COMP4702/COMP7703 - Machine Learning

Prac W3 – Regression and Parametric models

Aims:

- To gain some experience in performing regression with linear and polynomial models and classification with parametric models.
- To produce some assessable work for this subject.

Procedure:

Use a programming languages of your choice for this prac. Matlab or python is recommended (tutors can support these languages). There are similar functions in python compared to the Matlab functions mentioned below.

Polynomial Regression

→ Refer to the example presented in Fig.4.5 of the Alpaydin book. To start, we will recreate this example.
  - Plot the function (f(x) = 2sin(1.5x)) in the range shown.
  - Create a "sample training set" of 20 points as indicated in the text. That is, generate a random set of x-values, and corresponding outputs by adding Gaussian random noise (randn() in Matlab) to the function above. Plot the data over your sine wave plot (use the "hold on" command).

→ Now we will fit some models to this data. To implement linear and polynomial regression, Matlab has a curve fitting toolbox and curve fitting tool – to run it type "cftool". Refer to the Matlab help to learn how to use it.

→ Fit polynomials of increasing order to the data and observe the models fitting the data and the output in the results box on the cftool window.

→ **Q1:** Produce a plot of error as a function of model order up to 9, similar to that shown in Fig.4.7 of the Alpaydin book. To do this you will need to create a second (*validation*) dataset (in addition to the *training* set above). Some hints (for Matlab):
  - The cftool results show you the SSE (sum of squares error = N*MSE where N is the number of data points). It also shows the trained model (polynomial with coefficients). You can paste this model back into the Matlab command line (or a script or function) to calculate validation set error.
  - Alternatively, the polyfit() and polyval() functions may be of use. Read their help files to learn more.

→ Observe the effect of overfitting as the model complexity increased while you trained the models. To see this clearly, you might need to increase the variance of the additive noise to the function when generating the outputs of your training data (the book uses variance of 0.1).

Boston Housing Dataset

In the practical materials, you will find a copy of the widely-used “Boston housing dataset”. It is part of the UCI machine learning repository:

https://archive.ics.uci.edu/ml/index.html

→ **Q2:** Using column 6 as your input variable and column 14 as your target variable, fit linear and polynomial models of varying degree to this data. Produce a table of error values for models up to degree 5.

→ **Q3:** Split the dataset into two halves. Repeat Q2 using one half of the dataset for training and evaluate the error on the other half of the data. Write a few sentences comparing any differences. Can you offer some explanation of these differences (a paragraph or two)? Relate this to key concepts of supervised learning from lectures.

Parametric Probabilistic Classification

Write a function or script that implements a simple parametric classifier to produce plots similar to Figs 4.2 and 4.3 of Alpaydin. More specifically:

→ Your code should take as input data for a one-dimensional, two-class classification problem, assuming a real-valued input feature.

→ You should use Gaussian (Normal) distributions as your model; i.e. fit a Gaussian to each class distribution using maximum likelihood estimates for the parameters of each Gaussian. This produces a model for p(x|C1) and for p(x|C2).

→ Assume equal priors for the class distributions (P(C1) and P(C2)). In this case, the posteriors are just p(C1|x) and p(C2|x) and are equal to the discriminants g(x) for each class.

→ To test your code, create a “reduced” version of the Iris dataset (see Prac 1 – a plain text version of this dataset can be found in the practical materials); firstly by discarding all the features from the data except the first one (sepal length), secondly by discarding all examples for the third class (Iris-virginica).

→ Produce histograms (e.g. using the Matlab histogram() function) to give you a picture of the data and something to compare you results with.

→ **Q4:** Use your code to produce plots of the likelihoods and class posteriors for the reduced Iris data.