mu = [\mu_1, \ldots, \mu_K]^T, \quad t = [0, \ldots, 1, \ldots, 0]^T \quad \text{(one hot vector)}$$

Let $\mathbb{I}$ be the indicator variable where

$$\mathbb{I}[A] = \begin{cases} 1 & \text{if A is true} \\ 0 & \text{if A is false.} \end{cases}$$

Then we can rewrite

$$p(t_n|x_n, \mathbf{W}) = \prod_{i=1}^{K} p(t_n = i|x_n, \mathbf{W})^{\mathbb{I}[t_n = i]}.$$

To see why, note that only one of the $\mathbb{I}[t_n = i]$ is true, for which the term $p(t_n = i|x_n, \mathbf{W})$ survives, but the other terms $j \neq i$ in the product are going to be raised to the power of 0 and be eliminated from the product. Thus, if $t_n = i$, then this product is just $p(t_n = i|x_n, \mathbf{W})$. Notice this is the probability mass function of the Categorical distribution, which is the multiclass generalization of the Bernoulli distribution. The likelihood of observing $\mathcal{D}$ is

$$\begin{aligned}
\log p(\mathcal{D}|\mathbf{W}) &= \log \prod_{n=1}^{N} p(t_n|x_n, \mathbf{W}) \\
&= \log \prod_{n=1}^{N} \prod_{i=1}^{K} p(t_n = i|x_n, \mathbf{W})^{\mathbb{I}[t_n = i]} \\
&= \sum_{n=1}^{N} \sum_{i=1}^{K} \log p(t_n = i|x_n, \mathbf{W})^{\mathbb{I}[t_n = i]} \\
&= \sum_{n=1}^{N} \sum_{i=1}^{K} \mathbb{I}[t_n = i] \log p(t_n = i|x_n, \mathbf{W}) \\
&= \sum_{n=1}^{N} \sum_{i=1}^{K} \mathbb{I}[t_n = i] \log \left(\frac{\exp(\mathbf{w}_{(i)}^T x_n)}{\sum_{j=1}^{K} \exp(\mathbf{w}_{(j)}^T x_n)} \right)
\end{aligned}$$

Notice that there is a log of a sum of exponentials in the denominator, which can be computed stably with `logsumexp` function calls in machine learning libraries.
If we instead represent the event $t_n = i$ as a one-hot encoding $\mathbf{t_n} \in \mathbb{R}^K$ (note the bolded font to differentiate t_n and $\mathbf{t_n}$ apart) with value of 1 in the i-th entry and 0 otherwise, then the inner sum can be rewritten as ${\mathbf{t_n}}^T \log \text{softmax}(Wx)$. Here $\log \text{softmax}(Wx)$ refers to element-wise application of log to the vector $\text{softmax}(Wx)$. Thus

$$\begin{aligned} E(W) &= -\log p(\mathcal{D}|W) \\ &= -\sum_{n=1}^{N} {\mathbf{t_n}}^T \log \text{softmax}(Wx_n) \end{aligned}$$

Similar to `logsumexp`, the `logsoftmax` function is typical enough that it has its own dedicated function call in PyTorch. As such, we shall leave the answer as is. Since this function is differentiable (i.e. $\nabla_W E(W)$ exist), we can use gradient descent on $E(W)$ to obtain a good set of weights for our problem.

7 Naïve Bayes Model

7.1 Introduction

Let $\boldsymbol{\theta}$ be all the parameters of our machine learning model. In the discriminative probabilistic model, we directly attack $p(t|\mathbf{x}, \boldsymbol{\theta})$. However, now we decide to rewrite $p(t|\mathbf{x}, \boldsymbol{\theta})$ as $\frac{p(\mathbf{x}|t,\boldsymbol{\theta})p(t|\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})}$ using Bayes' theorem and doing the math from there.

The intuitive idea about this rewrite is that in order to find the probability that an animal is a cat given its length, 70cm, and weight, 4kg, and our model's parameters ($p(t|\mathbf{x}, \boldsymbol{\theta})$), we can first find out the probability that an animal is a cat ($p(t|\boldsymbol{\theta})$), and then figure out the probability that the animal is 70cm long and weights 4kg given it is a cat and our model's parameters ($p(\mathbf{x}|t, \boldsymbol{\theta})$). We ignore the denominator as it is the same for all the animals.

This is why the model is called generative, as our model allows us to generate the features given the class and the parameters of our model. Model: $p(t, x|\theta) = \underbrace{p(x|t,\theta)}_{\text{Class Conditional Distribution}} \underbrace{p(t|\theta)}_{\text{Class Prior}}$

7.2 Derivation of Loss for Naïve Bayes

We will now derive the loss function given the following assumption on the data and model. Similar to the previous models we have seen so far, our data $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$ contains all the features and their corresponding classes. For our model, we assume $t \sim \text{Bernoulli}(\pi) = \pi^{t_n}(1-\pi)^{1-t_n}$, where π is the probability that $t = 1$. Next, we assume that if $t_n = 1$, then $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$. For example, we assume that if something is a cat, then its length follows a Gaussian distribution with mean 70cm and weight 5kg with some variance. We do the same for $t_n = 0$, i.e. $p(\mathbf{x}_n|t_n = 0) = \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$. Following the trick with Bernoulli distribution, we can combine both likelihoods as $p(\mathbf{x}_n|t_n) = \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)^{t_n}\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)^{1-t_n}$. However, $p(\mathbf{x}_n|t_n) = \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)^{t_n}\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)^{1-t_n}$ is too hard to work with, so we will use the Naïve Bayes assumption where all the features are independent from each other. In the cat example, we would be assuming that the weight of a cat is independent from its length. From this example, we can see why this is a Naïve assumption, however, in practice, it works well. Since the features are independent, we can write

$$p(\mathbf{x}|t, \boldsymbol{\theta}) = \prod_{m=1}^{M} p(x^m|t, \boldsymbol{\theta}) \tag{1}$$

$$= \prod_{m=1}^{M} \mathcal{N}(\mu_t^m, v_t^m) \tag{2}$$

where x^m is the m^{th} feature.

We will now grind the math.

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}_n, t_n|\boldsymbol{\theta}) \tag{3}$$

$$= \prod_{n=1}^{N} p(\mathbf{x}_n|t_n, \boldsymbol{\theta})\, p(t_n|\boldsymbol{\theta}) \tag{4}$$

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(\mathbf{x}_n|t_n, \boldsymbol{\theta})\, p(t_n|\boldsymbol{\theta}) \tag{5}$$

$$= \sum_{n=1}^{N} \log p(t_n|\boldsymbol{\theta}) + \sum_{n=1}^{N} \log p(\mathbf{x}_n|t_n, \boldsymbol{\theta}) \tag{6}$$

$$= \sum_{n=1}^{N} \log \pi^{t_n} (1-\pi)^{1-t_n} + \sum_{n=1}^{N} \log \prod_{m=1}^{M} \mathcal{N}(\mu_t^m, v_t^m) \tag{7}$$

$$= \sum_{n=1}^{N} t_n \log(\pi) + (1-t_n)\log(1-\pi) + \sum_{n=1}^{N}\sum_{m=1}^{M} \log \mathcal{N}(x_n^m|\mu_1^m, v_1^m)^{t_n}\, \mathcal{N}(x_n^m|\mu_0^m, v_0^m)^{1-t_n} \tag{8}$$

$$= \sum_{n=1}^{N} t_n \log(\pi) + (1-t_n)\log(1-\pi) + \sum_{n=1}^{N}\sum_{m=1}^{M} t_n \log \mathcal{N}(x_n^m|\mu_1^m, v_1^m) + (1-t_n)\log \mathcal{N}(x_n^m|\mu_0^m, v_0^m) \tag{9}$$

$$= \sum_{n=1}^{N} t_n \left[\log(\pi) + \sum_{m=1}^{M} \log \mathcal{N}(x_n^m|\mu_1^m, v_1^m)\right] + \sum_{n=1}^{N}(1-t_n)\left[\log(1-\pi) + \sum_{m=1}^{M} \log \mathcal{N}(x_n^m|\mu_0^m, v_0^m)\right] \tag{10}$$

where x_i^m is the i^{th} value of the m^{th} feature.
We want to maximise $p(\mathcal{D}|\boldsymbol{\theta})$ over $\boldsymbol{\theta}$ which is the same as minimising $-\log p(\mathcal{D}|\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$. Now we need to find $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$. As $\boldsymbol{\theta}$ contains the parameters $\pi, \mu_1^1, \ldots, \mu_1^M, \mu_0^1, \ldots, \mu_0^M\, v_1^1, \ldots, v_1^M, v_0^1, \ldots, v_0^M$, to find the gradient we will find the partial derivatives with respect to each of these parameters.

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \pi} = \frac{\partial}{\partial \pi}\left[-\sum_{n=1}^{N} t_n \log(\pi) - \sum_{n=1}^{N}(1-t_n)\log(1-\pi)\right] \tag{11}$$

$$= -\sum_{n=1}^{N} \frac{t_n}{\pi} + \sum_{n=1}^{N} \frac{1-t_n}{1-\pi} \tag{12}$$

$$= -\frac{1}{\pi}\underbrace{\sum_{n=1}^{N} t_n}_{N_1} + \frac{1}{1-\pi}\underbrace{\sum_{n=1}^{N}(1-t_n)}_{N_0} \tag{13}$$

$$= -\frac{N_1}{\pi} + \frac{N_0}{1-\pi} = 0 \tag{14}$$

$$\implies \frac{N_1}{\pi} = \frac{N_0}{1-\pi} \tag{15}$$

$$\implies N_1 - N_1\pi = N_0\pi \tag{16}$$

$$\implies \pi_{ML} = \frac{N_1}{N_0 + N_1} \tag{17}$$

Equation (11) comes from equation (10) and ignoring the terms without π.

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \mu_1^m} = \frac{\partial}{\partial \mu_1^m}\left[-\sum_{n=1}^{N} t_n \log \mathcal{N}(x_n^m|\mu_1^m, v_1^m)\right] \tag{18}$$

$$= \frac{\partial}{\partial \mu_1^m}\left[-\sum_{n=1}^{N} t_n \log \frac{1}{\sqrt{2\pi v_1^m}} e^{-\frac{(x_n^m-\mu_1^m)^2}{2v_1^m}}\right] \tag{19}$$

$$= \frac{\partial}{\partial \mu_1^m}\left[-\sum_{n=1}^{N} t_n \cancel{\log \frac{1}{\sqrt{2\pi v_1^m}}} - t_n \frac{(x_n^m-\mu_1^m)^2}{2v_1^m}\right] \tag{20}$$

$$= -\sum_{n=1}^{N} \frac{t_n}{v_1^m}(x_n^m - \mu_1^m) = 0 \tag{21}$$

$$\implies \sum_{n=1}^{N} t_n x_n^m = \sum_{n=1}^{N} t_n \mu_1^m \tag{22}$$

$$\implies \mu_1^m \underbrace{\sum_{n=1}^{N} t_n}_{N_1} = \sum_{n=1}^{N} t_n x_n^m \tag{23}$$

$$\implies \mu_{1,ML}^m = \frac{1}{N_1}\sum_{n=1}^{N} t_n x_n^m \tag{24}$$

It can be similarly shown that $\mu_{0,ML}^m = \frac{1}{N_0}\sum_{n=1}^{N}(1-t_n)x_n^m$, $v_{1,ML}^m = \frac{1}{N_1}\sum_{n=1}^{N} t_n(x_n^m - \mu_{1,ML}^m)^2$ and $v_{0,ML}^m = \frac{1}{N_0}\sum_{n=1}^{N}(1-t_n)(x_n^m - \mu_{0,ML}^m)^2$.
$\mu_{t,ML}^m$ is the mean of feature m of class t in our data, and $v_{t,ML}^m$ is the variance of feature m of class t in our data. Using our cat example, $\mu_{1,ML}^1$ will be the mean length of cats in the data, $\mu_{1,ML}^2$ will be the mean weight of cats, $v_{1,ML}^1$ the variance in the length and $v_{1,ML}^2$ the variance in the weight.

7.3 Prediction

Given that we already have θ_{ML}, the Maximum Likelihood estimate of the parameters, let's consider how to predict t^* for a new input x^*.

$$p(t|x^*, \theta_{ML}) \propto p(x^*|t, \theta_{ML})p(t|\theta_{ML}) = [\prod_{m=1}^{M} N(x^*|\mu_{t,MLE}^m, v_{t,MLE}^m)]Bern[t|\pi_{MLE}]$$

The conditional probability of $t^* = 1$ given x^* and θ_{ML} is given by:

$$p(t^* = 1|x^*, \theta_{\text{ML}}) = \frac{p(t^* = 1, x^*|\theta_{\text{ML}})}{p(x^*|\theta_{\text{ML}})}$$

$$= \frac{p(t^* = 1, x^*|\theta_{\text{ML}})}{p(t^* = 1, x^*|\theta_{\text{ML}}) + p(t^* = 0, x^*|\theta_{\text{ML}})}$$

For $p(t^* = 1, x^*|\theta_{\text{ML}})$, we can express it as:

$$p(t^* = 1, x^*|\theta_{\text{ML}}) = p(t^* = 1|\theta_{\text{ML}})p(x^*|t^* = 1, \theta_{\text{ML}})$$

$$= \pi_{\text{ML}} \cdot N(x^*|\mu_{1,\text{ML}}, \Sigma_{1,\text{ML}})$$

Under the Naive Bayes assumption, we have:

$$N(x^*|\mu_{1,\text{ML}}, \Sigma_{1,\text{ML}}) = \prod_{m=1}^{M} N(x^{*,m}|\mu_{1,\text{ML}}^m, v_{1,\text{ML}}^m)$$

7.4 Extension 1: Multiclass Classification

In this extension, we consider a classification problem with K classes. Each t_n is represented as a one-hot encoded vector. **Example:** If the n^{th} data point belongs to class 2, then: $t_n = [0, 1, 0, \ldots, 0]$
In general,

$$t_n = [t_{n1}, t_{n2}, \ldots, t_{nk}]$$

Model: The probability of t_n given θ is modeled by a categorical distribution Cat[$\boldsymbol{\pi}$], where $\boldsymbol{\pi}$ is in $\mathbb{R}^K$:

$$\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_k]^T$$

Subject to:

$$\sum_{k=1}^{K} \pi_k = 1, \quad 0 \leq \pi_k \leq 1$$

Thus,

$$p(t_n|\theta) = Cat[\pi] = \prod_{k=1}^{K} \pi_k^{t_{nk}} = \pi_k \text{ where } t_{nk} = 1$$

For $p(x_n|t_n, \theta)$, we assume K Gaussian distributions, one for each class. Thus, we have $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$ and $\Sigma_1, \Sigma_2, \ldots, \Sigma_k$.
The negative log-likelihood for all t_n is given by:

$$-\log \prod_{n=1}^{N} p(t_n|\theta) = -\sum_{n=1}^{N} \log p(t_n|\theta) = -\sum_{n=1}^{N} \log \prod_{k=1}^{K} \pi_k^{t_{nk}} = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \pi_k$$

Constrained Optimisation: $\tilde{L}(\pi_k, \lambda) = f(x) + \lambda g(x)$, $f(x)$ is negative log-likelihood, $g(x) = \sum_{k=1}^{K} (\pi_k - 1)$ as by definition $g(x) = 0 \Rightarrow$ Solve $\nabla_x \tilde{L} = 0 \Rightarrow \therefore \pi_k = \frac{1}{\lambda} \sum_{n=1}^{N} t_{nk}$, $\lambda = N$

7.5 Extension 2: Non-Gaussian Features

In this extension, we relax the assumption that the features have to follow a Gaussian distribution. Given that the second term of $p(t_n|\theta)p(x_n|t_n, \theta)$ can be written as:

$$p(x_n, t_n, \theta) = \prod_{m=1}^{M} p(x_n^m|t_n, \theta)$$

the distribution $p(x_n^m|t_n, \theta)$ does not necessarily have to be Gaussian.
For example, if the feature represents the number of rooms, we can use categorical data instead of assuming a Gaussian distribution.

7.6 Extension 3: Missing Data

As we assume independent given class t,

$$p(x|t, \theta) = \prod_{m=1}^{M} N(x^m|\mu_t^m, v_t^m)$$

If feature is missing, just drop the product

$$p(x_{observed}|t, \theta) = \prod_{m \in_{observed}} p(x^m|t, \theta)$$

8 Connection between Naive Bayes and Logistic Regression (Tut3)

Multiclass logistic regression:

$$p(t=j|x) = \frac{\exp(w_j^\top x)}{\sum_{i=1}^K \exp(w_i^\top x)}$$

Case: K=2

$$p(t=0|x) = \frac{\exp(w^\top x)}{1+\exp(w^\top x)}, p(t=1|x) = \frac{1}{1+\exp(w^\top x)}$$

where $w = w_{(0)} - w_{(1)}$

Let us apply Bayes Rule to $p(t=1|\boldsymbol{x})$ to introduce the generative term $p(\boldsymbol{x}|t=1)$:

$$\begin{aligned}
p(t=1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|t=1)p(t=1)}{\sum_t p(\boldsymbol{x},t)} && (25)\\
&= \frac{p(\boldsymbol{x}|t=1)p(t=1)}{p(\boldsymbol{x}|t=0)p(t=0)+p(\boldsymbol{x}|t=1)p(t=1)} && (26)\\
&= \frac{1}{1+\frac{p(\boldsymbol{x}|t=0)p(t=0)}{p(\boldsymbol{x}|t=1)p(t=1)}} && (27)\\
&= \frac{1}{1+\exp\left(\log\frac{p(t=0)}{p(t=1)}+\log\frac{p(x|t=0)}{p(x|t=1)}\right)} && (28)
\end{aligned}$$

Let $\boldsymbol{x} \in \mathbb{R}^M$. We assume the setup in naive Bayes where

$$p(\boldsymbol{x}|t=j) = \prod_{m=1}^M \mathcal{N}(x_m|\mu_{(j)}^m, v^m)$$

that is, the features of $\boldsymbol{x}$ are i.i.d. The two classes have distinct means $\{\mu_0^m\}_{m=1}^M$ and $\{\mu_1^m\}_{m=1}^M$, but shared variances $\{v^m\}_{m=1}^M$. For the prior over t, we assume a standard Bernoulli distribution $p(t) = \pi^t(1-\pi)^{1-t}$. Let us try to simplify the denominator in the expression of Equation 1. Clearly, the first term is

$$\log\frac{p(t=0)}{p(t=1)} = \log\frac{1-\pi}{\pi}.$$

Then, the second term is

$$\begin{aligned}
\log\frac{p(x|t=0)}{p(x|t=1)} &= \sum_i \log\frac{\exp\left(-\frac{(x^i-\mu_0^i)^2}{2v^i}\right)}{\exp\left(-\frac{(x^i-\mu_1^i)^2}{2v^i}\right)} && (29)\\
&= \sum_i \log\exp\left(-\frac{(x^i-\mu_0^i)^2}{2v^i}+\frac{(x^i-\mu_1^i)^2}{2v^i}\right) && (30)\\
&= \sum_i \left(-\frac{(x^i-\mu_0^i)^2}{2v^i}+\frac{(x^i-\mu_1^i)^2}{2v^i}\right) && (31)\\
&= \sum_i \left(\frac{\mu_0^i-\mu_1^i}{v^i}x^i+\frac{(\mu_1^i)^2-(\mu_0^i)^2}{2v^i}\right) && (32)
\end{aligned}$$

We simply compare terms based on the order of x^i between

$$p_{\mathrm{LR}}(t=1|\boldsymbol{x}) = \frac{1}{1+\exp(w^0 + \sum_{i=1}^{D} w^i \boldsymbol{x}^i)}$$

and

$$p_{\mathrm{NB}}(t=1|\boldsymbol{x}) = \frac{1}{1+\exp\left(\log\frac{1-\pi}{\pi} + \sum_i \frac{(\mu_1^i)^2-(\mu_0^i)^2}{2v^i} + \sum_i \frac{\mu_0^i-\mu_1^i}{v^i}x^i\right)}$$

In other words, logistic regression and naive Bayes with Gaussian features share the same form. The weights learnt in logistic regression are related to the Gaussian parameters learnt in naive Bayes as given below.

$$w^0 = \log\frac{1-\pi}{\pi} + \sum_i \frac{(\mu_1^i)^2-(\mu_0^i)^2}{2v^i}$$

$$w^i = \frac{\mu_0^i - \mu_1^i}{v^i}$$

9 Bias Variance Decomposition

9.1 Introduction

Recall that in CS2109S, we have learnt that under-fitting of the model results in high bias, while over-fitting results in high variance. In this section, we will decompose the generalisation error, which is the loss function, into its bias and variance components and see how we can represent the bias and variance mathematically.

9.2 The Decomposition

We begin by assuming that we are using mean squared error, but we can show similar results using other types of error. So we have $\mathcal{L}(\mathbf{x}, t, \mathcal{D}) = (y(\mathbf{x}, \mathcal{D}) - t)^2$ where $y(\mathbf{x}, \mathcal{D})$ is our prediction function or model, which makes use of data and the features of the thing we want to predict. We want to find the expected value of our loss function.

$$\mathbb{E}_{p(\mathbf{x},t)\,p(\mathcal{D})}[\mathcal{L}(\mathbf{x}, t, \mathcal{D})] = \mathbb{E}[(y(\mathbf{x}, \mathcal{D}) - t)^2] \tag{33}$$

$$= \mathbb{E}[\underbrace{(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x})}_{a} + \underbrace{\bar{y}(\mathbf{x}) - t)}_{b}{}^2] \tag{34}$$

$$= \underbrace{\mathbb{E}[(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x}))^2]}_{\text{variance}} + \mathbb{E}[(\bar{y}(\mathbf{x}) - t)^2] - 2\underbrace{\mathbb{E}[(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x}))(\bar{y}(\mathbf{x}) - t)]}_{=0,\text{ will show later}} \tag{35}$$

$$= \underbrace{\mathbb{E}[(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x}))^2]}_{\text{variance}} + \mathbb{E}[(\underbrace{\bar{y}(\mathbf{x}) - \bar{t}(\mathbf{x})}_{a} + \underbrace{\bar{t}(\mathbf{x}) - t)}_{b}{}^2] \tag{36}$$

$$= \underbrace{\mathbb{E}[(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x}))^2]}_{\text{variance}} + \underbrace{\mathbb{E}[(\bar{y}(\mathbf{x}) - \bar{t}(\mathbf{x}))^2]}_{\text{bias}} + \underbrace{\mathbb{E}[(\bar{t}(\mathbf{x}) - t)^2]}_{\text{noise}}$$
$$+ 2\underbrace{\mathbb{E}[(\bar{y}(\mathbf{x}) - \bar{t}(\mathbf{x}))(\bar{t}(\mathbf{x}) - t)]}_{=0,\text{ will not show later}} \tag{37}$$

$$= \underbrace{\mathbb{E}_{\mathbf{x},\mathcal{D}}[(y(\mathbf{x}, \mathcal{D}) - \bar{y}(\mathbf{x}))^2]}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}[(\bar{y}(\mathbf{x}) - \bar{t}(\mathbf{x}))^2]}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathbf{x},\mathbf{t}}[(\bar{t}(\mathbf{x}) - t)^2]}_{\text{noise}} \tag{38}$$

For equation (33), we are finding the expectation over all possible data sets, the features $\mathbf{x}$ and measured output values t. Equation (35) and (37) come from expanding $(a + b)^2 = a^2 + 2ab + b^2$ with the a and b in equations (34) and (36) respectively.
$\bar{y}(x)$ is the expected value of $y(\mathbf{x}, \mathcal{D})$ over all possible data $\mathcal{D}$, and $\bar{t}(\mathbf{x})$ is the expected value of t for a specific set of features $\mathbf{x}$.
The variance is the mean squared difference between the model's output given a fixed data set and the expected model output over all data sets. This corresponds to the variance we have seen in CS2109S as over-fitting results in our model predicting wildly different results depending on the data we are given. For bias, we have the mean squared difference between the average prediction of our model over all data sets and the average measured actual output which we need to predict. A model that under-fits would result in high bias as the average model is unable to predict the results well. Finally, noise is the mean squared difference between the expected actual output and the output we measure, a component which we are unable to reduce.

We now will show the result which we promised to show in (59).

$$\mathbb{E}_{p(\mathbf{x},t)\,p(\mathcal{D})}[(y(\mathbf{x},\mathcal{D})-\bar{y}(\mathbf{x}))(\bar{y}(\mathbf{x})-t)] = \mathbb{E}_{p(\mathbf{x},t)}\mathbb{E}_{p(\mathcal{D})}[(y(\mathbf{x},\mathcal{D})-\bar{y}(\mathbf{x}))(\bar{y}(\mathbf{x})-t)] \tag{39}$$
$$= \mathbb{E}_{p(\mathbf{x},t)}\ [\ \mathbb{E}_{p(\mathcal{D})}[(y(\mathbf{x},\mathcal{D})-\bar{y}(\mathbf{x}))]\ (\bar{y}(\mathbf{x})-t)] \tag{40}$$
$$= \mathbb{E}_{p(\mathbf{x},t)}\ [(\underbrace{\mathbb{E}_{p(\mathcal{D})}[y(\mathbf{x},\mathcal{D})]}_{\bar{y}(\mathbf{x})} - \bar{y}(\mathbf{x}))\ (\bar{y}(\mathbf{x})-t)] \tag{41}$$
$$= \mathbb{E}_{p(\mathbf{x},t)}[(0)\ (\bar{y}(\mathbf{x})-t)] \tag{42}$$
$$= 0 \tag{43}$$

In equation (39) we split the expectation over $p(\mathbf{x},t)$ and $p(\mathcal{D})$ to be the expectation over $p(\mathcal{D})$ first followed by expectation over $p(\mathbf{x},t)$. In equations (40) and (41) we bring in the expectation over $p(\mathcal{D})$ as no other terms depend on $p(\mathcal{D})$.
We now do a little lying and show the result we promised not to show later. First we notice that $\mathbb{E}_{p(\mathbf{x},t)}[t] = \mathbb{E}_{p(\mathbf{x})}[\bar{t}(\mathbf{x})]$ by the definition of $\bar{t}(\mathbf{x})$.

$$\mathbb{E}_{p(\mathbf{x},t)}[(\bar{y}(\mathbf{x})-\bar{t}(\mathbf{x}))(\bar{t}(\mathbf{x})-t)] = \mathbb{E}_{p(\mathbf{x},t)}[(\bar{y}(\mathbf{x})-\bar{t}(\mathbf{x}))(\bar{t}(\mathbf{x})-\bar{t}(\mathbf{x}))] \tag{44}$$
$$= \mathbb{E}_{p(\mathbf{x},t)}[(\bar{y}(\mathbf{x})-\bar{t}(\mathbf{x}))(0)] \tag{45}$$
$$= 0 \tag{46}$$

- The expected value of t given $\mathbf{x}$ is defined as:

$$\bar{t}(\mathbf{x}) = \int t\, p(t|\mathbf{x})\, dt = \mathbb{E}_{p(t|\mathbf{x})}[t]$$

- Similarly, the expected value of $y(\mathbf{x},\mathcal{D})$ over all possible datasets $\mathcal{D}$ is:

$$\bar{y}(\mathbf{x}) = \int y(\mathbf{x},\mathcal{D})\, p(\mathcal{D})\, d\mathcal{D} = \mathbb{E}_{p(\mathcal{D})}[y(\mathbf{x},\mathcal{D})]$$

10 PAC

10.1 Introduction

The PAC framework is a theoretical model that defines conditions under which learning is feasible. In the PAC setting, the goal is to find a hypothesis that is approximately correct (within some error ϵ) with high probability (at least $1-\delta$), assuming that the hypothesis space $\mathcal{H}$ is finite.

10.2 Is Learning Probable?

In the PAC learning framework, one of the key questions we ask is whether learning is probable. Specifically, we want to bound the probability of a "bad event" where the error of the hypothesis selected by the learner deviates significantly from the true error.

$$P(\text{Bad event}) \leq \delta \tag{47}$$

10.3 Hoeffding's inequality

Suppose we are working with a dataset $D = \{z_1, z_2, \ldots, z_N\}$ of size N, where each z_i is drawn from a Bernoulli distribution with unknown parameter μ, i.e., $z_i \sim \text{Bern}(\mu)$
The goal is to estimate μ using the empirical mean $\hat{\mu}_N$, which is the average of the observed data. We want to know how close $\hat{\mu}_N$ is to the true parameter μ. Specifically, we want to bound the probability that the absolute difference between $\hat{\mu}_N$ and μ is larger than a given tolerance ϵ:

$$P\left(|\hat{\mu}_N - \mu| > \epsilon\right) \tag{48}$$

To bound this probability, we can use Hoeffding's inequality, which provides a concentration bound for the sum of independent random variables. Hoeffding's inequality states that:

$$P\left(|\hat{\mu}_N - \mu| > \epsilon\right) \leq 2\exp\left(-2N\epsilon^2\right) \tag{49}$$

This inequality tells us that the probability of the empirical mean deviating from the true mean by more than ϵ decreases exponentially with N, the number of samples, and ϵ^2.

- As N increases, the probability of a large deviation decreases (i.e., learning becomes more probable as we get more data).
- As ϵ decreases, the probability of a large deviation increases, as we require the estimate to be more precise.

10.4 Developing a Bound for a Given Hypothesis y

We aim to bound the probability of a "bad event" where the empirical error $L_D(y)$ deviates from the true error $L_p(y)$ by more than ϵ. Formally, we want to ensure that:

$$P\left(|L_D(y) - L_p(y)| > \epsilon\right) < \delta_y \tag{50}$$

where δ_y is the failure probability.

- True error $L_p(y)$ is the proportion of incorrect classifications over the entire distribution.
- Dataset error $L_D(y)$ is the observed error on the dataset D.

The true goal is to minimize the difference between these two quantities. To do this, we use Hoeffding's inequality to bound the probability of large deviations. Using Hoeffding's inequality, we bound the probability that the empirical error $L_D(y)$ deviates from the true error $L_p(y)$ by more than ϵ:

$$P\left(|L_D(y) - L_p(y)| > \epsilon\right) \leq 2\exp\left(-2N\epsilon^2\right) \tag{51}$$

As the number of samples N increases, the probability δ_y decreases. Conversely, as ϵ decreases (i.e., we require more accuracy), δ_y increases.

10.5 Developing a Bound for All Hypotheses y in the Entire Hypothesis Space $\mathcal{H}$

The above process is only good for verification, i.e., finding the bound for a given y. This is not what we do at the start of the machine learning process, where we instead seek to show that:
For the problem we are trying to learn, the hypothesis y that we will eventually obtain will be bounded by ϵ with probability at least $1 - \delta$.
Our learning algorithm selects the hypothesis with the lowest $L_D(y)$. However, for simplicity, we reduce the problem to finding the probability that at least 1 hypothesis in $\mathcal{H}$ satisfies

$$\left|L_D(y) - L_p(y)\right| > \epsilon.$$

Hence, we want to bound the probability that any hypothesis $y \in \mathcal{H}$ results in a large deviation between the empirical and true error. The problem can therefore be formulated as finding a bound for the following probability:

$$P\left(\exists y \in \mathcal{H}, |L_D(y) - L_p(y)| > \epsilon\right) \tag{52}$$

Using the union bound, we can decompose this probability over the set of hypotheses:

$$P\left(\exists y \in \mathcal{H}, |L_D(y) - L_p(y)| > \epsilon\right) \leq \sum_{i=1}^{M_{\mathcal{H}}} P\left(|L_D(y_i) - L_p(y_i)| > \epsilon\right) \tag{53}$$

Each of these probabilities can be bounded using Hoeffding's inequality:

$$P\left(|L_D(y_i) - L_p(y_i)| > \epsilon\right) \leq 2\exp\left(-2N\epsilon^2\right) \tag{54}$$

Thus, the overall bound becomes:

$$P\left(\exists y \in \mathcal{H}, |L_D(y) - L_p(y)| > \epsilon\right) \leq M_{\mathcal{H}} \cdot 2\exp\left(-2N\epsilon^2\right) \tag{55}$$

To ensure that the probability of a bad event is smaller than δ, we need the number of samples N to satisfy:

$$N \geq \frac{1}{2\epsilon^2} \ln\left(\frac{2M_{\mathcal{H}}}{\delta}\right) \tag{56}$$

Thus, if we choose N according to this inequality, the probability that the empirical error deviates from the true error by more than ϵ will be less than δ.
Note that as the number of hypotheses in the hypothesis class $M_{\mathcal{H}}$ increases, the probability of finding a hypothesis with low $L_D(y)$ increases, but $L_p(y)$ remains the same. Thus, a larger N is required to maintain the same bound. This is related to overfitting.

10.6 Limitations

One limitation of this conceptual tool is that it is built upon the assumption that the hypotheses class is finite in size, which is not necessarily true. Another is that the PAC bound is very loose because no assumptions are made about x or y. This makes it more general, but can make the required N far higher than actually needed in practice.

11 Neural Networks

11.1 Neural Networks as Adaptive Basis Functions

Consider a multi-layer perceptron (MLP) with a single hidden layer with a non-linear activation function h. Suppose our data is in the form of $\{(\mathbf{x}, t)_n\}_N$. Then the neural network can be written as $y = W_2^T\, h(W_1^T \mathbf{x})$, where W_1 are the weights to obtain the hidden layer, and W_2 are the weights to obtain the output. ff Notice that this form is similar to linear regression, which has the form of $y = W^T \phi(\mathbf{x})$. In fact, $h(W_1^T \mathbf{x})$ can be thought of as an adaptive basis function, where the weights W_1 are the parameters, θ, of the basis function, $\phi_\theta(\mathbf{x})$.

11.2 Backpropagation

Backpropagation is a technique to optimise the weights of neural networks through a series of local computations. The principle idea is that $\frac{\delta \mathcal{L}}{\delta w_{kj}} = \delta_k Z_j$ where δ_k refers to the error signal at the tail neuron and z_j refers to the input at the head neuron . Suppose there is a hidden layer a_j such that $z_j = h(a_j)$ and $a_j = \sum_{i=1} w_{ji}.z_i$, then by using chain rule,

$$\begin{aligned}\frac{\delta \ell}{\delta w_{ji}} &= \frac{\delta \ell}{\delta a_j}\frac{\delta a_j}{\delta w_{ji}} \\ &= \sum_k \frac{\delta \ell}{\delta a_k}\frac{\delta a_k}{\delta a_j} z_i \\ &= \sum_k \delta_k w_{kj} h^{'}(a_j) z_i\end{aligned}$$

where δ_k refers to the error signal for the $k'th$ head attached to a_j. This process results in an efficient process for weight updates ad δ_k and z_j can be reused

11.3 Gradient Descent

Recall the algorithm for gradient descent.

Algorithm 1 Gradient Descent

1: Initialize $\mathbf{w}_0$
2: **while** termination condition not met **do**
3: $\quad \mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta_k \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_k)$

We will now show why gradient descent works. Assume our loss function $\mathcal{L}(\mathbf{w})$ is L-smooth. Then, by definition, we have

$$\frac{\|\nabla \mathcal{L}(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{v})\|}{\|\mathbf{w} - \mathbf{v}\|} \leq L,$$

which is equivalent to (proof omitted, just trust prof)

$$|\mathbf{u}^T \nabla^2 \mathcal{L}(\mathbf{w}) \mathbf{u}| \leq L\|\mathbf{u}\|,$$

where $\nabla^2 \mathcal{L}(\mathbf{w})$ is the Hessian matrix and

$$\nabla^2 \mathcal{L}(\mathbf{w}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1^2} & \cdots & \frac{\partial \mathcal{L}}{\partial w_1 w_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial w_d w_1} & \cdots & \frac{\partial \mathcal{L}}{\partial w_d^2} \end{bmatrix}$$

where d is the length of the $\mathbf{w}$ vector.
Next, we have the Taylor's theorem, which states that $\exists \xi \in \mathbb{R}$ such that

$$g(x) = g(x_0) + g'(x_0)(x - x_0) + \frac{g''(\xi)}{2!}(x - x_0)^2$$

or

$$g(\mathbf{x}) = g(\mathbf{x}_0) + g'(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \nabla^2 g(\xi)\ (\mathbf{x} - \mathbf{x}_0)$$

for vectors.

Assuming a constant learning rate η,

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{k+1}) &= \mathcal{L}(\mathbf{w}_k) + \nabla\mathcal{L}(\mathbf{w}_k)^T(\mathbf{w}_{k+1} - \mathbf{w}_k) + \frac{1}{2}(\mathbf{w}_{k+1} - \mathbf{w}_k)^T\nabla^2\mathcal{L}(\xi)(\mathbf{w}_{k+1} - \mathbf{w}_k) \\ &= \mathcal{L}(\mathbf{w}_k) + \nabla\mathcal{L}(\mathbf{w}_k)^T(-\eta\nabla\mathcal{L}(\mathbf{w}_k)) + \frac{1}{2}(-\eta\nabla\mathcal{L}(\mathbf{w}_k))^T\nabla^2\mathcal{L}(\xi)(-\eta\nabla\mathcal{L}(\mathbf{w}_k)) \\ &= \mathcal{L}(\mathbf{w}_k) - \eta\nabla\mathcal{L}(\mathbf{w}_k)^T\nabla\mathcal{L}(\mathbf{w}_k) + \frac{1}{2}\eta^2\nabla\mathcal{L}(\mathbf{w}_k)^T\nabla^2\mathcal{L}(\xi)\nabla\mathcal{L}(\mathbf{w}_k) \\ &\leq \mathcal{L}(\mathbf{w}_k) - \eta\|\nabla\mathcal{L}(\mathbf{w}_k)\|^2 + \frac{\eta^2 L}{2}\|\nabla\mathcal{L}(\mathbf{w}_k)\|^2\end{aligned}$$

The first line uses Taylor's theorem, the second line uses definition of gradient descent, and the fourth line uses the second definition of L-smooth.
Choosing η such that $\eta L \leq 1$, we can show that

$$\mathcal{L}(\mathbf{w}_k) - \mathcal{L}(\mathbf{w}_{k+1}) \geq \frac{1}{2}\eta\|\mathcal{L}(\mathbf{w}_k)\|^2.$$

Furthermore, we can show that

$$\min_{k\in\{0,1,\ldots,K\}}\|\nabla\mathcal{L}(\mathbf{w}_k)\|^2 \leq \frac{2(\mathcal{L}(\mathbf{w}_k) - \mathcal{L}^*)}{\eta K},$$

so as K increases, the gradient decreases, and the loss will converge.

11.4 Stochastic Gradient Descent

By further assuming that $\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w})\|^2] - \|\nabla\mathcal{L}(\mathbf{w})\|^2 < \sigma^2$, or the variance of gradient samples is bounded, we can show

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w})\|^2] \leq \frac{2(\mathcal{L}(\mathbf{w}_k) - \mathcal{L}^*)}{\eta K} + \underbrace{\frac{\eta\sigma^2 L}{2}}_{\text{“noise ball”}}.$$

This shows that there will be a region which the stochastic gradient descent will converge to, rather than a point.

11.5 Tutorial 6 (Gradient Descent and Stochastic Gradient Descent)

11.5.1 Gradient Descent

Assume that $f''(w) < L$ by some constant $L > 0$. Recall that Taylor's Theorem states that for any function $g : \mathbb{R} \to \mathbb{R}$ that satisfy some conditions, there exist $\varepsilon \in \mathbb{R}$ such that

$$h(x) = h(x_0) + h'(x_0)(x - x_0) + \frac{h''(\varepsilon)}{2}(x - x_0)^2$$

Hence, from Taylor's Theorem we have:

$$\begin{aligned}f(w_{t+1}) &= f(w_t - \alpha f'(w_t)) \\ &= f(w_t) - \alpha f'(w_t)^2 + \frac{f''(\varepsilon)^2}{2}(\alpha f'(w_t))^2 \\ &\leq f(w_t) - \alpha f'(w_t)^2 + \frac{\alpha^2 L}{2}f'(w_t)^2 \\ &= f(w_t) - \alpha(1 - \frac{\alpha L}{2})f'(w_t)^2\end{aligned}$$

Rearranging the equation and suppose $\alpha < 1/L$,

$$\begin{aligned} f(w_t) - f(w_{t+1}) &\geq \alpha(1 - \frac{\alpha L}{2}) f'(w_t)^2 \\ &> \alpha \left(1 - \frac{(1/L) \cdot L}{2}\right) f'(w_t)^2 \\ &= \frac{\alpha f'(w_t)^2}{2} \end{aligned}$$

Rearranging the equation we have $f'(w_t)^2 \leq \frac{2}{\alpha}(f(w_t) - f(w_{t+1}))$

Let $f* = \min_w f(w)$ be the optimal value of f

$$\begin{aligned} \frac{\alpha}{2} \sum_{t=0}^{T-1} f'(w_t)^2 &\leq \frac{\alpha}{2} \sum_{t=0}^{T-1} \frac{2}{\alpha}(f(w_t) - f(w_{t+1})) \\ &= \sum_{t=0}^{T-1} f(w_t) - f(w_{t+1}) \\ &= f(w_0) - f(w_T) \\ &\leq f(w_0) - f* \end{aligned}$$

Observe that we have

$$\begin{aligned} \min_{t \in \{0,\ldots,T-1\}} f'(w_t)^2 &\leq \frac{1}{T} \sum_{t=0}^{T-1} f'(w_t)^2 \\ &\leq \frac{2(f(w_0) - f*)}{\alpha T} \end{aligned}$$

11.5.2 Stochastic Gradient Descent

Now, we'll do the same for Stochastic Gradient Descent. At each step t, we update the weights using the formula $w_{t+1} = w_t - \alpha g_t$ where g_t is a stochastic gradient (i.e. a random variable) satisfying $\mathbb{E}[g_t|w_t] = f'(w_t)$ and $\mathbb{E}[g_t^2|w_t] \leq \sigma^2$ for some $\sigma^2 > 0$

Using the update equation and Taylor's theorem, we have:

$$f(w_{t+1}) = f(w_t - \alpha g_t) \leq f(w_t) - \alpha f'(w_t) g_t + \frac{\alpha^2 g_t^2 L}{2}$$

Taking conditional expectation w.r.t w_t on both sides, we have

$$\begin{aligned} \mathbb{E}[f(w_{t+1})|w_t] &\leq f(w_t) - \alpha f'(w_t)\mathbb{E}[g_t|w_t] + \frac{\alpha^2 \mathbb{E}[g_t^2|w_t] L}{2} \\ &\leq f(w_t) - \alpha f'(w_t)^2 + \frac{\alpha^2 \sigma^2 L}{2} \end{aligned}$$

Rearranging, we have

$$\begin{aligned} \mathbb{E}[f(w_t) - f(w_{t+1})] &= \mathbb{E}[\mathbb{E}[f(w_t) - f(w_{t+1})|w_t]] \\ &\geq \mathbb{E}\left[f(w_t) - f(w_t) + \alpha f'(w_t)^2 - \frac{\alpha^2 \sigma^2 L}{2}\right] \\ &= \alpha \mathbb{E}[f'(w_t)] - \frac{\alpha^2 \sigma^2 L}{2} \end{aligned}$$

Summing over both sides, we have

$$\begin{aligned}\alpha \sum_{t=0}^{T-1} \mathbb{E}[f'(w_t)^2] &\leq \sum_{t=0}^{T-1} \left(\mathbb{E}[f(w_t) - f(w_{t+1})] + \frac{\alpha^2 \sigma^2 L}{2} \right) \\ &= \sum_{t=0}^{T-1} (\mathbb{E}[f(w_t)] - \mathbb{E}[f(w_{t+1})]) + \frac{\alpha^2 \sigma^2 LT}{2} \\ &\leq f(w_0) - f* + \frac{\alpha^2 \sigma^2 LT}{2}\end{aligned}$$

assuming we choose w_0 deterministically. As such, we have

$$\min_{t \in \{0,\ldots,T-1\}} \mathbb{E}[f'(w_t)^2] \leq \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[f'(w_t)^2] \leq \frac{f(w_0) - f*}{\alpha T} + \frac{\alpha \sigma^2 L}{2}$$

11.6 Convolution

Convolutions are sparse, if we convert a convolution into a fully connected linear layer, most weights will be 0. There is also parameter tying or weight sharing, where many weights in the equivalent fully connected linear layer would share the same weights. Finally, convolutions are equivariant to translations but not rotations. A function $f(x)$ is equivariant to another function $g(x)$ if $f(g(x)) = g(f(x))$.
Convolutions are usually accompanied by max-pooling, which makes feature detection more robust to minor perturbations.

11.7 Transformers

RNNs have the following advantages:

1. Handles variable-length sequences
2. Maintains order information
3. Scalable due to shared parameters

However, they also suffer from the following disadvantages:

1. Exploding and Vanishing Gradients
2. Memory Encoding Bottleneck (where memory from older words are "forgotten")
3. Iterative update is slow as it cannot be parallelized

These issues can be solved using transformers. The following are the steps for the transformer layer.

1. **Compute positional embeddings.** Each word is given an embedding, and the positional embedding of each word is its embedding + a number to encode its position.
2. **Compute Query, Key and Value matrices.** Using the positional embeddings E and learned weights W_Q, W_K, W_V, we compute the query matrix $Q = E \times W_Q$, key matrix $K = E \times W_K$ and value matrix $V = E \times W_V$.
3. **Compute attention weights.** The attention weights $S = \text{softmax}\left(\frac{QK^T}{d_S}\right)$ is computed, where d_S is the dimension of the output.
4. **Extract features based on attention.** The output of the transformer layer would be $O = S \times V$.

12 Kernels

Kernel functions are ways to implicitly compute the similarity between two feature spaces. This trick can be leveraged in dual representations to define non parametric models.

12.1 Non-parametric Linear Regression

Consider $y(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$ and $\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_n \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) - t_n\right)^2 + \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w}$

Setting $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = 0$

$$\begin{aligned}\mathbf{w} &= \sum_n \left(-\frac{1}{\lambda}\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) - t_n\right)\boldsymbol{\phi}_n\right) \\ &= \sum_n a_n \boldsymbol{\phi}_n = \boldsymbol{\Phi}^\top \mathbf{a}\end{aligned}$$

Then, the model can be rewritten as,

$$\begin{aligned}y &= \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) \\ &= \left(\boldsymbol{\Phi}^\top \mathbf{a}\right)^\top \boldsymbol{\phi}(\mathbf{x}) \\ &= \mathbf{a}^\top \boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) \\ &= \sum_n a_n \boldsymbol{\phi}(\mathbf{x_n})^\top \boldsymbol{\phi}(\mathbf{x})\end{aligned}$$

Similarly, the loss function can also be written in terms of $\mathbf{a}$ and $\boldsymbol{K} = \boldsymbol{\phi}\boldsymbol{\phi}^\top$ by replacing $\mathbf{w}$ with $\boldsymbol{\Phi}^\top \mathbf{a}$ resulting in,

$$\mathcal{L}(\mathbf{a}) = \frac{1}{2}\mathbf{a}^\top \mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^\top \mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^\top \mathbf{t} + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{K}\mathbf{a}$$

12.2 Valid Kernels

For a Kernel Matrix $\mathbf{K}$ to be valid, it must be symmetric and positive semi-definite. Therefore,

$$\begin{aligned}&k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x_j}) = \boldsymbol{\phi}(\mathbf{x}_j)^\top \boldsymbol{\phi}(\mathbf{x}_i) \\ &\mathbf{v}^\top \mathbf{K}\mathbf{v} \geq 0 \quad \text{for all } \mathbf{v} \in \mathbb{R}^N \setminus \{\mathbf{0}\}\end{aligned}$$

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= q(k_1(\mathbf{x}, \mathbf{x}')) \\
k(\mathbf{x}, \mathbf{x}') &= \exp(k_1(\mathbf{x}, \mathbf{x}')) \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_3(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')) \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)
\end{aligned}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.
Gaussian Kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^{\mathbf{2}}}{2\sigma^2}\right)$$

For a function $k(\mathbf{x}, \mathbf{x}')$ to be a kernel, there must exist some basis function $\phi(\mathbf{x})$ such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. Also note that kernels must satisfy $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}) > 0$.

12.3 Lagrangians

Suppose we have an optimization problem in its standard form

$$\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & f_0(\mathbf{x}) \\
\text{subject to} \quad & f_i(\mathbf{x}) \leq 0 \quad \text{for all } i \in \{1, 2, \ldots, m\} \\
& g_i(\mathbf{x}) = 0 \quad \text{for all } i \in \{1, 2, \ldots, n\}
\end{aligned}$$

We can then obtain the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{n} \nu_i g_i(\mathbf{x}).$$

where each of λ_i, ν_i are Lagrangian multipliers for each of our constraints above. Now consider the Lagrange dual function

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{n} \nu_i g_i(\mathbf{x}) \right)$$

The Lagrange dual function has the property that if $\lambda_i \geq 0$ for all i, then it is always a lower bound for the primal problem. Let the optimal value to the primal problem ($\min f_0(\mathbf{x})$) be p^*. We then have $\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^*$ if $\boldsymbol{\lambda} \geq \mathbf{0}$.
Therefore, instead of attempting to minimise the primal problem, we can instead maximise the Lagrangian Dual to obtain a "lower bound", thus formulating the dual problem as

$$\begin{aligned}
\underset{\boldsymbol{\lambda}, \boldsymbol{\nu}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\
\text{subject to} \quad & \lambda_i \geq 0 \quad \text{for all } i \in \{1, 2, \ldots, m\}
\end{aligned}$$

Let d^* denote the optimal value to the dual problem ($\max \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\nu})$). By the lower bound property of the Lagrange dual function, $d^* \leq p^*$. This is known as *weak duality*. Furthermore, if f_0 and f_i are convex functions and g_i are affine functions ($g_i(\mathbf{x}) = \mathbf{a}_i \cdot \mathbf{x} - b_i$), and Slater's conditions are satisfied, then $d^* = p^*$. This is known as *strong duality*.

Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are a set of conditions that a set of possible primal and dual variables $\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}$ may or may not hold.

1. Primal feasibility: $f_i(\tilde{\mathbf{x}}) \leq 0$ and $g_i(\tilde{\mathbf{x}}) = 0$.
2. Dual feasibility: $\tilde{\boldsymbol{\lambda}} \geq \mathbf{0}$.
3. Complementary slackness: $\tilde{\lambda}_i f_i(\tilde{\mathbf{x}}) = 0$.
4. Vanishing gradient: $\nabla_{\mathbf{x}} L(\mathbf{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}) = 0$ at $\mathbf{x} = \tilde{\mathbf{x}}$.

KKT and optimality. Here are two facts relating KKT and optimality of the primal and dual problems.

1. If strong duality holds and $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ is optimal, then KKT conditions must hold for $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$.
2. Let f_i be convex for all $i \in \{1, 2, \ldots, m\}$ and g_i be affine for all $i \in \{1, 2, \ldots, n\}$. If KKT conditions hold for $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$, then $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ is optimal with strong duality.

13 Tutorial 9: Multivariate Gaussians

Conditional Gaussian Distributions

Condition: Variables $\mathbf{x}_a$ and $\mathbf{x}_b$ must be jointly Gaussian.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \mathcal{N}(\boldsymbol{\mu}, \Lambda^{-1})$$

Using Precision Matrix Λ

Definition: $\Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}$

Note: Λ_{aa} is the top-left block of the inverted matrix, not $(\Sigma_{aa})^{-1}$.

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \Lambda_{aa}^{-1} \Lambda_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \\ \Sigma_{a|b} &= \Lambda_{aa}^{-1} \end{aligned}$$

Using Covariance Matrix Σ

Definition: $\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \Sigma_{ab} \Sigma_{bb}^{-1} (\mathbf{x}_b - \boldsymbol{\mu}_b) \\ \Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba} \end{aligned}$$

Univariate Case

This is the scalar application of the Standard Form.

$$\begin{aligned} \mathbb{E}[t|x] &= \mu_t + \frac{\mathrm{Cov}[t, x]}{\mathrm{Var}[x]} (x - \mu_x) \\ \mathrm{Var}[t|x] &= \mathrm{Var}[t] - \frac{\mathrm{Cov}[t, x]^2}{\mathrm{Var}[x]} \end{aligned}$$

13.1 SVM

SVMs are a type of non parametric classifier where the goal is to maximise the smallest distance between the decision boundary and any sample.

- Data: $D = \{(x_n, t_n)\}_{n=1}^{N}$, where $t_n = \pm 1$
- Model: $y(x) = w^T x + b$
- Assumption: $\forall n, t_n y(x_n) > 0$

13.2 Quadratic Program

The loss function can be easily written in the form

$$\max_{w,b} \min_{n} \frac{t_n(\mathbf{w}^\top \mathbf{x_n} + b)}{\|\mathbf{w}\|}$$

This can be shown given by the fact that r, the distance between a point and the decision boundary, can be written as follows,

$$\begin{aligned} r &= \frac{|y(x)|}{\|w\|}, t_n y(x_n) > 0 \\ &= \frac{t_n(\mathbf{w}^\top \mathbf{x_n} + b)}{\|w\|} \end{aligned}$$

However, this loss function above is hard to optimise. Instead, it can be written in the form,

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$\textbf{such that } t_n(w^T x_n + b) \geq 1 \text{ for } n = 1, ..., N$$

Step 1: Eliminate the "min"

1. Consider $w \to kw$ and $b \to kb$.
2. Set $t_n(w^T x_n + b) = 1$ for points closest to the boundary
3. $r_{\min} = 1/||w||$
4. impose the constraint that $\forall (x_n, t_n) \in D, t_n(\mathbf{w}^\top \mathbf{x_n} + b) \geq 1$

Step 2: Invert $\frac{1}{\|w\|}$: $\quad max_{w,b} \frac{1}{||w||} \equiv min_{w,b} \frac{1}{2}||w||^2$

This results in an equation that can be solved using a Quadratic solver

13.3 Applying Lagrange multipliers

The quadratic program can be written as a Lagrangian with the following form

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{n=1}^{N} \lambda_n \big(1 - t_n(\mathbf{w}^\top \mathbf{x}_n + b)\big)$$

By taking the derivative w.r.t $\mathbf{w}$ and b, and setting them to 0 we get,

$$\mathbf{w} = \sum_{n=1}^{N} t_n \lambda_n \mathbf{x}_n \text{ and } \sum_{n=1}^{N} \lambda_n t_n = 0$$

Substituting $\mathbf{w}$ into the Lagrangian results in the Lagrange dual function for the SVM primal

$$\mathcal{L}(\boldsymbol{\lambda}) = \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \begin{cases} \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{x}_n^\top \mathbf{x}_m & \text{if } \sum_{n=1}^{N} \lambda_n t_n = 0, \\ -\infty & \text{otherwise.} \end{cases}$$

The Lagrange dual is always a lower bound when $\lambda_i \geq 0 \ \forall i$. Hence, by maximising the dual, we can find the tighest lower bound for $p*$, the optimal solution for the primal problem.

14 K-Means Clustering

K-means clustering is an unsupervised learning algorithm that partitions data into K clusters by iteratively assigning points to the nearest centroid and updating the centroids accordingly.

- Data: $D = \{x_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}$
- Model:
  - Assume K clusters
  - Cluster centers $\{\mu_k\}_{k=1}^{K}$, $\mu_k \in \mathbb{R}^D$
  - Assignment $\{r_n\}_{n=1}^{N}$ of x_n to cluster k, 1-of-K encoding. $r_{nk} \in \{0, 1\} \forall k = 1, ..., K$
- Loss: $min_{\theta=\{r_n,\mu_k\}} J(\theta)$

Algorithm 2 K-Means algorithm

1: **Initialization** Randomly choose some initial values for $\{\mu_k\}$

2: **Assignment** Fix $\{\mu_k\}$. Optimize each r_n separately for each x_n. Assign each point to its closest cluster

$$\arg\min_{\boldsymbol{r_n}} \sum_k r_{nk} \left\| x_n - \mu_k \right\|^2 \quad \text{subject to} \quad \sum_k r_{nk} = 1$$

$$r_{nk} = \begin{cases} 1, & \text{if } k = \arg\min_j \left\| x_n - \mu_j \right\|^2, \\ 0, & \text{otherwise.} \end{cases}$$

3: **Update** Fix $\{r_{nk}\}$. Optimize each μ_k separately. Set each cluster center = average of points in the cluster

$$\arg\min_{\boldsymbol{\mu_k}} \sum_n r_{nk} ||x_n - \mu_k||^2 \quad \text{subject to} \quad \sum_k r_{nk} = 1$$

$$L_{\mu_k} = \sum_n 2 r_{nk} (x_n - \mu_k) \overset{\text{set}}{=} 0 \rightarrow \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

4: **Repeat** 2 and 3 until convergence

$$\min_{\boldsymbol{\theta=\{r_n,\mu_k\}}} \overbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\boldsymbol{x_n} - \mu_k||^2}^{J(\theta)} \text{, such that } \sum_k r_{nk} = 1$$

15 Mixture Models

Mixture Models are generative models similar to Naive Bayes, in which we find $p(t|\mathbf{x}, \boldsymbol{\theta})$ through $\frac{p(\mathbf{x}|t,\boldsymbol{\theta})p(t|\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})}$. However, in this case, each of our targets t are a probability distribution which can be Gaussian, Exponential, Binomial etc., and we do not know our labels t since we are performing unsupervised learning.
Let's assume we have K different probability distributions as our targets. We first start by introducing a K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is 1, and all other elements are 0. This means that each vector $\mathbf{z}$ has K possible states. Let us also say that $\mathbf{z} \sim \text{Cat}(\boldsymbol{\pi})$, where $\boldsymbol{\pi}$ refers to a vector of probabilities for each class. Then, we can succinctly write $p(z_n) = \prod_k \pi_k^{z_{nk}}$ due to the one-hot encoding of z_n and the notational trick which we have introduced earlier.
Now, when we first sample some x_n from the dataset, we need to first sample a cluster z_n from which we will derive x_n. Specifically, each x_n is obtained using

$$p(x_n|z_n) = \prod_k N(x_n|\mu_k, \Sigma_k)^{z_{nk}}$$

assuming the underlying distributions are Gaussian in nature. Using the product rule, we can write the joint distribution for each of our (x_n, z_n) pairs in the data as

$$\begin{aligned} p(x_n, z_n) &= p(x_n|z_n)p(z_n) \\ &= \prod_k \pi_k^{z_{nk}} N(x_n|\mu_k, \Sigma_k)^{z_{nk}}. \end{aligned}$$

To extract out x_n, we marginalise over all possible z_n to give us

$$\begin{aligned} p(x_n) &= \sum_{z_n} p(x_n, z_n) \\ &= \sum_{z_n} \prod_k \pi_k^{z_{nk}} N(x_n|\mu_k, \Sigma_k)^{z_{nk}} \\ &= \sum_k \pi_k N(x_n|\mu_k, \Sigma_k) \end{aligned}$$

where π_k are our mixing coefficients, giving rise to the "mixture" of distributions. Clearly, our model parameters $\boldsymbol{\theta} = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ because we need to have our model distributions and respective probabilities to create our mixtures. Let us attempt to learn this $\boldsymbol{\theta}$ by MLE:

$$\begin{aligned} \boldsymbol{\theta}_{MLE} &= \arg\max_{\boldsymbol{\theta}} \log p(D|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \log \prod_{n=1}^N p(x_n|\boldsymbol{\theta}) \\ \mathcal{L}(\boldsymbol{\theta}) &= \sum_n \log \left[\sum_k \pi_k N(x_n|\mu_k, \Sigma_k) \right] \end{aligned}$$

We find our ideal parameters using differentiation. First, let's handle μ_k. Skipping the derivations, we arrive at

$$\nabla_{\mu_k} \mathcal{L} = \sum_n \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_k \pi_k N(x_n|\mu_k, \Sigma_k)} \Sigma_k^{-1}(x_n - \mu_k)$$

where the responsibility term $\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_k \pi_k N(x_n|\mu_k,\Sigma_k)}$ is the probability that the point x_n comes from a particular cluster k. It is, in essence, a "soft assignment" to a particular cluster, and equivalent to

$p(z_{nk} = 1|x_n)$. Further solving the derivative, we get $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$ where $N_k = \sum_n \gamma(z_{nk})$. N_k refers to the number of points belonging to a cluster k as $z_{nk} = 1$ if the point is from k and 0 otherwise, so μ_k gives us the "weighted average" of all points belonging to cluster k.
But let's stop here: Note that our $\gamma(z_{nk})$ requires us to know some μ_k and Σ_k! This results in a chicken-and-egg problem: We cannot seem to solve for $\boldsymbol{\theta}_{MLE}$ in closed form because of these circular dependencies. To move forward, we need to make use of the EM algorithm.

16 EM Algorithm

The general EM algorithm allows us to find the probability of X, Z where Z are latent (*unseen*) variables in the dataset, with respect to a external parameter choice $\boldsymbol{\theta}$. It works as follows:

Algorithm 3 General EM algorithm

Input: Joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$
Result: Maximize the likelihood $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$
1: Initialize parameters $\hat{\boldsymbol{\theta}}$
2: **E-Step.** Compute the distribution $p(\mathbf{X}, \mathbf{Z}|\hat{\boldsymbol{\theta}})$ and formulate

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathbb{E}_{\mathbf{Z} \sim p(\mathbf{Z}|\mathbf{X}, \hat{\boldsymbol{\theta}})}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}')]$$

3: **M-Step.** Evaluate $\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}'} \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$
4: **if** $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ are close **then**
5: **return** $\boldsymbol{\theta}$
6: **else**
7: set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$
8: repeat from step 2

Let's begin by first writing the log-likelihood of the data, $\log p(X|\boldsymbol{\theta})$. We massage this term by introducing a new distribution $q(z)$ and marginalising over it to get

$$\begin{aligned}
\log p(X|\boldsymbol{\theta}) &= \sum_z q(z) \log p(X|\boldsymbol{\theta}) \\
&= \sum_z q(z) \log \left[p(X|\boldsymbol{\theta}) \frac{q(z)}{q(z)} \frac{p(Z|X, \boldsymbol{\theta})}{p(Z|X, \boldsymbol{\theta})} \right] \\
&= \sum_z q(z) \log \frac{p(X, Z|\boldsymbol{\theta})}{q(z)} + \sum_z q(z) \log \frac{q(z)}{p(Z|X, \boldsymbol{\theta})}
\end{aligned}$$

by rearranging and bringing in of the log.
We call the left term before the addition (+) the 'lower bound' or ELBO $\mathcal{L}(q, \boldsymbol{\theta})$, and the right term the KL divergence $\mathcal{D}[q||p]$. In the E-step, we fix our $\boldsymbol{\theta}$ and optimise q, and in the M-step we fix our q_{t+1} and optimise $\boldsymbol{\theta}$.
Specifically, to maximise $\mathcal{L}(q, \boldsymbol{\theta})$, we first fix $\boldsymbol{\theta}_t$ and find $q_{t+1} = \arg\max_q \mathcal{L}(q, \boldsymbol{\theta_t})$, then we fix q_{t+1} and find $\boldsymbol{\theta}_{t+1} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q_{t+1}, \boldsymbol{\theta_t})$.
For the first step, notice that $\log p(X|\boldsymbol{\theta})$ is constant with respect to q. So instead of increasing our ELBO, we can instead minimise the KL divergence (which is always positive). A property of the KL divergence is that it is minimised when the two distributions p and q are identical - so we simply set $q = p$ in this case. This gives us $q(z) = p(Z|X, \boldsymbol{\theta}_t)$, which is what we are doing in step (2) of the EM algorithm.

For the next step, we try and find our new parameters $\boldsymbol{\theta}_{t+1}$. Substituting our q with p, we get

$$\begin{aligned}\mathcal{L}(q,\boldsymbol{\theta}) &= \sum_z q(z)\log\frac{p(X,Z|\boldsymbol{\theta})}{q(z)}\\ &= \sum_z p(Z|X,\boldsymbol{\theta}_t)\log\frac{p(X,Z|\boldsymbol{\theta})}{p(Z|X,\boldsymbol{\theta}_t)}\\ &= \sum_z p(Z|X,\boldsymbol{\theta}_t)\log p(X,Z|\boldsymbol{\theta}) + \left[-\sum_z p(Z|X,\boldsymbol{\theta}_t)\log p(Z|X,\boldsymbol{\theta}_t)\right]\end{aligned}$$

We note that the term on the right of the $(+)$ above is the entropy of Z, and is constant w.r.t $\boldsymbol{\theta}$, so we exclude it from the optimisation problem. On the other hand, the left side can be written as $\mathbb{E}_{Z\sim p(Z|X,\boldsymbol{\theta}_t)}\left[\log p(X,Z|\boldsymbol{\theta})\right]$ which is our $\mathcal{Q}$ function in step 3 of the algorithm!

16.1 Using it in practice

Consider the GMM case. Observe that $p(X,Z|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}N(x_n|\mu_k,\Sigma_k)^{z_{nk}}$. Then, by the EM algorithm, we need to find new parameters $\boldsymbol{\theta}$ to maximise

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) &= \mathbb{E}_{Z\sim p(Z|X,\hat{\boldsymbol{\theta}})}\left[\log p(X,Z|\boldsymbol{\theta})\right]\\ &= \mathbb{E}_{Z\sim p(Z|X,\hat{\boldsymbol{\theta}})}\left[\log\prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}N(x_n|\mu_k,\Sigma_k)^{z_{nk}}\right]\\ &= \mathbb{E}_{Z\sim p(Z|X,\hat{\boldsymbol{\theta}})}\left[\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}(\log\pi_k + \log N(x_n|\mu_k,\Sigma_k))\right]\\ &= \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_{Z\sim p(Z|X,\hat{\boldsymbol{\theta}})}[z_{nk}](\log\pi_k + \log N(x_n|\mu_k,\Sigma_k))\end{aligned}$$

as only z_{nk} depends on the latent variables Z. To evaluate $p(Z|X,\hat{\boldsymbol{\theta}})$, we first observe that

$$\begin{aligned}p(X|\hat{\boldsymbol{\theta}}) &= \prod_{n=1}^{N}p(x_n|\hat{\boldsymbol{\theta}})\\ &= \prod_{n=1}^{N}\sum_{k=1}^{K}p(x_n,z_{nk}=1|\hat{\boldsymbol{\theta}})\\ &= \prod_{n=1}^{N}\sum_{k=1}^{K}\hat{\pi}_k N(x_n|\hat{\mu}_k,\hat{\Sigma}_k).\end{aligned}$$

Then, through Bayes' rule, we have

$$p(Z|X,\hat{\boldsymbol{\theta}}) = \frac{p(X,Z|\hat{\boldsymbol{\theta}})}{p(X|\hat{\boldsymbol{\theta}})} = \prod_{n=1}^{N}\frac{\prod_{k=1}^{K}\hat{\pi}_k^{z_{nk}}N(x_n|\hat{\mu}_k,\hat{\Sigma}_k)^{z_{nk}}}{\sum_{j=1}^{K}\hat{\pi}_j N(x_n|\hat{\mu}_j,\hat{\Sigma}_j)}$$

If we use the definition of expectation,

$$\begin{aligned}\mathbb{E}_{Z\sim p(Z|X,\hat{\boldsymbol{\theta}})}[z_{nk}] &= p(z_{nk}=1|X,\hat{\boldsymbol{\theta}})\\ &= \gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_k \pi_k N(x_n|\mu_k,\Sigma_k)}\end{aligned}$$

as we want to find the expected probability of x_n belonging to class k. Finally, we arrive at $\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})(\log\pi_k + \log N(x_n|\mu_k,\Sigma_k))$ and we are done!

16.2 Finding the next iteration of $\boldsymbol{\theta}$

Recall that the responsibility term $\gamma(z_{nk})$ depends on some fixed, non-updated parameters $\boldsymbol{\theta}_t$ from step 1 of the EM algorithm. So, if we want to find updated parameters $\boldsymbol{\theta'}$, we only need to differentiate with respect to the new parameters which are assigned to the distributions: μ_k, Σ_k for Gaussian, and λ_{km} for exponential.

16.3 Exponential Mixture Model

Let K be the number of mixture components. For each $n \in \{1, ..., N\}$, we model the latent variable $\mathbf{z_n}$ as a categorical distribution with parameter $\boldsymbol{\pi} \in \mathbb{R}^K$. Given that the n-th point belongs to the k-th component (i.e. $z_{nk} = 1$), we model the M-dimensional observed variable $\mathbf{x}_n$ as an exponential distribution with parameter $\boldsymbol{\lambda}_k \in \mathbb{R}^M$ where we assume independence across the dimensions, i.e. we can factor the distribution $p(\mathbf{x}_n) = \Pi_m p(x_{nm})$ with $p(x_{nm}) = Exp(\lambda_{km})$. The set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \{\boldsymbol{\lambda}_k\}_{k=1}^K\}$

$$\begin{aligned} p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) &= \prod_{n=1}^{N} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) \\ &= \prod_{n=1}^{N}\prod_{k=1}^{K}\left(\pi_k \prod_{m=1}^{M} Exp(x_{nm}|\lambda_{km})\right)^{z_{nk}} \end{aligned}$$

We begin by finding the marginal:

$$\begin{aligned} p(\mathbf{X}|\theta) &= \prod_{n=1}^{N} p(\mathbf{x}_n|\theta) \\ &= \prod_{n=1}^{N}\sum_{k=1}^{K} p(\mathbf{x}_n, z_{nk} = 1|\theta) \\ &= \prod_{n=1}^{N}\sum_{k=1}^{K} \pi_k \prod_{m=1}^{M} \mathrm{Exp}(x_{nm}|\lambda_{km}). \end{aligned}$$

By applying Bayes rule, we have

$$\begin{aligned} p(\mathbf{Z}|\mathbf{X}, \theta) &= \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{p(\mathbf{X}|\theta)} \\ &= \prod_{n=1}^{N} \frac{\prod_{k=1}^{K}\left(\pi_k \prod_{m=1}^{M} \mathrm{Exp}(x_{nm}|\lambda_{km})\right)^{z_{nk}}}{\sum_{j=1}^{K} \pi_j \prod_{m=1}^{M} \mathrm{Exp}(x_{nm}|\lambda_{jm})}. \end{aligned}$$

Substituting the terms into the definition of $\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, we have

$$\begin{aligned} \mathcal{Q}(\theta, \hat{\theta}) &= \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[\log p(\mathbf{X}, \mathbf{Z}|\theta)\right] \\ &= \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[\log \prod_{n=1}^{N}\prod_{k=1}^{K}\left(\pi_k \prod_{m=1}^{M} \mathrm{Exp}(x_{nm}|\lambda_{km})\right)^{z_{nk}}\right] \\ &= \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left(\log \pi_k + \sum_{m=1}^{M} \log \mathrm{Exp}(x_{nm}|\lambda_{km})\right)\right] \\ &= \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left(\log \pi_k + \sum_{m=1}^{M} (\log \lambda_{km} - \lambda_{km} x_{nm})\right)\right] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[z_{nk}\right]\left(\log \pi_k + \sum_{m=1}^{M} (\log \lambda_{km} - \lambda_{km} x_{nm})\right) \end{aligned}$$

Thus, it suffice to find the expectation of z_{nk} with respect to $p(\boldsymbol{Z}|\boldsymbol{X}, \hat{\boldsymbol{\theta}})$. In particular, we have:

$$\mathbb{E}_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\hat{\theta})}\left[z_{nk}\right] = \frac{\hat{\pi}_k \prod_{m=1}^{M} \text{Exp}(x_{nm}|\hat{\lambda}_{km})}{\sum_{j=1}^{K} \hat{\pi}_j \prod_{m=1}^{M} \text{Exp}(x_{nm}|\hat{\lambda}_{jm})} = \gamma(z_{nk})$$

Hence,

$$\mathcal{Q}(\theta, \hat{\theta}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\log \pi_k + \sum_{m=1}^{M}(\log \lambda_{km} - \lambda_{km}x_{nm})\right)$$

To find the updated λ_{km}, we take the derivative of $\mathcal{Q}$ with respect to λ_{km}, set it to 0, and solve it.

$$\begin{aligned}
\nabla_{\lambda_{km}}\mathcal{Q} &= \sum_{n=1}^{N}\gamma(z_{nk})\left(\frac{1}{\lambda_{km}} - x_{nm}\right) = 0 \\
&\implies \frac{1}{\lambda_{km}}\sum_{n=1}^{N}\gamma(z_{nk}) = \sum_{n=1}^{N}\gamma(z_{nk})x_{nm} \\
&\implies \lambda_{km} = \frac{N_k}{\sum_{n=1}^{N}\gamma(z_{nk})x_{nm}} = \left(\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_{nm}\right)^{-1}.
\end{aligned}$$

To find the updated π_k, we need to apply the Lagrangian. The derivation is exactly the same as GMM. In particular:

$$\begin{aligned}
\nabla_{\pi_k}\left(Q + \nu\left(\sum_{k=1}^{K}\pi_k - 1\right)\right) &= \sum_{n=1}^{N}\frac{\gamma(z_{nk})}{\pi_k} + \nu = 0 \\
&\implies \pi_k = -\frac{N_k}{\nu} \\
&\sum_{k=1}^{K}\pi_k = 1 = -\sum_{k=1}^{K}\frac{N_k}{\nu} \\
&\implies \nu = -N \\
&\implies \pi_k = \frac{1}{N}\sum_{n=1}^{N}\gamma(z_{nk}).
\end{aligned}$$

17 Variational Autoencoder (VAE)

17.1 Data, Model and Loss

We first define the data and model in the context of the VAE:

- Data: $\mathcal{D} = \{(\mathbf{x}_n)\}_{n=1}^{N}$ is a set of all inputs $\mathbf{x}$ (e.g. a dataset of images).
- Model: Similar to the case of the Gaussian Mixture Model, we have a Data Generator and a Learner. For each data point, the Data Generator samples the "latent code" $\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and samples features $\mathbf{x_n} \sim \mathcal{N}(\mathbf{x}_n|f_\theta(\mathbf{z}_n), \sigma^2\mathbf{I})$ where f_θ is a neural network with parameters θ. The Learner tries to learn the parameters θ given only the $\mathbf{x_n}$'s, making this an "incomplete/missing data" problem.

We can then define the joint probability distribution used by the Data Generator.

$$p_\theta(X, Z) = \prod_{n=1}^{N} p_\theta(\mathbf{x}_n, \mathbf{z}_n) = \prod_{n=1}^{N} p_\theta(\mathbf{x}_n|\mathbf{z}_n)p_\theta(\mathbf{z}_n) = \prod_{n=1}^{N} N(\mathbf{x}_n|f_\theta(\mathbf{z}_n), \sigma^2\mathbf{I})N(\mathbf{z}_n; \mathbf{0}, \mathbf{I})$$

Now, we use MLE to derive the loss for VAE. We want to maximize $\log p_\theta(X)$. We find that the marginal $p_\theta(\mathbf{x}_n) = \int_{z_n} p_\theta(\mathbf{x}_n, \mathbf{z}_n) d\mathbf{z_n} = \int_{z_n} N(\mathbf{x}_n | f_\theta(\mathbf{z}_n), \sigma^2 \mathbf{I}) N(\mathbf{z}_n; \mathbf{0}, \mathbf{I}) d\mathbf{z_n}$ which is difficult to compute and maximize directly. Hence, we can maximize the lower bound/ELBO instead:

$$\log p_\theta(X; \theta) = \mathcal{L}(q, \theta) + \mathcal{D}_{KL}[q||p] = \sum_z q(z) \log \frac{p(X, Z|\boldsymbol{\theta})}{q(z)} + \sum_z q(z) \log \frac{q(z)}{p(Z|X, \boldsymbol{\theta})}$$

In the case of the VAE, we can't directly use the EM algorithm to maximize the ELBO as the M-step requires us to calculate and maximize $E_{\mathbf{z}|\mathbf{x}, \theta^{(t)}}[\log p_\theta(\mathbf{x}, \mathbf{z})]$ which is taken over the true posterior $p_{\theta^{(t)}}(\mathbf{z}|\mathbf{x})$. The calculation of this true posterior is intractable for VAE, so we would not be able to use the EM algorithm. Instead, we use SGD to maximize the ELBO.

Since q is a distribution, we can think of optimizing the ELBO with respect to q as finding a $q_\psi \in \mathcal{Q}$ where $\mathcal{Q}$ is a family of distributions (e.g. Gaussians). We are optimizing the ELBO with respect to the parameters ψ of distribution q (e.g. $\psi = (\mu, \Sigma)$ in the case where q is a Gaussian). We call $q_\psi(Z)$ the variational distribution.

We break down the ELBO into the Reconstruction term and Sum of KL Divergence terms as follows:

$$\begin{aligned}
\mathcal{L}(\psi, \theta) &= \mathbb{E}_{q_\psi(Z)} \log \frac{p_\theta(X, Z)}{q_\psi(z)} \\
&= \mathbb{E}_{q_\psi(Z)} \log \frac{p_\theta(X|Z) p_\theta(Z)}{q_\psi(z)} \\
&= \mathbb{E}_{q_\psi(Z)}[\log p_\theta(X|Z)] + \mathbb{E}_{q_\psi(Z)} \log \frac{p_\theta(Z)}{q_\psi(z)} \\
&= \mathbb{E}_{q_\psi(Z)}[\log p_\theta(X|Z)] - \mathbb{D}_{KL}[q_\psi(z)||p_\theta(Z)] \\
&= \mathbb{E}_{q_\psi(Z)}[\log \prod_{n=1}^{N} p_\theta(\mathbf{x}_n|\mathbf{z}_n)] - \mathbb{D}_{KL}[\prod_{n=1}^{N} q_\psi(\mathbf{z}_n)|| \prod_{n=1}^{N} p_\theta(\mathbf{z}_n)] \\
&= \sum_{n=1}^{N} \mathbb{E}_{q_\psi(\mathbf{z}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] - \sum_{n=1}^{N} \mathbb{D}_{KL}[q_\psi(\mathbf{z}_n)||p_\theta(\mathbf{z}_n)]
\end{aligned}$$

The first term is the Reconstruction term and the second is the Sum of KL Divergence terms.

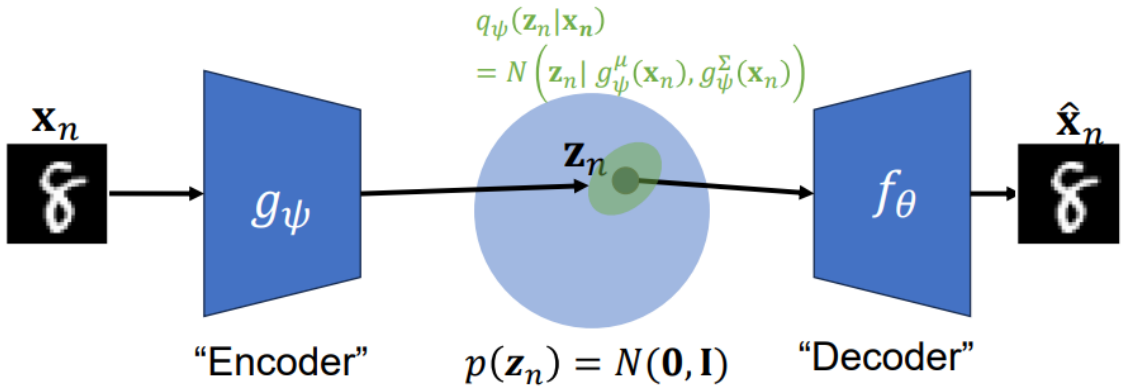
For the Reconstruction term, we can approximate it using Monte Carlo approximation (sampling):

$$\mathbb{E}_{q_\psi(\mathbf{z}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] \approx \frac{1}{M} \sum_{m=1}^{M} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(m)})$$

where $\mathbf{z}_n^{(m)} \sim q_\psi(\mathbf{z}_n)$.

We see 3 distributions in our deconstructed ELBO: $p_\theta(\mathbf{x}_n|\mathbf{z}_n)$, $p_\theta(\mathbf{z}_n)$ and $q_\psi(\mathbf{z}_n)$. From earlier, we know that $p_\theta(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n; \mathbf{0}, \mathbf{I})$ which has no dependence on θ and $p_\theta(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|f_\theta(\mathbf{z}_n), \sigma^2\mathbf{I})$. Hence, all we need to do now is to find $q_\psi(\mathbf{z}_n)$. We can make q_ψ conditional on $\mathbf{x}_n$ as seen in the diagram below. So $q_\psi(\mathbf{z}_n|\mathbf{x}_n) = \mathcal{N}(\mathbf{z}_n|g_\psi^\mu(\mathbf{x}_n), g_\psi^\Sigma(\mathbf{x}_n))$. We end up with these formulas:

- Reconstruction term: $\mathbb{E}_{q_\psi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] \approx \frac{1}{M} \sum_{m=1}^{M} \log \mathcal{N}(\mathbf{x}_n|f_\theta(z_n^{(m)}), \sigma^2 I)$
- KL Divergence term: $\mathbb{D}_{KL}[q_\psi(\mathbf{z}_n|\mathbf{x}_n)||p_\theta(\mathbf{z}_n) = \mathbb{D}_{KL}[\mathcal{N}(\mathbf{z}_n|g_\psi^\mu(\mathbf{x}_n), g_\psi^\Sigma(\mathbf{x}_n))||\mathcal{N}(0, I)]$



$$L(\psi,\theta) = \sum_n \mathbb{E}_{q_\psi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] - \sum_n \mathbb{D}_{KL}\left[q_\psi(\mathbf{z}_n|\mathbf{x}_n)\|p_\theta(\mathbf{z}_n)\right]$$

17.2 Optimization and the reparameterization trick

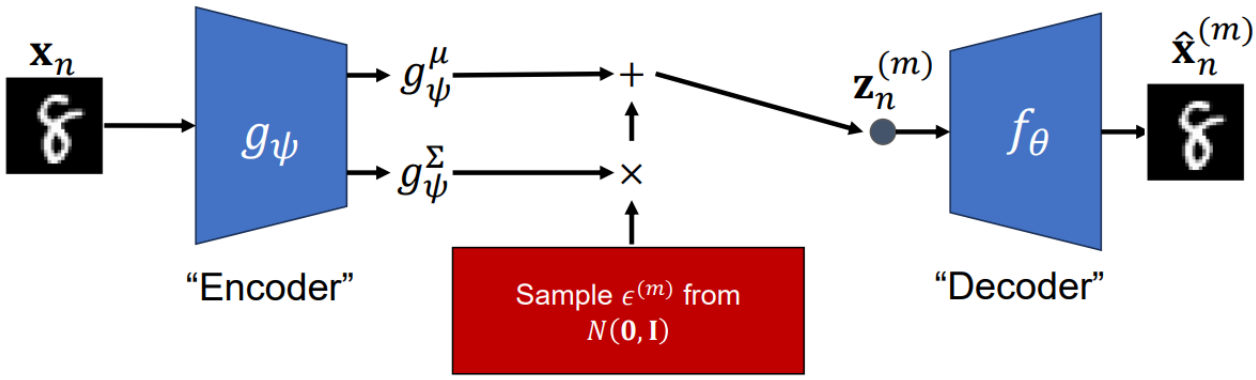
Recall that we are using sampling to approximate the Reconstruction term. As sampling is not differentiable, we are unable to use SGD to optimize our loss term just yet. This is where the reparameterization trick comes in. We will separate the randomness from the "Encoder" and leverage the following property of Gaussians:

$$\text{If } \epsilon \sim \mathcal{N}(0, I) \text{ and } y = \mu + \sigma \times \epsilon, \text{ then } y \sim \mathcal{N}(\mu, \sigma^2).$$

We rewrite $\mathbf{z}_n^{(m)}$ as $(\epsilon_n^{(m)} \times g_\psi^\Sigma(\mathbf{x}_n)) + g_\psi^\mu(\mathbf{x}_n)$.

This gives us the ELBO in a form that is differentiable. By optimizing the ELBO, we are therefore able to find our loss term for the VAE.

$$L(\psi,\theta) = \sum_n \mathbb{E}_{q_\psi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] - \sum_n \mathbb{D}_{KL}\left[q_\psi(\mathbf{z}_n)\|p_\theta(\mathbf{z}_n)\right]$$



$$\mathbb{E}_{q_\psi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] \approx \frac{1}{M}\sum_m \log N(\mathbf{x}_n|f_\theta\left(z_n^{(m)}\right), \sigma^2\mathbf{I})$$

$$\text{where } \mathbf{z}_n^{(m)} = \left(\epsilon_n^{(m)} \times g_\psi^\Sigma(\mathbf{x}_n)\right) + g_\psi^\mu(\mathbf{x}_n)$$

$$\mathbb{D}_{KL}\left[q_\psi(\mathbf{z}_n|\mathbf{x}_n)\|p_\theta(\mathbf{z}_n)\right] = \mathbb{D}_{KL}\left[N\left(\mathbf{z}_n|\, g_\psi^\mu(\mathbf{x}_n), g_\psi^\Sigma(\mathbf{x}_n)\right) \| N(0, \mathrm{I})\right]$$