Regression, generalization, and probabilistic approach

CE-717: Machine Learning
Sharif University of Technology

M. Soleymani
Fall 2015
Topics

- Beyond linear regression models
- Evaluation & model selection
- Regularization
- Probabilistic perspective for the regression problem
Recall: Linear regression (squared loss)

- **Linear regression functions**

  \[ f : \mathbb{R} \rightarrow \mathbb{R} \quad f(x; \mathbf{w}) = w_0 + w_1 x \]

  \[ f : \mathbb{R}^d \rightarrow \mathbb{R} \quad f(x; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d \]

  \[ \mathbf{w} = [w_0, w_1, \ldots, w_d]^T \] are the parameters we need to set.

- **Minimizing the squared loss for linear regression**

  \[ J(\mathbf{w}) = \frac{1}{2} \| \mathbf{y} - \mathbf{X} \mathbf{w} \|_2^2 \]

- **We obtain** \( \hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \)
Beyond linear regression

- How to extend the linear regression to non-linear functions?
  - Transform the data using basis functions
  - Learn a linear regression on the new feature vectors (obtained by basis functions)
Beyond linear regression

- $m^{th}$ order polynomial regression (univariate $f : \mathbb{R} \rightarrow \mathbb{R}$)
  \[ f(x; w) = w_0 + w_1 x + \ldots + w_{m-1} x^{m-1} + w_m x^m \]

- Solution: \( \hat{w} = (X'^T X')^{-1} X'^T y \)

\[
\begin{align*}
y &= \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \\
X' &= \begin{bmatrix} 1 & x^{(1)1} & x^{(1)2} & \ldots & x^{(1)m} \\
1 & x^{(2)1} & x^{(2)2} & \ldots & x^{(2)m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x^{(n)1} & x^{(n)2} & \ldots & x^{(n)m} \end{bmatrix} \\
w &= \begin{bmatrix} \hat{w}_0 \\ \hat{w}_1 \\ \vdots \\ \hat{w}_m \end{bmatrix}
\end{align*}
\]
Polynomial regression: example

$m = 1$

$m = 3$

$m = 5$

$m = 7$
Generalized linear

- Linear combination of fixed non-linear function of the input vector

\[ f(x; w) = w_0 + w_1 \phi_1(x) + \ldots + w_m \phi_m(x) \]

\{\phi_1(x), \ldots, \phi_m(x)\}: set of basis functions (or features)

\[ \phi_i(x): \mathbb{R}^d \to \mathbb{R} \]
Basis functions: examples

- **Linear**

  \[ f(x; w) = w_0 + w_1 x_1 + \ldots + w_d x_d \]

- **Polynomial (univariate)**

  \[ f(x; w) = w_0 + w_1 x + \ldots + w_{m-1} x^{m-1} + w_m x^m \]
Basis functions: examples

- **Gaussian**: \( \phi_j(x) = \exp \left\{ -\frac{(x-c_j)^2}{2\sigma_j^2} \right\} \)

- **Sigmoid**: \( \phi_j(x) = \sigma \left( \frac{\|x-c_j\|}{\sigma_j} \right) \quad \sigma(a) = \frac{1}{1+\exp(-a)} \)
Radial Basis Functions: prototypes

- Predictions based on similarity to “prototypes”:

\[ \phi_j(x) = \exp \left\{ -\frac{1}{2\sigma_j^2} \| x - c_j \|^2 \right\} \]

- Measuring the similarity to the prototypes \( c_1, \ldots, c_m \)
  - \( \sigma^2 \) controls how quickly the basis function vanishes as a function of the distance to the prototype.
  - Training examples themselves could serve as prototypes
Generalized linear: optimization

\[ J(w) = \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]

\[ = \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - w^T \phi(x^{(i)}) \right)^2 \]

\[
\begin{align*}
y &= \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \\
\Phi &= \begin{bmatrix} 1 & \phi_1(x^{(1)}) & \cdots & \phi_m(x^{(1)}) \\ 1 & \phi_1(x^{(2)}) & \cdots & \phi_m(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x^{(n)}) & \cdots & \phi_m(x^{(n)}) \end{bmatrix} \\
w &= \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}
\end{align*}
\]

\[
\hat{w} = (\Phi^T \Phi)^{-1} \Phi^T y
\]
Model complexity and overfitting

- With limited training data, models may achieve zero training error but a large test error.

\[ \frac{1}{n} \sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; \theta) \right)^2 \approx 0 \]

\[ E_{x,y} \left\{ (y - f(x; \theta))^2 \right\} \gg 0 \]

- **Over-fitting**: when the training loss no longer bears any relation to the test (generalization) loss.

- Fails to generalize to unseen examples.
How to evaluate the learner’s performance?

- Generalization error: true (or expected) error that we would like to optimize

- Two ways to assess the generalization error is:
  - Practical: Use a separate data set to test the model
  - Theoretical: Law of Large numbers
    - statistical bounds on the difference between training and expected errors
Polynomial regression

\[ m = 0 \]

\[ m = 1 \]

\[ m = 3 \]

\[ m = 9 \]
Polynomial regression: training and test error

\[ RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}; \theta))^2}{n}} \]

[Bishop]
Over-fitting causes

- Model complexity
  - E.g., Model with a large number of parameters (degrees of freedom)

- Low number of training data
  - Small data size compared to the complexity of the model
Model complexity

- Example:
  - Polynomials with larger $m$ are becoming increasingly tuned to the random noise on the target values.

\[ m = 0 \]
\[ m = 1 \]
\[ m = 3 \]
\[ m = 9 \]
Over-fitting problem becomes less severe as the size of training data increases.

\[ m = 9 \]

\[ n = 15 \]

\[ \text{[Bishop]} \]
Evaluation and model selection

**Evaluation:**
- After training a learning algorithm, we need to measure how well the learned prediction function can predict the target for unseen examples.

**Model selection:**
- Most of the time we need to select among a set of models (e.g., polynomials with different degree $m$) and thus we need to evaluate these models first.
Avoiding over-fitting

- Determine a suitable value for model complexity
  - **Simple method**: Hold some data out of the training set called validation set
    - Use held-out data to optimize model complexity
  - **Cross-validation, bootstrap**

- Regularization (Occam’s Razor)
  - Explicit preference towards simple models
  - Penalize for the model complexity in the objective function

- Bayesian inference
Simple hold-out: model selection

Steps:

- Divide training data into training and validation set \( v_{set} \)
- Use only the training set to train a set of models
- Evaluate each learned model on the validation set
  \[ J_v(w) = \frac{1}{|v_{set}|} \sum_{i \in v_{set}} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]
- Choose the best model based on the validation set error

Usually, too wasteful of valuable training data

- Training data may be limited.
- On the other hand, small validation set give a relatively noisy estimate of performance.
Simple hold out: training, validation, and test sets

- Simple hold-out chooses the model that minimizes error on validation set.

- $J_v(\hat{w})$ is likely to be an optimistic estimate of generalization error.
  - extra parameter (e.g., degree of polynomial) is fit to this set.

- Estimate generalization error for the test set
  - performance of the selected model is finally evaluated on the test set

<table>
<thead>
<tr>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
</table>
Cross-Validation (CV): Evaluation

- **$k$-fold cross-validation steps:**
  - Shuffle the dataset and randomly partition training data into $k$ groups of approximately equal size
  - for $i = 1$ to $k$
    - Choose the $i$-th group as the held-out validation group
    - Train the model on all but the $i$-th group of data
    - Evaluate the model on the held-out group
  - Performance scores of the model from $k$ runs are **averaged**.
    - The average error rate can be considered as an estimation of the true performance.

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

First run

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Second run

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

(k-1)th run

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

k-th run

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Cross-Validation (CV): Model Selection

- $k$-fold cross-validation for model selection:
  - Shuffle the dataset and randomly partition training data into $k$ groups of approximately equal size
  - for $i = 1$ to $k$
    - Choose the $i$-th group as the held-out group
    - Train the set of models on all but the $i$-th group
    - Evaluate models on the held-out group
  - Performance scores of each model from $k$ runs are averaged.
  - The model with the best average performance is selected.
Leave-One-Out Cross Validation (LOOCV)

- When data is particularly scarce, cross-validation with $k = N$
  - Leave-one-out treats each training sample in turn as a test example and all other samples as the training set.

- Use for small datasets
  - When training data is valuable
  - LOOCV can be time expensive as $N$ training steps are required.
Bootstrap

- **Bootstrap sampling:** Samples the given dataset $N$ times uniformly with replacement (resulting in a set of $N$ samples)
  - Some samples in the original set may be included several times in the bootstrap sampled data
  - The remaining samples that has not appeared in the bootstrap sampled set are considered as the validation set.

- **Bootstrap technique for evaluation of a model:**
  - Use the above bootstrap sampling to generate $M$ training (and validation) sets
  - For each of these generated data, train the model on the training set and evaluate its performance on the corresponding validation set
  - The obtained results on the validation sets are then **averaged** to find the bootstrap estimate of performance.
Cross-validation: polynomial regression example

- 5-fold CV
- 100 runs
  - average

![Graph](image)

$m = 1$
CV: $MSE = 0.30$

$m = 3$
CV: $MSE = 1.45$

$m = 5$
CV: $MSE = 45.44$

$m = 7$
CV: $MSE = 31759$
Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

- Ridge regression (weight decay):

\[
J(w) = \sum_{i=1}^{n} \left( y^{(i)} - w^T \phi(x^{(i)}) \right)^2 + \lambda w^T w
\]

\[
\hat{w} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y
\]
Polynomial order

- Polynomials with larger $m$ are becoming increasingly tuned to the random noise on the target values.
  - magnitude of the coefficients typically gets larger by increasing $m$.

\[
\begin{array}{|l|cccc|}
\hline
 & M = 0 & M = 1 & M = 6 & M = 9 \\
\hline
w_0^* & 0.19 & 0.82 & 0.31 & 0.35 \\
& & -1.27 & 7.99 & 232.37 \\
w_1^* & -25.43 & -5321.83 & & \\
w_2^* & 17.37 & & 48568.31 & \\
w_3^* & & -231639.30 & & \\
w_4^* & & 640042.26 & & \\
w_5^* & & -1061800.52 & & \\
w_6^* & & 1042400.18 & & \\
w_7^* & & -557682.99 & & \\
w_8^* & & 125201.43 & & \\
w_9^* & & & & \\
\hline
\end{array}
\]

[Bishop]
## Regularization parameter

**$m = 9$**

<table>
<thead>
<tr>
<th>$\hat{w}_0$</th>
<th>$\hat{w}_1$</th>
<th>$\hat{w}_2$</th>
<th>$\hat{w}_3$</th>
<th>$\hat{w}_4$</th>
<th>$\hat{w}_5$</th>
<th>$\hat{w}_6$</th>
<th>$\hat{w}_7$</th>
<th>$\hat{w}_8$</th>
<th>$\hat{w}_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_1$</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_2$</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_3$</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_4$</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_5$</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_6$</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_7$</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_8$</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\hat{w}_9$</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

[Bishop]
Regularization parameter

- Generalization
  - $\lambda$ now controls the effective complexity of the model and hence determines the degree of over-fitting

![Graph showing $E_{\text{RMS}}$ vs. $\ln \lambda$ for training and test data.](image-url)
Choosing the regularization parameter

- A set of models with different values of $\lambda$.

- Find $\hat{\mathbf{w}}$ for each model based on training data

- Find $J_v(\hat{\mathbf{w}})$ (or $J_{cv}(\hat{\mathbf{w}})$) for each model
  
  \[ J_v(\mathbf{w}) = \frac{1}{n_v} \sum_{i \in \text{v_set}} \left( y^{(i)} - f(x^{(i)}; \mathbf{w}) \right)^2 \]

- Select the model with the best $J_v(\hat{\mathbf{w}})$ (or $J_{cv}(\hat{\mathbf{w}})$)
Model complexity: Bias-variance trade-off

- Maximum likelihood, or equivalently least squares, can lead to severe over-fitting if complex models are trained using data sets of limited size.

- A frequentist viewpoint of the model complexity issue, known as the *bias-variance trade-off*.

- We intend to discuss in the following slides:
  - Approximation-generalization tradeoff
  - The bias and variance concepts and definitions
  - And finally the bias-variance decomposition of the error
Complexity of Hypothesis Space: Example

This example has been adapted from: Prof. Andrew Ng’s slides
Complexity of Hypothesis Space: Example

Less complex $\mathcal{H}$

"Just right"

More complex $\mathcal{H}$

$w_0 + w_1 x$

$w_0 + w_1 x + w_2 x^2$

$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$
Complexity of Hypothesis Space: Example

\[ J_v(w) = \frac{1}{n_v} \sum_{i \in \text{eval\_set}} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]

\[ J_{\text{train}}(w) = \frac{1}{n_{\text{train}}} \sum_{i \in \text{train\_set}} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]
Complexity of Hypothesis Space

- **Less complex $\mathcal{H}$:**
  - $J_{\text{train}}(\hat{\mathbf{w}}) \approx J_{v}(\hat{\mathbf{w}})$ and $J_{\text{train}}(\hat{\mathbf{w}})$ is very high

- **More complex $\mathcal{H}$:**
  - $J_{\text{train}}(\hat{\mathbf{w}}) \ll J_{v}(\hat{\mathbf{w}})$ and $J_{\text{train}}(\hat{\mathbf{w}})$ is low
Regularization: Example

\[ f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \]

\[ J(\mathbf{w}) = \frac{1}{n} \left( \sum_{i=1}^{n} \left( y^{(i)} - f(x^{(i)}; \mathbf{w}) \right)^2 + \lambda \mathbf{w}^T \mathbf{w} \right) \]

- **Large \( \lambda \)**
  - (Prefer to more simple models)
  - \( w_1 = w_2 \approx 0 \)

- **Intermediate \( \lambda \)**

- **Small \( \lambda \)**
  - (Prefer to more complex models)
  - \( \lambda = 0 \)

This example has been adapted from: Prof. Andrew Ng’s slides
Choosing the regularization parameter

\[ J_v \] vs. \[ J_{\text{train}} \] for different values of \[ \lambda \].
Size of training set

\[ J_v(w) = \frac{1}{n_v} \sum_{i \in \text{val\_set}} (y^{(i)} - f(x^{(i)}; w))^2 \]

\[ J_{\text{train}}(w) = \frac{1}{n_{\text{train}}} \sum_{i \in \text{train\_set}} (y^{(i)} - f(x^{(i)}; w))^2 \]

\[ f(x; w) = w_0 + w_1 x + w_2 x^2 \]

This slide has been adapted from: Prof. Andrew Ng’s slides
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.
Less complex $\mathcal{H}$

If a learning algorithm is suffering from high variance, getting more training data is likely to help.

$$f(x; \mathbf{w}) = w_0 + w_1 x + \cdots w_{100} x^{100}$$

This slide has been adapted from: Prof. Andrew Ng's slides
The approximation-generalization trade-off

- Small true error shows good approximation of $f$ out of sample
- More complex $\mathcal{H} \Rightarrow$ better chance of approximating $f$
- Less complex $\mathcal{H} \Rightarrow$ better chance of generalization out of $f$
Formal discussion on bias, variance, and noise

- Best unrestricted regression function
- Noise
- Bias and variance
The learning diagram: deterministic target

[Y.S. Abou Mostafa, et. al, “Learning From Data”, 2012]
The learning diagram including noisy target

- **Type**

  - **Unknown Target Distribution**
    - $P(y \mid x)$
    - Target function $h: \mathcal{X} \rightarrow \mathcal{Y}$ plus noise

  - **Training Examples**
    - $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$

  - **Learning Algorithm**
    - $f(x) = h(x)$

  - **Final Hypothesis**
    - $f: \mathcal{X} \rightarrow \mathcal{Y}$

  - **Hypothesis Set**
    - $\mathcal{H}$

  - **Probability Distribution**
    - $P$ on $\mathcal{X}$

  - **Distribution on Features**
    - $P(x, y) = P(x)P(y \mid x)$

  - **Target Distribution**

[Y.S. Abou Mostafa, et. al, “Learning From Data”, 2012]
Best unrestricted regression function

- If we know the joint distribution \( P(x, y) \) and no constraints on the regression function?
  - unconstrained regression function \( h : \mathbb{R}^d \rightarrow \mathbb{R} \)
  - cost function: mean squared error
    - \( h \) minimizes the expected loss \( E_{x,y}[L(y, h(x))] \) (for the squared loss):

\[
h^* = \arg\min_{h : \mathbb{R}^d \rightarrow \mathbb{R}} E_{x,y} \left[ (y - h(x))^2 \right]
\]

\[
h^*(x) = E_{y|x}[y]
\]
Best unrestricted regression function: Proof

\[ E_{x,y} \left[ (y - h(x))^2 \right] = \iint (y - h(x))^2 p(x, y) dx dy \]

- For each \( x \) separately minimize loss since \( h(x) \) can be chosen independently for each different \( x \):

\[ \frac{\delta E_{x,y} \left[ (y - h(x))^2 \right]}{\delta h(x)} = \int 2(y - h(x))p(x, y) dy = 0 \]

\( \Rightarrow \) \( h(x) = \frac{\int yp(x, y) dy}{\int p(x, y) dy} = \frac{\int yp(x, y) dy}{p(x)} = \int yp(y|x) dy = E_{y|x}[y] \)

\( \Rightarrow h^*(x) = E_{y|x}[y] \)
Error decomposition

\[ \text{Expected loss} = E_{x,y}[(f(x; \hat{\mathbf{w}}) - y)^2] \]

\[ = E_{x,y}[(f(x; \hat{\mathbf{w}}) - h(x) + h(x) - y)^2] \]

\[ = E_x[(f(x; \hat{\mathbf{w}}) - h(x))^2] + E_{x,y}[(h(x) - y)^2] + 2E_{x,y}[(f(x; \hat{\mathbf{w}}) - h(x))(h(x) - y)] \]

\[ E_x[(f(x; \hat{\mathbf{w}}) - h(x))E_{y|x}[(h(x) - y)] \]

\[ 0 \]
Error decomposition

\[ (x, y) \sim P \]
\[ h(x) : \text{minimizes the expected loss} \]

\[
\text{Expected loss} = E_{x,y} [(f(x; \hat{w}) - y)^2]
\]
\[
= E_{x,y} [(f(x; \hat{w}) - h(x) + h(x) - y)^2]
\]
\[
= E_x [(f(x; \hat{w}) - h(x))^2] + E_{x,y} [(h(x) - y)^2]
+ 0
\]

- Noise shows the irreducible minimum value of the loss function
Bias, variance, and noise

Expected loss = \( E_{x,y} [(f(x; \hat{w}) - y)^2] \)

\[
= E_x \left[ (f(x; \hat{w}) - h(x))^2 \right] + E_{x,y} [(h(x) - y)^2]
\]

\( \hat{w} = w_D \)

\[
E_D \left\{ E_{x,y} \left[ (f(x; w_D) - h(x))^2 \right] \right\} = bias + variance
\]

\[
bias = \int (\bar{f}(x) - h(x))^2 p(x)dx
\]

\[
variance = \int E_D \left[ (f(x; w_D) - \bar{f}(x))^2 \right] p(x)dx
\]

\( \bar{f}(x) \equiv E_D [f(x; w_D)] \)

\( \bar{f}(x) \approx \frac{1}{K} \sum_{k=1}^{K} f(x; w_{D(k)}) \)

Training sets (of size \( N \)) sampled from \( P(x, y): D^{(1)}, D^{(2)}, ... \)
Regularization and bias/variance

$\lambda$ is large

$\lambda$ is intermediate

$\lambda$ is small

$L = 100$ data sets
$n = 25$
$m = 25$

[Bishop]
Learning curves of bias, variance, and noise

[Bishop]
Bias-variance trade-off

[Y.S. Abou Mostafa, et. al, “Learning From Data”, 2012]
Bias-variance decomposition: summary

- The noise term is unavoidable.
- The terms we are interested in are bias and variance.
- The approximation-generalization trade-off is seen in the bias-variance decomposition.
Curve fitting: probabilistic perspective

- Describing uncertainty over value of target variable as a probability distribution
- Example:
Curve fitting: probabilistic perspective (Example)

- Special case:
  
  Observed output = function + noise

  \[ y = f(x; \mathbf{w}) + \epsilon \]

  e.g., \( \epsilon \sim N(0, \sigma^2) \)

- Noise: Whatever we cannot capture with our chosen family of functions
Curve fitting: probabilistic perspective (Example)

- Best regression

\[ E_{y|x}[y] = E[f(x; w) + \epsilon] = f(x; w) \]

- \( f(x; w) \) is trying to capture the mean of the observations \( y \) given the input \( x \):

- \( E_{y|x}[y] \): conditional expectation of \( y \) given \( x \)
  - evaluated according to the model (not according to the underlying distribution \( P \))
Curve fitting using probabilistic estimation

- Maximum Likelihood (ML) estimation
- Maximum A Posteriori (MAP) estimation
- Bayesian approach
Maximum likelihood estimation

- Given observations $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$
- Find the parameters that maximize the (conditional) likelihood of the outputs:

$$L(\mathcal{D}; \theta) = p(y|X, \theta) = \prod_{i=1}^n p(y^{(i)}|x^{(i)}, \theta)$$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$
Maximum likelihood estimation (Cont’d)

\[ y = f(x; w) + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \]

- \( y \) given \( x \) is normally distributed with mean \( f(x; w) \) and variance \( \sigma^2 \):

  - we model the uncertainty in the predictions, not just the mean

\[ p(y|x, w, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{ -\frac{1}{2\sigma^2} (y - f(x; w))^2 \right\} \]

- Loss function?

61
Maximum likelihood estimation (Cont’d)

- Example: univariate linear function

\[ p(y|x, w, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{ -\frac{1}{2\sigma^2} (y - w_0 - w_1 x)^2 \right\} \]

Why is this line a bad fit according to the likelihood criterion?

\[ p(y|x, w, \sigma^2) \] for most of the points will be near zero (as they are far from this line)
Maximum likelihood estimation (Cont’d)

- Maximize the likelihood of the outputs (i.i.d):

\[ L(\mathcal{D}; \mathbf{w}, \sigma^2) = p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \sigma^2) \]

\[ = \prod_{i=1}^{n} p(y^{(i)} | x^{(i)}, \mathbf{w}, \sigma^2) \]

\[ \hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(\mathcal{D}; \mathbf{w}, \sigma^2) \]

\[ = \arg\max_{\mathbf{w}} p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \sigma^2) \]
Maximum likelihood estimation (Cont’d)

- It is often easier (but equivalent) to try to maximize the log-likelihood:

\[
\hat{w} = \arg\max_w \ln p(y|X, w, \sigma^2)
\]

\[
\ln p(y|X, w, \sigma^2) = \sum_{i=1}^{n} \ln \mathcal{N}(y^{(i)}|x^{(i)}, w, \sigma^2)
\]

\[
= -n \ln \sigma - \frac{n}{2} \ln 2\pi - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}; w))^2
\]

sum of squares error
Maximum likelihood estimation (Cont’d)

- Maximizing log-likelihood (when we assume \( y = f(x; w) + \epsilon \), \( \epsilon \sim N(0, \sigma^2) \)) is equivalent to minimizing SSE.

- Let \( \hat{w} \) be the maximum likelihood (here least squares) setting of the parameters.

- What is the maximum likelihood estimate of \( \sigma^2 \)?

\[
\frac{\partial \log L(D; w, \sigma^2)}{\partial \sigma^2} = 0
\]

\[
\Rightarrow \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}; \hat{w}))^2
\]

Mean squared prediction error
Generally, maximizing log-likelihood is equivalent to minimizing empirical loss when the loss is defined according to:

\[
\text{Loss} \left( y^{(i)}, f(x^{(i)}, w) \right) = -\ln p(y^{(i)}|x^{(i)}, w, \theta)
\]

- Loss: negative log-probability
  - More general distributions for \( p(y|x) \) can be considered
Maximum A Posterior (MAP) estimation

MAP:

- Given observations $\mathcal{D}$
- Find the parameters that maximize the probabilities of the parameters after observing the data (maximize the posterior probabilities):

  $$\max_{\theta} p(\theta | \mathcal{D})$$

Since $p(\theta | \mathcal{D}) \propto p(\mathcal{D} | \theta)p(\theta)$

$$\max_{\theta} p(\mathcal{D} | \theta)p(\theta)$$
Maximum A Posterior (MAP) estimation

- Given observations \( \mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n} \)
- Find the parameters that maximize the probabilities of observations

\[
\max_{\mathbf{w}} p(\mathbf{w}|X, \mathbf{y}) \propto p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})
\]

\[
p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^2 \mathbf{I}) = \left(\frac{1}{\sqrt{2\pi\alpha}}\right)^{d+1} \exp\left\{-\frac{1}{2\alpha^2} \mathbf{w}^T \mathbf{w}\right\}
\]
Maximum A Posterior (MAP) estimation

- Given observations $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
- Find the parameters that maximize the probabilities of observations

$$\max_w p(y|X, w, \sigma^2)p(w)$$

$$\min_w \frac{1}{\sigma^2} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}; w))^2 + \frac{1}{\alpha^2} w^T w$$

- Equivalent to regularized SSE with $\lambda = \frac{\sigma^2}{\alpha^2}$
Bayesian approach

- Given observations $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
- Find the parameters that maximize the probabilities of observations

$$p(y|x, \mathcal{D}) = \int p(y|w, x)p(w|\mathcal{D})dw$$

- Example of prior distribution

$$p(w) = \mathcal{N}(0, \alpha^2 I)$$
Bayesian approach

- Given observations $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
- Find the parameters that maximize the probabilities of observations

\[
p(\mathcal{D}|\mathbf{w}) = L(\mathcal{D}; \mathbf{w}, \theta) = \prod_{i=1}^{N} p(y^{(i)}|\mathbf{w}^T x^{(i)}, \theta)
\]

\[
p(y^{(i)}|f(x^{(i)}, \mathbf{w}), \theta) = \mathcal{N}(y^{(i)}|\mathbf{w}^T x^{(i)}, \sigma^2)
\]

\[
p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^2 \mathbf{I})
\]

\[
p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})
\]

Predictive distribution

\[
p(y|x, \mathcal{D}) = \int p(y|\mathbf{w}, x)p(\mathbf{w}|\mathcal{D})d\mathbf{w}
\]

\[
p(y|x, \mathcal{D}) = \mathcal{N}(\mathbf{m}_N^T x, \sigma^2_N(x))
\]
Example: Sinusoidal data, 9 Gaussian basis functions

Red curve shows the mean of the predictive distribution
Pink region spans one standard deviation either side of the mean

[Bishop]
Predictive distribution: example

- Functions whose parameters are sampled from $p(w|\mathcal{D})$