Nonlinear prediction and generalization
Outline of the lecture

This lecture will teach you how to fit nonlinear functions by using bases functions and how to control model complexity. The goal is for you to:

- Learn **polynomial regression**.
- Understand that, if the bases are given, the problem of learning the parameters is still linear.
- Understand the effect of the regularizer on function **smoothness**.
- Learn about **RBFs** and **kernel regression**.
- Learn to choose the regularization coefficient by **cross-validation**.
- Learn about training error and validation error.
- Understand the effects of the number of data and the number of basis functions on **generalization**.
Going nonlinear via basis functions

We introduce basis functions $\phi(\cdot)$ to deal with nonlinearity:

$$y(x) = \phi(x)\theta + \epsilon$$

For example, $\phi(x) = [1, x, x^2]$
Going nonlinear via basis functions

$$y(x) = \phi(x)\theta + \epsilon$$

$$\phi(x) = [1, x_1, x_2] \quad \phi(x) = [1, x_1, x_2, x_1^2, x_2^2]$$
Example: Ridge regression with a polynomial of degree 14

\[ \hat{y}(x_i) = 1 \theta_0 + x_i \theta_1 + x_i^2 \theta_2 + \ldots + x_i^{13} \theta_{13} + x_i^{14} \theta_{14} \]

\[ \Phi = \begin{bmatrix} 1 & x_i & x_i^2 & \ldots & x_i^{13} & x_i^{14} \end{bmatrix} \]

\[ J(\theta) = (y - \Phi \theta)^T (y - \Phi \theta) + \delta \theta^T \theta \]

- small \( \delta \)
- medium \( \delta \)
- large \( \delta \)
Kernel regression and RBFs

We can use kernels or radial basis functions (RBFs) as features:

\[ \phi(x) = [\kappa(x, \mu_1, \lambda), \ldots, \kappa(x, \mu_d, \lambda)], \quad \text{e.g.} \quad \kappa(x, \mu_i, \lambda) = e^{-\frac{1}{\lambda} \|x - \mu_i\|^2} \]

\[ \hat{y}(x_i) = \phi(x_i) \theta = 1 \theta_0 + k(x_i, \mu_1, \lambda) \theta_1 + \ldots + k(x_i, \mu_d, \lambda) \theta_d \]

^
We can choose the locations $\mu$ of the basis functions to be the inputs. That is, $\mu_i = x_i$. These basis functions are known as kernels. The choice of width $\lambda$ is tricky, as illustrated below.

**kernels**

Too small $\lambda$

Right $\lambda$

Too large $\lambda$
The big question is how do we choose the regularization coefficient, the width of the kernels or the polynomial order?
Solution: **cross-validation**

training data \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) \(= (x, y)\) \(\Rightarrow \) choose \(S=2\)

test data \((x_{n+1}, y_{n+1}), \ldots, (x_{n+k}, y_{n+k})\) \(= (x^*, y^*)\)

\[
\hat{\theta}_r = (X^T X + 2^E I)^{-1} X^T y
\]

\[
\hat{y}_{\text{train}} = X \hat{\theta}_r
\]

\[
\hat{y}_{\text{test}} = X^* \hat{\theta}_r
\]

<table>
<thead>
<tr>
<th>(S^2)</th>
<th>Train error (\sum_{i=1}^{n} (y_i - \hat{y}_i)^2)</th>
<th>Test error (\sum_{i=n+1}^{n+k} (y_i - \hat{y}_i)^2)</th>
<th>max</th>
<th>min-max</th>
<th>aog</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>2</td>
<td>100</td>
<td>(S^2 = 1)</td>
<td>51</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>(\sqrt{10} = 3.16)</td>
<td>(21/2 = 10.5)</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>19</td>
<td>19</td>
<td></td>
<td>10 (\checkmark) (S^2 = 10)</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td></td>
<td>10 (\checkmark)</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
<td></td>
<td>1100/2</td>
</tr>
</tbody>
</table>
The idea is simple: we split the training data into $K$ folds; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the $k$’th, and test on the $k$’th, in a round-robin fashion.

It is common to use $K = 5$; this is called 5-fold CV.

If we set $K = N$, then we get a method called leave-one out cross validation, or LOOCV, since in fold $i$, we train on all the data cases except for $i$, and then test on $i$. 
Example: Ridge regression with polynomial of degree 14

The larger $\delta$, the larger the train set error. However, the test set error improves. For good generalization we want to do well in all test sets.
Effect of data when we have the right model

\[ y_i = \theta_0 + x_i \theta_1 + x_i^2 \theta_2 + \mathcal{N}(0, \sigma^2) \]

\[ \hat{y} = \hat{\theta}_0 + x \hat{\theta}_1 + x \hat{\theta}_2 \]

truth=degree 2, model = degree 2

mse vs size of training set
Effect of data when the model is too simple

\[ y_i = \theta_0 + x_i \theta_1 + x_i^2 \theta_2 + \mathcal{N}(0, \sigma^2) \]

\[ \hat{y} = \theta_0 + x\theta_1 \]

truth=degree 2, model = degree 1
Effect of data when the model is very complex

\[ y_i = \theta_0 + x_i \theta_1 + x_i^2 \theta_2 + \mathcal{N}(0, \sigma^2) \]
In the next lecture, we study the problem of feature selection and introduce a new form of regularizer: the $L_1$ norm.

This type of regularization is at the heart of a recent revolution in data acquisition known as compressed sensing.