IAML: Regularization and Ridge Regression

Nigel Goddard and Victor Lavrenko
School of Informatics

Semester 1

Regularization

- Regularization is a general approach to add a “complexity parameter” to a learning algorithm. Requires that the **model** parameters be continuous. (i.e., Regression OK, Decision trees not.)
- If we penalize polynomials that have large values for their coefficients we will get less wiggly solutions

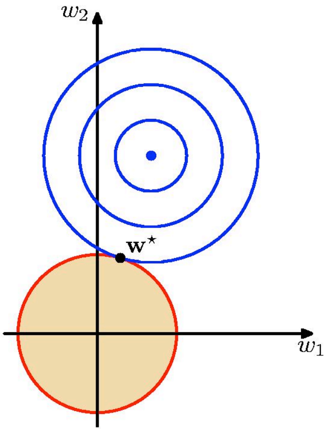
$$\tilde{E}(\mathbf{w}) = |\mathbf{y} - \Phi\mathbf{w}|^2 + \lambda|\mathbf{w}|^2$$

- Solution is

$$\hat{\mathbf{w}} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$$

- This is known as *ridge regression*
- Rather than using a discrete control parameter like M (model order) we can use a continuous parameter λ
- Caution: Don’t shrink the bias term! (The one that corresponds to the all 1 feature.)

Regularized Loss Function



- The overall cost function is the sum of two parabolic bowls. The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The regularizer just shrinks the weights.

Credit: Geoff Hinton

The effect of regularization for $M = 9$

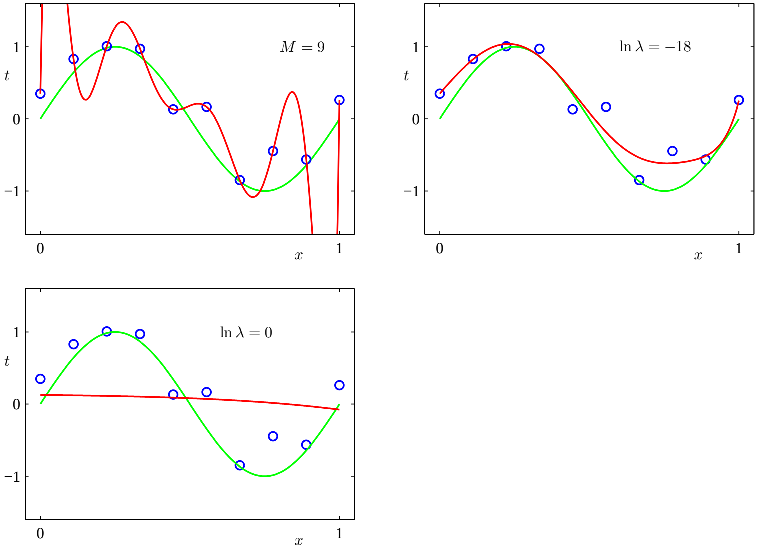
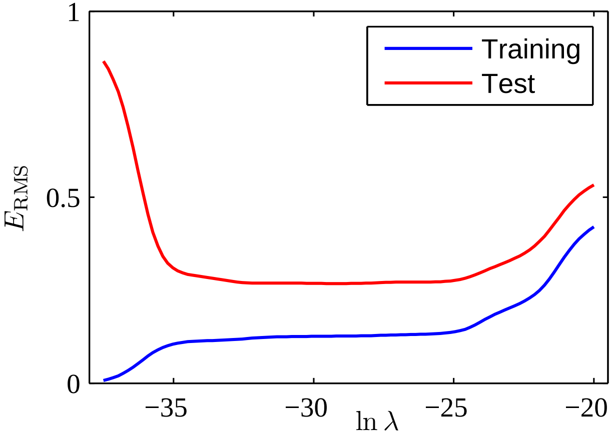


Figure credit: Chris Bishop, PRML

$M = 9$



Chris Bishop, PRML

For standard linear regression, we had

- Define the **task**: regression
- Decide on the **model structure**: linear regression model
- Decide on the **score function**: squared error (likelihood)
- Decide on **optimization/search method** to optimize the score function: calculus (analytic solution)

But with ridge regression we have

- Define the **task**: regression
- Decide on the **model structure**: linear regression model
- Decide on the **score function**: squared error **with quadratic regularizaton**
- Decide on **optimization/search method** to optimize the score function: calculus (analytic solution)

Notice how you can train the same model structure with different score functions. This is the first time we have seen this. This is important.

A Control-Parameter-Setting Procedure

- Regularization was a way of adding a “capacity control” parameter.
- But how do we set the value? e.g., the regularization parameter λ
- Won’t work to do it on the training set (why not?)
- Two choices to consider:
  - Validation set
  - Cross-validation

Using a validation set

- Split the labelled data into a training set, validation set, and a test set.
- Training set: Use for training
- Validation set: Tune the “control parameters” according to performance on the validation set
- Test set: to check how the final model performs
- No right answers, but for example, could choose 60% training, 20% validation, 20% test

Example of using a validation set

Consider polynomial regression:

1. For each $m = 1, 2, \ldots M$ (you choose M in advance
2. Train the polynomial regression using $\phi(x) = (1, x, x^2, \ldots, x_m)^T$ on training set (e.g., by minimizing squared error). This produces a predictor $f_m(\mathbf{x})$.
3. Measure the error of f_m on the validation set
4. End for
5. Choose the f_m with the best validation error.
6. Measure the error of f_m on the test set to see how well you should expect it to perform

Continuous Control Parameters

- For a discrete control parameter like polynomial order m we could simply search all values.
- What about a quadratic regularization parameter λ. What do we do then?

Continuous Control Parameters

- For a discrete control parameter like polynomial order *m* we could simply search all values.
- What about a quadratic regularization parameter λ. What do we do then?
- Pick a grid of values to search. In practice you want the grid to vary geometrically for this sort of parameter. e.g., Try $\lambda \in \{0.01, 0.1, 0.5, 1.0, 5.0, 10.0\}$. Don't bother trying 2.0, 3.0, 7.0.