E1 213: Pattern Recognition and Neural Networks **Spring 2021**

Assignment 1: Implementing Bayes' Classifier

Author: Abijith J. Kamath *Email:* abijithj@iisc.ac.in

Introduction

Let patterns be in $\mathbb{R}^d$ and consider the 2-class classification problem - given a pattern $\mathbf{x}$, classify the pattern with a label y. In classification using linear least-squares and logistic regression the training data $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^n$ is used to directly learn a discriminative function that scores the new patterns, and the classification is achieved by thresholding the score.

Linear Least-Squares Classification

Consider the two-class classification problem of classifying patterns into labels $y \in \{-1, +1\}$. The discriminative function in linear least-squares classification is a function that is linear in the weights, using features of the patterns. In this case, the patterns are directly used as the features, and the discriminative function is given as:

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{W}^\mathsf{T} \tilde{\mathbf{x}}, \tag{0.1}$$

where $\mathbf{W} = [w_0, w_1, \cdots, w_d] \in \mathbb{R}^{d+1}$ are the weights to be learnt from the training samples and $\tilde{\mathbf{x}}$ is the augmented feature vector with 1 padded as its first entry. The classification is then done using the sign of the score function $f(\mathbf{x})$. This can be easily extended to a multi-class classifier by consider the labels to be one-hot vectors and choosing the class that gives the largest score.

The weights are obtained by minimising the ℓ^2-norm between the predictions $f(\mathbf{x}^i)$ and $y^{(i)}$:

$$J(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{n} \left(\mathbf{W}^\mathsf{T} \mathbf{x}^{(i)} - y^{(i)} \right)^2. \tag{0.2}$$

The minimiser of $J(\mathbf{W})$ can be obtained in closed form as $\mathbf{W}^* = \mathbf{A}^\dagger \mathbf{y}$, where $\mathbf{A} = [\mathbf{x}^{(1)}\ \mathbf{x}^{(2)}\ \cdots ; \mathbf{x}^{(n)}]^\mathsf{T}$ and $\mathbf{y} = [y^{(1)}\ y^{(2)}\ \cdots\ y^{(n)}]^\mathsf{T}$.

Logistic Regression

Consider the two-class classification problem of classifying patterns into labels $y \in \{-1, +1\}$. The discriminative function in logistic regression is the sigmoid:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{W}^\mathsf{T}\tilde{\mathbf{x}}}}, \tag{0.3}$$

which is treated as the probability of assigning $y = +1$ to the pattern $\mathbf{x}$ ($\mathbf{W}$ and $\tilde{\mathbf{x}}$ have the same meaning as above). If the score is less than $1/2$, the pattern is classified into $y = -1$.

The weights are obtained from the training samples by maximising the likelihood using gradient descent. The weights cannot be obtained in closed form like in Linear Least-Squares classification. The gradient descent updates for the weights is given by:

$$\mathbf{W}^+ = \mathbf{W} + \eta \sum_{i=1}^{n} \left(y^{(i)} - \sigma(\mathbf{W}^\mathsf{T}\tilde{\mathbf{x}}^{(i)})\right) \tilde{\mathbf{x}}^{(i)}, \tag{0.4}$$

where σ is the sigmoid function and η is the step size.

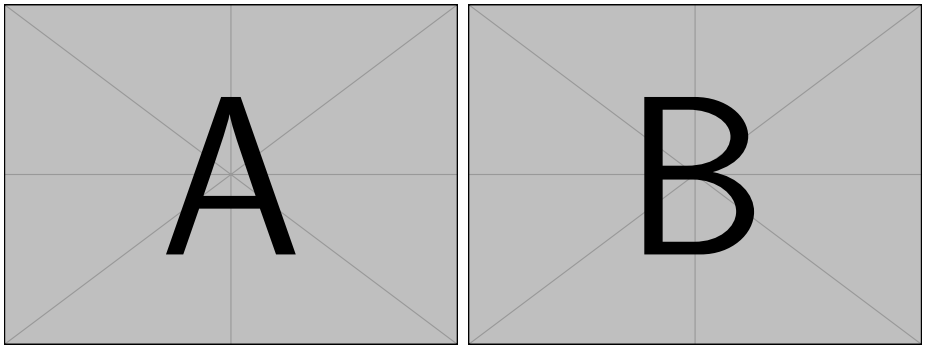
Logistic regression can be extended to multi-class classification by taking the score function for each class to be the softmax function, which is interpreted as assigning the probability of the pattern being in each class and then choosing the class that gives the highest probability.

1 Linear Least-Squares and Logistic Regression

Problem (1.a) Gamma-distributed Classes in 2D

2 Code Repository

The Python codes to reproduce the results can be found in the GitHub repository https://github.com/kamath-abhijith/Bayes_Classifier. Use `requirements.txt` to install the dependencies.



(a) Figure A (b) Figure B

Figure 1: my caption