CMSC 510 – Fall 2018

Homework Assignment 1

Announced: 9/26

Due: Monday, 10/15, 3pm

Parametric Modeling

- Maximum likelihood (actually, maximum a posteriori) estimation of model parameters, for two-class classification

  - Part A: $p(x|y)$ is Gaussian
    we have two distributions:
    - $p(x|y=classA)$
    - $P(x|y=classB)$

  - Part B: $p(y|x)$ as in Logistic Regression

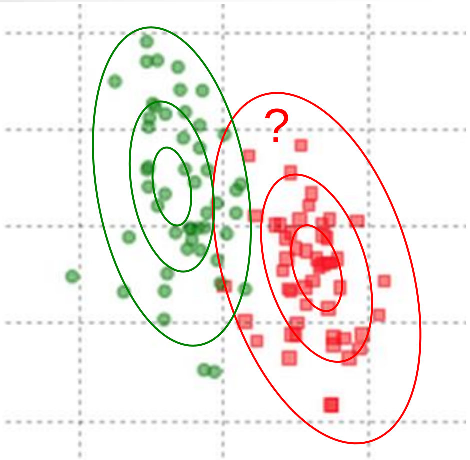
Parametric Modeling

- We will use PyMC3
  - Statistical library for Python 3
  - We specify a statistical model with some unknown parameters
    - E.g. $P(x_k|\theta_i) \Leftrightarrow x \sim Normal(mean,covariance)$
      mean and covariance are unknown
  - We tie it to actual observed data
    - E.g. samples (feature vectors) x_k
  - And then we estimate the parameters
    - E.g. we find some good values for mean and covariance

Parametric Modeling

- Part A: p(x|y) is Gaussian

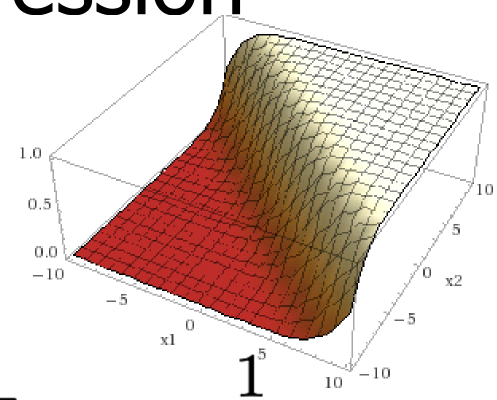
  p(x|y=classA) and P(x|y=classB)



- See L06:
  - Maximum a posteriori (MAP):
    - Finding θ_i that maximizes $P(\theta_i|S_i) \sim P(S_i|\theta_i)P(\theta_i)$
    - Where $P(S_i|\theta_i)P(\theta_i) = \Pi_k P(x_k|\theta_i)P(\theta_i)$
- In our case $P(x_k|\theta_i)$ = Normal(mean,covariance)
  - That is, θ_i represents the mean and the covariance matrix of the gaussian
- In PyMC3, we model that as (unknown, known):
- x1_obs = pm.MvNormal('x1', mu=mu1,chol=chol, observed = x1);
- x0_obs = pm.MvNormal('x0', mu=mu0,chol=chol, observed = x0);
  - Note: different means, same covariance (given here as Cholesky Decomposition of Cov. Mx)

Parametric Modeling

- Part B: p(y|x) as in Logistic Regression
- p(x|y=classA) and P(x|y=classB)



- See L06:
  - We have:

$$P(y_i \,|x_i, w\,) = a(y_i w^T x_i) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$$

    - We will also use a prior P(w) (a Gaussian prior)
  - Y is modelled as a Bernoulli distribution

In probability theory and statistics, the **Bernoulli distribution**, named after Swiss mathematician Jacob Bernoulli,[1] is the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability $q = 1 - p$, that is, the probability distribution of any single experiment that asks a yes–no question; the question results in a boolean-valued outcome, a single bit of information whose value is success/yes/true/one with probability *p* and failure/no/false/zero with probability *q*. It can be used to represent a coin toss where 1 and 0 would represent "head" and "tail" (or vice versa), respectively. In particular, unfair coins would have $p \neq 0.5$.

  - In PyMC3, that's:
    - Y_obs=pm.Bernoulli('Y_obs', p=prob, observed = Y);

PyMC3

- We will use PyMC3
  - There are two .py files on BB, with a working program for 2-dimensional (2 features) fake dataset
  - That should be your starting point
- You task is to perform experiments with the two approaches on MNIST dataset
  - from keras.datasets import mnist
  - (x_train, y_train), (x_test, y_test) = mnist.load_data()
- This is 10-class classification problem:
  hand-written digits

- It has 5,000 samples per class
  - During development, you can randomly select a subset of samples, to speed up the calculations

MNIST problem

- MNIST: a 10-class classification problem
- Convert it into 2-class problem by: taking last digit of your V# (class A), taking last different digit of your V# (class B)
  - E.g. V# V00078965: 6-vs-5, V00078966: 9-vs-6
- Train both methods on the training set, then apply them to test set, and calculate the % of error predictions
  - For method A, estimate means and covariance mx of the gaussians from training data, then use the means and covariance mx to classify samples from the test set
    - how? use formula for $p(x|y=A)$, and $p(x|y=B)$, and pick class with higher probability
  - For method B, estimate w,b from training data, then use w,b to classify samples from the test set
- Try the same for smaller training set (e.g. using only random 50%, 10%, 5% of training samples), but always use the same full test set
  - see if/how much the error rate on the test set increases

Returning the Assignment

- Solution code should be written by you and you only (no web/book/friend/etc. code)
  - You can freely use the code provided on BB as your starting point

- Upload through Blackboard
  - A report in PDF
    - Results of tests of the two methods on MNIST dataset, for decreasing training set sizes (include you V#, and what are your two digits defining the two-class problem).
  - Code in python for solving the MNIST classification problem (for full size of the training set):
    - FamilyNameFirstName-PartA.py
    - FamilyNameFirstName-PartB.py
      - The files should have your name in a comment at the top