

Machine Learning

Homework 1

系級：醫工三 學號：B12508025 姓名：盧雅筠

Problem 1: cross-validation, feature selection, and classification

I. Reports

1. The multivariate Gaussian Distribution models and Parameter Estimation

- For each class $y \in \{0,1\}$, $\mathbf{x} \in \mathbb{R}^d$ assume the multivariate Gaussian distribution:

$$\mathbf{x} | y = c \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \quad c \in \{0,1\}$$

$$p(\mathbf{x} | y = c) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_c|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c) \right]$$

where $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ denote the mean vector and covariance matrix of class c .

- For 單一樣本 x , log-likelihood 如下:

$$\log p(x | \mu, \Sigma) = -\frac{1}{2} \left[d \log(2\pi) + \log \det(\Sigma) + (x - \mu)^T \Sigma^{-1} (x - \mu) \right]$$

Cholesky decomposition of the covariance matrix: $\Sigma = LL^T$, 這個表示法可以使數值穩定,

跳過反矩陣運算, 得 $\log \det(\Sigma) = 2 \sum_{i=1}^d \log L_{ii}$, $(x - \mu)^T \Sigma^{-1} (x - \mu) = \|L^{-1}(x - \mu)\|_2^2$

- Parameter Estimation: maximum likelihood estimation (MLE) use

`np.mean(X_c, axis=0)` and `cov = np.cov(X_c, rowvar=False)` and `MLE_Estimator()` to compute:

$$\ell(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \sum_{i: y_i=c} \log p(\mathbf{x}_i | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{i: y_i=c} \mathbf{x}_i \quad \hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c - 1} \sum_{i: y_i=c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T$$

- Posterior Probability: use

`Multivariate_Gaussian_Distribution_log_likelihood()` and `Bayesian_decision_classifier()` to compute:

$$\ell_0 = \log p(x | y = 0) + \log p_0$$

$$\ell_1 = \log p(x | y = 1) + \log p_1$$

$$P(y = 1 | \mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=1)p(y=1) + p(\mathbf{x}|y=0)p(y=0)} = \frac{e^{\ell_1}}{e^{\ell_0} + e^{\ell_1}} = \frac{1}{1 + e^{-(\ell_1 - \ell_0)}}$$

- Discriminant Function: use

`Multivariate_Gaussian_Distribution_log_likelihood()` and `Bayesian_decision_classifier()` to compute:

$$g_c(\mathbf{x}) = \ln P(\mathbf{x}|y = c) + \ln p(y = c)$$

$\Delta(\mathbf{x}) = g_1(\mathbf{x}) - g_0(\mathbf{x})$, the decision boundary is $\Delta(\mathbf{x}) = 0$

$$\Delta \stackrel{\text{def}}{=} \ell_1 - \ell_0 = \log \frac{P(y = 1 | x)}{P(y = 0 | x)}$$

為了避免 overflow，使用 log-sum-exp trick : $m = \max(\ell_0, \ell_1)$

$$P(y = 1 | x) = \frac{e^{\ell_1 - m}}{e^{\ell_0 - m} + e^{\ell_1 - m}}$$

2. Forward Feature Selection and its cost Function

- Feature selection:

use `Forward_Selection_AUC()` and its cost function `cv_AUC_score()`

Starting from an empty feature set, at each iteration the algorithm tests all

remaining features and adds the one that maximizes the AUC score. `cv_AUC_score()`

use Stratified K-Fold calculate AUC score.

- Determine the number of features to be selected

The maximum features are determined by

$$\text{max_features} = \min(\max(1, \min(n_0, n_1) - 1), d)$$

where n_0, n_1 is the number of two classes samples, d is $X_{\text{train}}. \text{shape}[1]$

3. Selected Features per Fold

- Each of the 103 data samples serves as a test point once in leave-one-out cross-validation, yielding 103 different feature subsets. The selected features for each fold are stored in `selected_features`. The list of selected features per fold is shown in

Figure 1

4. The Test Performance

- Accuracy, Sensitivity, Specificity, AUC are shown in Figure 2

```
Test Performance
Confusion Matrix:
[[56  6]
 [ 7 34]]
Accuracy=0.874, Sensitivity=0.829, Specificity=0.903, AUC=0.906
```

Figure 2

- The ROC curve is shown in Figure 3

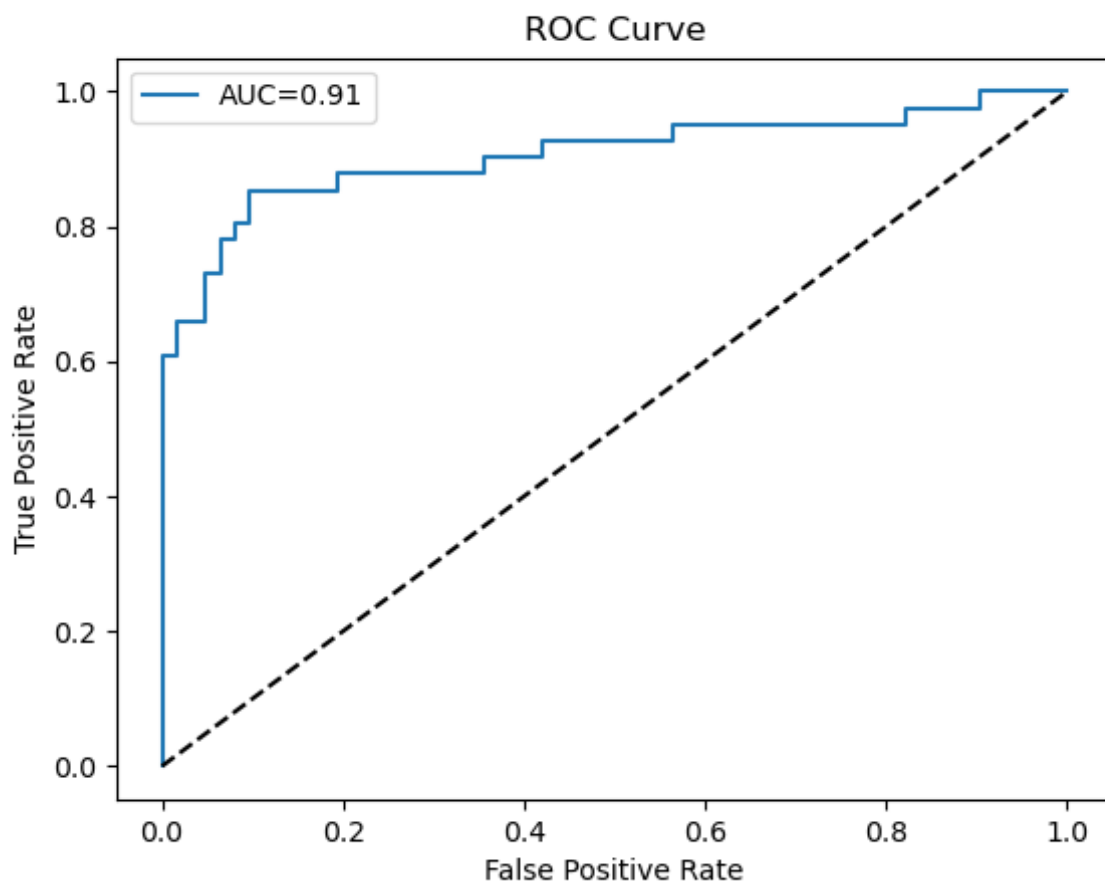


Figure 3

5. Top-2 Most Frequently Selected Features and Bivariate Gaussian Bayesian Model Visualization

- Top-2 Most Frequently Selected Features use

`counts = Counter()` and `counts.most_common(2)` to find ,
shown in Figure 4

```
Top-2 features: [('NoseVol', 92), ('LVermilionH', 87)]
```

Figure 4

- Bivariate Gaussian Bayesian Model Visualization is shown in Figure 5
 - Scatter plot: class 0 as circles o(gray), class 1 as plus signs +(green)
 - Contour maps: equal-probability contours of each class distribution
 - Decision boundary: the curve where $\Delta(\mathbf{x}) = 0$

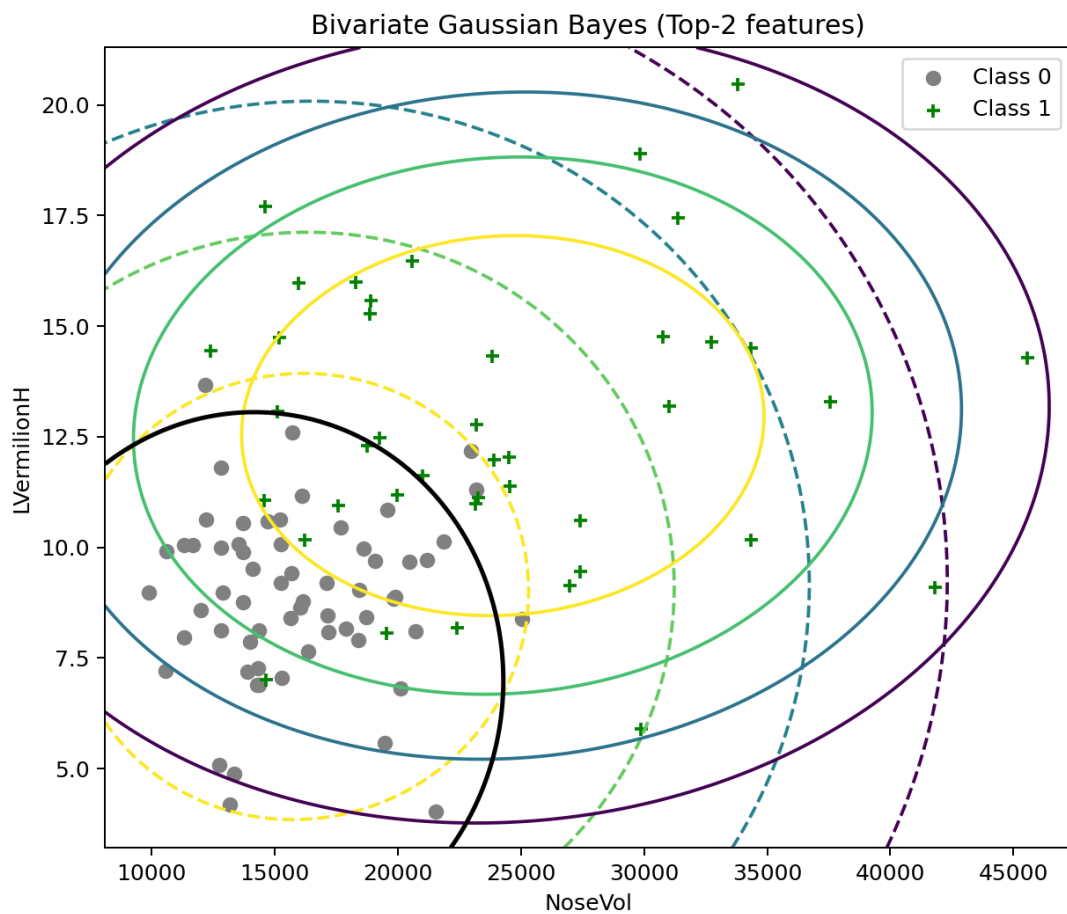


Figure 5

II. Code

Set up the environment `environment.yml` and run `1.py` shown in Figure 7, making sure to remove unnecessary information from `AcromegalyFeatureSet.xlsx` (eg. Figure 6)

98	97	1	2	153.062	154.967
99	98	1	2	137.798	135.533
100	99	1	2	151.329	150.528
101	100	1	2	153.679	152.093
102	101	1	1	175.044	166.512
103	102	1	1	161.504	160.874
104	103	1	2	156.574	148.620
105					
106		0: control	1: male		
107		1: patient	2: female		
108					
109					
110					
111					
112					
113					

Figure 6. The blue rectangle must be deleted

```

1  import numpy as np
2  import pandas as pd
3  from sklearn.metrics import roc_curve, auc, confusion_matrix
4  import matplotlib.pyplot as plt
5  from collections import Counter
6  from sklearn.model_selection import StratifiedKFold
7  from sklearn.metrics import roc_auc_score
8
9  ''' Multivariate Gaussian Bayesian classifier '''
10 def Multivariate_Gaussian_Distribution_log_likelihood(X, mu, cov):
11     X = np.atleast_2d(X)
12     d = X.shape[1]
13     L = np.linalg.cholesky(cov)
14     Z = np.linalg.solve(L, (X - mu).T)
15     quad = np.sum(Z*Z, axis=0)
16     logdet = 2.0 * np.sum(np.log(np.diag(L)))
17     log_likelihood = -0.5 * (d*np.log(2*np.pi) + logdet + quad)
18     return log_likelihood
19
20 def Bayesian_decision_classifier(X, mu0, mu1, cov0, cov1, p0, p1, eps=1e-15):
21     ll0 = Multivariate_Gaussian_Distribution_log_likelihood(X, mu0, cov0) + np.log(p0 + eps) # (n,)
22     ll1 = Multivariate_Gaussian_Distribution_log_likelihood(X, mu1, cov1) + np.log(p1 + eps) # (n,)
23     delta = ll1 - ll0
24     m = np.maximum(ll0, ll1)
25     num1 = np.exp(ll1 - m)
26     den = np.exp(ll0 - m) + num1
27     posterior_1 = num1 / den
28     # If single sample, return scalars
29     if posterior_1.shape[0] == 1:
30         return float(delta[0]), float(posterior_1[0])
31     return delta, posterior_1
32
33 ''' MLE Estimator for Gaussian parameters '''
34 def MLE_Estimator(X, y, ridge=1e-6):
35     X0, X1 = X[y == 0], X[y == 1]
36     mu0, mu1 = X0.mean(0), X1.mean(0)
37     cov0 = np.atleast_2d(np.cov(X0, rowvar=False)) + np.eye(X.shape[1]) * ridge
38     cov1 = np.atleast_2d(np.cov(X1, rowvar=False)) + np.eye(X.shape[1]) * ridge
39     p0, p1 = len(X0)/len(X), len(X1)/len(X)
40     return mu0, mu1, cov0, cov1, p0, p1
41
42 ''' Cross-Validation AUC Score '''
43 def cv_AUC_score(X_train_sub, y_train_sub, max_splits=5, random_state=42):
44     n_pos = int(np.sum(y_train_sub == 1))
45     n_neg = int(np.sum(y_train_sub == 0))
46     n_splits = max(2, min(max_splits, n_pos, n_neg))
47     if n_pos == 0 or n_neg == 0 or n_splits < 2:
48         return 0.5
49
50     skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=random_state)
51     y_all, s_all = [], []
52     for tr_idx, va_idx in skf.split(X_train_sub, y_train_sub):
53         Xtr, Xva = X_train_sub[tr_idx], X_train_sub[va_idx]
54         ytr, yva = y_train_sub[tr_idx], y_train_sub[va_idx]
55         mu0, mu1, cov0, cov1, p0, p1 = MLE_Estimator(Xtr, ytr, ridge=1e-6)
56         s = Bayesian_decision_classifier(Xva, mu0, mu1, cov0, cov1, p0, p1)
57         y_all.append(yva); s_all.append(s)
58     y_all = np.concatenate(y_all); s_all = np.concatenate(s_all)
59     try:
60         return roc_auc_score(y_all, s_all)
61     except Exception:
62         return 0.5

```

```

64 ''' Forward Feature Selection based on CV-AUC '''
65 def Forward_Selection_AUC(X_train, y_train, max_features):
66     remaining = list(range(X_train.shape[1]))
67     selected = []
68     best_score = -np.inf
69
70     while remaining and len(selected) < max_features:
71         candidates = []
72         for f in remaining:
73             trial = selected + [f]
74             score = cv_AUC_score(X_train[:, trial], y_train)
75             candidates.append((score, f))
76             new_score, new_f = max(candidates, key=lambda x: x[0])
77
78             if (new_score > best_score + 1e-9):
79                 best_score = new_score
80                 selected.append(new_f)
81                 remaining.remove(new_f)
82     return selected
83
84 ''' Load Data '''
85 data = pd.read_excel('AcromegalyFeatureSet.xlsx')
86 data.rename(columns=lambda s: s.strip() if isinstance(s, str) else s, inplace=True)
87 X = data.drop(columns=['SeqNum', 'Gender', 'GroundTruth']).values
88 y = data['GroundTruth'].values
89 feature_names = data.drop(columns=['SeqNum', 'Gender', 'GroundTruth']).columns
90 n = len(y)
91
92 ''' Leave-One-Out Cross Validation '''
93 posts1 = []
94 selected_features = []
95 deltas = []
96 print(f"Leave-One-Out Cross Validation")
97 for i in range(n):
98     X_train = np.delete(X, i, axis=0)
99     y_train = np.delete(y, i)
100     X_test = X[i]
101
102     n0 = int(np.sum(y_train == 0))
103     n1 = int(np.sum(y_train == 1))
104     print(f"-----")
105     print(f"Fold: {i+1}/{n}, Class 0 samples: {n0}, Class 1 samples: {n1}")
106
107     max_features = min(max(1, min(n0, n1) - 1), X_train.shape[1])
108
109     selected = Forward_Selection_AUC(X_train, y_train, max_features)
110
111     selected_features.append([feature_names[j] for j in selected])
112     print(f"Selected features: {[feature_names[j] for j in selected]}")
113     X_train_sel = X_train[:, selected]
114     X_test_sel = X_test[selected]
115
116     # mu0, mu1 = mean vectors for class 0 and 1
117     mu0 = X_train_sel[y_train == 0].mean(axis=0)
118     mu1 = X_train_sel[y_train == 1].mean(axis=0)
119
120     # cov0, cov1 = covariance matrices for class 0 and 1
121     cov0 = np.atleast_2d(np.cov(X_train_sel[y_train == 0], rowvar=False)) + np.eye(X_train_sel.shape[1]) * 1e-6
122     cov1 = np.atleast_2d(np.cov(X_train_sel[y_train == 1], rowvar=False)) + np.eye(X_train_sel.shape[1]) * 1e-6
123
124     # p0 = prior for class 0, p1 = prior for class 1
125     p0, p1 = np.mean(y_train == 0), np.mean(y_train == 1)
126
127     delta, posterior1 = Bayesian_decision_classifier(X_test_sel, mu0, mu1, cov0, cov1, p0, p1)
128     posts1.append(posterior1)
129     deltas.append(delta)
130 posts1 = np.array(posts1)
131 deltas = np.array(deltas)
132
133 ''' Performance '''
134 fpr, tpr, _ = roc_curve(y, posts1)
135 roc_auc = auc(fpr, tpr)
136 preds1 = (deltas >= 0).astype(int)
137 cm = confusion_matrix(y, preds1)
138 TN, FP, FN, TP = cm.ravel()
139 acc = (TP + TN) / np.sum(cm)
140 sen = TP / (TP + FN)
141 spe = TN / (TN + FP)
142 print(f"-----")
143 print("Test Performance")
144 print(f"Confusion Matrix:\n{cm}")
145 print(f"Accuracy={acc:.3f}, Sensitivity={sen:.3f}, Specificity={spe:.3f}, AUC={roc_auc:.3f}")
146
147 # ROC curve plot
148 plt.plot(fpr, tpr, label=f"AUC={roc_auc:.2f}")
149 plt.plot([0, 1], [0, 1], 'k--')
150 plt.xlabel('False Positive Rate')
151 plt.ylabel('True Positive Rate')
152 plt.legend()
153 plt.title('ROC Curve')
154 plt.savefig('roc_curve.png')
155
156 ''' Bivariate Gaussian Bayes classifier with top-2 features '''
157 counts = Counter([f for fs in selected_features for f in fs])
158 print('Top-2 features:', counts.most_common(2))
159
160 Top2_features = [f for f, _ in counts.most_common(2)]
161
162 Top2_X = data[Top2_features].values.astype(float)
163 Top2_y = data['GroundTruth'].values.astype(int)
164
165 Top2_X0, Top2_X1 = Top2_X[Top2_y==0], Top2_X[Top2_y==1]
166 Top2_mu0, Top2_mu1 = Top2_X0.mean(0), Top2_X1.mean(0)
167 Top2_cov0 = np.cov(Top2_X0, rowvar=False) + np.eye(Top2_X0.shape[1]) * 1e-6
168 Top2_cov1 = np.cov(Top2_X1, rowvar=False) + np.eye(Top2_X1.shape[1]) * 1e-6
169 Top2_p0, Top2_p1 = len(Top2_X0)/len(Top2_X), len(Top2_X1)/len(Top2_X)

```



```

171 # grid points
172 m, M = Top2_X.min(0), Top2_X.max(0); pad = 0.05*(M-m)
173 xs = np.linspace(m[0]-pad[0], M[0]+pad[0], 300)
174 ys = np.linspace(m[1]-pad[1], M[1]+pad[1], 300)
175 xx, yy = np.meshgrid(xs, ys)
176 pts = np.c_[xx.ravel(), yy.ravel()]
177
178 # log-likelihoods and decision boundary
179 LL0 = Multivariate_Gaussian_Distribution_log_likelihood(pts, Top2_mu0, Top2_cov0).reshape(xx.shape)
180 LL1 = Multivariate_Gaussian_Distribution_log_likelihood(pts, Top2_mu1, Top2_cov1).reshape(xx.shape)
181 g0 = LL0 + np.log(Top2_p0)
182 g1 = LL1 + np.log(Top2_p1)
183 Top2_delta = g1 - g0 # decision boundary: delta=0
184
185 # plot
186 plt.figure(figsize=(7,6))
187 plt.scatter(Top2_X0[:,0], Top2_X0[:,1], marker='o', label='Class 0', color = 'gray')
188 plt.scatter(Top2_X1[:,0], Top2_X1[:,1], marker='+', label='Class 1', color = 'green')
189 levels0 = np.unique(np.sort(np.percentile(LL0, [20, 40, 60, 80])))
190 levels1 = np.unique(np.sort(np.percentile(LL1, [20, 40, 60, 80])))
191
192 plt.contour(xx, yy, LL0, levels=levels0, linestyles='dashed')
193 plt.contour(xx, yy, LL1, levels=levels1, linestyles='solid')
194
195 # The decision boundary
196 plt.contour(xx, yy, Top2_delta, levels=[0.0], linewidths=2, colors='black')
197
198 plt.xlabel(Top2_features[0]); plt.ylabel(Top2_features[1])
199 plt.title('Bivariate Gaussian Bayes (Top-2 features)')
200 plt.legend(); plt.tight_layout()
201 plt.savefig('bivariate_bayes_qda.png', dpi=180)

```

Figure 7

Problem 2. Proof of Bayesian estimator.

Suppose $x^t \sim N(\theta, \sigma^2)$ and $\theta \sim N(u_0, \sigma_0^2)$, where $u_0, \sigma_0^2, \sigma^2$ are known. That is

$$p(X|\theta) = \frac{1}{(2\pi)^{N/2} \sigma^N} \exp \left[-\frac{\sum_t (x^t - \theta)^2}{2\sigma^2} \right]$$

$$p(\theta) = \frac{1}{\sqrt{2\pi} \sigma_0} \exp \left[-\frac{(\theta - \mu_0)^2}{2\sigma_0^2} \right]$$

Please show that

$$E[\theta|X] = \frac{N/\sigma^2}{N/\sigma^2 + 1/\sigma_0^2} m + \frac{1/\sigma_0^2}{N/\sigma^2 + 1/\sigma_0^2} \mu_0$$

where m is the maximum likelihood estimator of the sample mean.

$$\left. \begin{array}{l} \text{suppose Data: } x^t \sim N(\theta, \sigma^2) \quad t \in \{1, \dots, N\} \\ \text{Prior distribution } \theta \sim N(\mu_0, \sigma_0^2) \end{array} \right\} \Rightarrow p(X|\theta) = \frac{1}{(2\pi)^{N/2} \sigma^N} \exp \left[-\frac{\sum_t (x^t - \theta)^2}{2\sigma^2} \right]$$

$$p(\theta) = \frac{1}{\sqrt{2\pi} \sigma_0} \exp \left[-\frac{(\theta - \mu_0)^2}{2\sigma_0^2} \right]$$

由 Bayes' Rule 得 $p(\theta|X) \propto p(X|\theta) p(\theta)$

$$\Rightarrow p(\theta|X) \propto \exp \left[-\frac{1}{2\sigma^2} \sum_t (x^t - \theta)^2 - \frac{1}{2\sigma_0^2} (\theta - \mu_0)^2 \right] \because \mu_0, \sigma_0^2, \sigma^2 \text{ are known}$$

$$\sum_t (x^t - \theta)^2 = \sum_t (x^t)^2 - 2\theta \sum_t x^t + N\theta^2, \quad (\theta - \mu_0)^2 = \theta^2 - 2\theta\mu_0 + \mu_0^2$$

$$\Rightarrow -\frac{1}{2\sigma^2} \sum_t (x^t - \theta)^2 - \frac{1}{2\sigma_0^2} (\theta - \mu_0)^2 = -\frac{1}{2} \left[\frac{N}{\sigma^2} \theta^2 - 2 \frac{\sum_t x^t}{\sigma^2} \theta + \frac{1}{\sigma_0^2} \theta^2 - 2 \frac{\mu_0}{\sigma_0^2} \theta \right] + \underbrace{\left(-\frac{\sum_t (x^t)^2}{2\sigma^2} - \frac{1}{2\sigma_0^2} \sum_t (x^t)^2 \right)}_{\text{constant}}$$

$$\text{set } A = \frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}, \quad B = \frac{\sum_t x^t}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}$$

$$\Rightarrow p(\theta|X) \propto \exp \left[-\frac{1}{2} (A\theta^2 - 2B\theta) \right] = \exp \left[-\frac{A}{2} \left(\theta^2 - 2 \frac{B}{A} \theta \right) \right] \propto \exp \left[-\frac{A}{2} \left(\theta^2 - 2 \frac{B}{A} \theta + \left(\frac{B}{A} \right)^2 \right) \right]_{\text{constant}}$$

$$= \exp \left[-\frac{A}{2} \left(\theta - \frac{B}{A} \right)^2 \right]$$

$$\Rightarrow p(\theta|X) = N \left(\text{mean} = \frac{B}{A}, \text{variance} = \frac{1}{A} \right)$$

$$\Rightarrow E(\theta|X) = \frac{B}{A} = \frac{\frac{\sum_t x^t}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}}$$

' $m = \frac{1}{N} \sum_t x^t$ the maximum likelihood of sample mean $\Rightarrow \sum_t x^t = Nm$

$$\Rightarrow E(\theta|X) = \frac{\frac{Nm}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{N/\sigma^2}{N/\sigma^2 + 1/\sigma_0^2} m + \frac{1/\sigma_0^2}{N/\sigma^2 + 1/\sigma_0^2} \mu_0$$

得證 #