Lab 1A: Linear Regression and Overfitting

NOTE: Part B will be handed out next week.

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.
- Part A and B should be handed in together.
- 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.

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_indic
    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.