Lab 1: Linear Regression and Overfitting

Machine Learning and Pattern Recognition, September 2013

- The lab exercises should be made in groups of three people, or at most two people.
- The deadline is september 22nd (sunday) 23:59.
- Assignment should be sent to N.Hu@uva.nl (Ninhang Hu). The subject line of your email should be "#lab_lastname1_lastname2_lastname3".
- Put your and your teammates' names in the body of the email
- Attach the .IPYNB (IPython Notebook) file containing your code and answers. Naming of the file follows the same rule as the subject line. For example, if the subject line is "lab01_Kingma_Hu", the attached file should be "lab01_Kingma_Hu.ipynb". Only use underscores ("_") to connect names, otherwise the files cannot be parsed.

Notes on implementation:

- You should write your code and answers in an IPython Notebook: http://ipython.org/notebook.html. If you have problems, please contact us.
- Among the first lines of your notebook should be "%pylab inline". This imports all required modules, and your plots will appear inline.
- For this lab, your regression solutions should be in closed form, i.e., should not perform iterative gradient-based optimization but find the exact optimum directly.
- NOTE: Make sure we can run your notebook / scripts!

Part 1: Polynomial Regression

1.1. Generate sinusoidal data

Write a method `gen_sinusoidal(N)` that generates toy data like in fig 1.2 of the MLPR book. The method should have a parameter N, and should return N-dimensional vectors $\mathbf{x}$ and $\mathbf{t}$, where $\mathbf{x}$ contains evenly spaced values from 0 to (including) 2π, and the elements t_i of $\mathbf{t}$ are distributed according to:

$$t_i \sim \mathcal{N}(\mu, \sigma^2)$$

where x_i is the i-th elements of $\mathbf{x}$, the mean $\mu = sin(x_i)$ and the standard deviation $\sigma = 0.2$.

1.2 Polynomial regression

Write a method `fit_polynomial(x, t, M)` that finds the maximum-likelihood solution of an *unregularized* M-th order polynomial for some dataset `x`. The error function to minimize w.r.t. $\mathbf{w}$ is:

$$E(\mathbf{w}) = \frac{1}{2} (\mathbf{\Phi w} - \mathbf{t})^T (\mathbf{\Phi w} - \mathbf{t})$$

where $\mathbf{\Phi}$ is the *feature matrix* (or *design matrix*) as explained in the MLPR book, $\mathbf{t}$ is the vector of target values. Your method should return a vector $\mathbf{w}$ with the maximum-likelihood parameter estimates.

1.3 Plot

Sample a dataset with $N = 9$, and fit four polynomials with $M \in (0, 1, 3, 9)$. For each value of M, plot the prediction function, along with the data and the original sine function. The resulting figure should look similar to fig 1.4 of the MLPR book. Note that you can use matplotlib's `plt.pyplot(.)` functionality for creating grids of figures.

1.4 Regularized linear regression

Write a method `fit_polynomial_reg(x, t, M, lamb)` that fits a *regularized* M-th order polynomial to the sinusoidal data, as discussed in the lectures, where `lamb` is the regularization term *lambda*. (Note that 'lambda' cannot be used as a variable name in Python since it has a special meaning). The error function to minimize w.r.t. $\mathbf{w}$:

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{\Phi w} - \mathbf{t})^T(\mathbf{\Phi w} - \mathbf{t}) + \frac{\lambda}{2}\,\mathbf{w}^T\mathbf{w}$$

For background, see section 3.1.4 of the MLPR book.

1.5 Model selection by cross-validation

Use cross-validation to find a good choice of M and λ, given a dataset of $N = 9$ datapoints generated with `gen_sinusoidal(9)`. You should write a function that tries (loops over) a reasonable range of choices of M and λ, and returns the choice with the best cross-validation error. In this case you can use $K = 9$ folds, corresponding to *leave-one-out* crossvalidation.

You can let $M \in (0, 1, \ldots, 10)$, and let $\lambda \in (e^{-10}, e^{-9}, \ldots, e^{0})$.

To get you started, here's a method you can use to generate indices of cross-validation folds.

```
In [ ]: def kfold_indices(N, k):
            all_indices = np.arange(N,dtype=int)
            np.random.shuffle(all_indices)
            idx = np.floor(np.linspace(0,N,k+1))
            train_folds = []
            valid_folds = []
            for fold in range(k):
                valid_indices = all_indices[idx[fold]:idx[fold+1]]
                valid_folds.append(valid_indices)
                train_folds.append(np.setdiff1d(all_indices, valid_indices))
            return train_folds, valid_folds
```

Create a comprehensible plot of the cross-validation error for each choice of M and λ. Highlight the best choice.

Question: Explain over-fitting and underfitting, illuminated by your plot. Explain the relationship with model bias and model variance.

1.6 Plot best cross-validated fit

For some dataset with $N = 9$, plot the model with the optimal M and λ according to the cross-validation error, using the method you just wrote. Let the plot make clear which M and λ were found.

Part 2: Bayesian Linear (Polynomial) Regression

2.1 Sinusoidal data 2

Write a function `gen_sinusoidal2(N)` that behaves identically to `gen_sinusoidal(N)` except that the generated values x_i are not linearly spaced, but drawn from a uniform distribution between 0 and 2π.

2.2 Compute Posterior

You're going to implement a Bayesian linear regression model, and fit it to the sinusoidal data. Your regression model has a zero-mean isotropic Gaussian prior over the parameters, governed by a single (scalar) precision parameter α, i.e.:

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

The covariance and mean of the posterior are given by:

$$\mathbf{S}_N = (\alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}$$
$$\mathbf{m}_N = \beta\ \mathbf{S}_N\mathbf{\Phi}^T\mathbf{t}$$

where α is the precision of the predictive distribution. See MLPR chapter 3.3 for background.

Write a method `fit_polynomial_bayes(x, t, M, alpha, beta)` that returns the mean $\mathbf{m}_N$ and covariance $\mathbf{S}_N$ of the posterior for a M-th order polynomial, given a dataset, where `x`, `t` and `M` have the same meaning as in question 1.2.

2.3 Prediction

The predictive distribution of Bayesian linear regression is:

$$p(t \mid \mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t \mid \mathbf{m}_N^T\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

$$\sigma_N^2 = \frac{1}{\beta} + \phi(\mathbf{x})^T\mathbf{S}_N\phi(\mathbf{x})$$

where $\phi(\mathbf{x})$ are the computed features for a new datapoint $\mathbf{x}$, and t is the predicted variable for datapoint $\mathbf{x}$.

Write a function that `predict_polynomial_bayes(x, m, S, beta)` that returns the predictive mean and variance given a new datapoint `x`, posterior mean `m`, posterior variance `S` and a choice of model variance `beta`.

2.4 Plot predictive distribution

a) Generate 7 datapoints with `gen_sinusoidal2(7)`. Compute the posterior mean and covariance for a Bayesian polynomial regression model with $M = 5$, $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{0.2^2}$. Plot the Bayesian predictive distribution, where you draw you plot (for x between 0 and 2π) t's predictive mean and a 1-sigma predictive variance using `plt.fill_between(..., alpha=0.1)` (the alpha argument induces transparency).

Include the datapoints in your plot.

b) For a second plot, draw 100 samples from the parameters' posterior distribution. Each of these samples is a certain choice of parameters for 5-th order polynomial regression. Display each of these 100 polynomials.

2.5 Additional questions

a) Why is $\beta = \frac{1}{0.2^2}$ the best choice of β in section 2.4?

b) In the case of Bayesian linear regression, both the posterior of the parameters $p(\mathbf{w} \mid \mathbf{t}, \alpha, \beta)$ and the predictive distribution $p(t \mid \mathbf{w}, \beta)$ are Gaussian. In consequence (and conveniently), $p(t \mid \mathbf{t}, \alpha, \beta)$ is also Gaussian (See MLPR section 3.3.2 and homework 2 question 4). This is actually one of the (rare) cases where we can make Bayesian predictions without resorting to approximative methods.

Suppose you have to work with some model $p(t \mid x, \mathbf{w})$ with parameters $\mathbf{w}$, where the posterior distribution $p(\mathbf{w} \mid \mathcal{D})$ given dataset $\mathcal{D}$ can not be integrated out when making predictions, but where you can still generate samples from the posterior distribution of the parameters. Explain how you can still make approximate Bayesian predictions using samples from the parameters' posterior distribution.