Regression II
CE-717: Machine Learning
Sharif University of Technology

M. Soleymani
Fall 2014
Topics

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

- Linear regression functions

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

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

  \( \mathbf{w} = [w_0, w_1, \ldots, w_d]^T \) are 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

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

\[
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix} = 
\begin{bmatrix}
  1 & x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\
  1 & x^{(2)} & x^{(2)} & \ldots & x^{(m)} \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  1 & x^{(n)} & x^{(n)} & \ldots & x^{(n)}
\end{bmatrix}
\begin{bmatrix}
  w_0 \\
  w_1 \\
  \vdots \\
  w_m
\end{bmatrix}
\]
Polynomial regression: example

$m = 1$

$m = 3$

$m = 5$

$m = 7$
Generalized linear

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

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

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

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

- Linear

If $m = d$, $\phi_i(x) = x_i$, $i = 1, \ldots, d$, then

$$f(x; \mathbf{w}) = w_0 + w_1x_1 + \ldots + w_dx_d$$

- Polynomial (univariate)

If $\phi_i(x) = x^i$, $i = 1, \ldots, m$, then

$$f(x; \mathbf{w}) = w_0 + w_1x + \ldots + w_{m-1}x^{m-1} + w_mx^m$$
Basis functions: examples

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

- Sigmoid: \( \phi_j(x) = \sigma \left( \frac{\|x-c_j\|}{\sigma_j} \right) \) \( \sigma(a) = \frac{1}{1+\exp(-a)} \)
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

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

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

\[ y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \]

\[ \Phi = \begin{bmatrix} 1 & \phi_1(x^{(1)}) & \cdots & \phi_m(x^{(1)}) \\ 1 & \phi_1(x^{(2)}) & \cdots & \phi_m(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x^{(n)}) & \cdots & \phi_m(x^{(n)}) \end{bmatrix} \]

\[ w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \]

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

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

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

Training (empirical) loss

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

Expected (test) loss

- **Over-fitting**: when the training loss no longer bears any relation to the test (generalization) loss.
- Fails to generalize to unseen examples.
Polynomial regression

\[ m = 0 \]

\[ m = 1 \]

\[ m = 3 \]

\[ m = 9 \]

[Bishop]
Polynomial regression: training and test error

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

[Graph showing training and test error vs. m, with lines for each.]
How to evaluate the learner’s performance?

- Generalization error: true (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:
    - statistical bounds on the difference between training and expected errors
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.

![Graphs showing model complexity with different degrees of polynomials](image_url)
Number of training data & overfitting

- Over-fitting problem becomes less severe as the size of training data increases.

\[ m = 9 \]
\[ n = 15 \]

\[ m = 9 \]
\[ n = 100 \]

[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
- Use only the training set to train a set of models
- Evaluate each learned model on the validation set
  \[ J_v(w) = \frac{1}{n_v} \sum_{i \in v \text{ set}} \left( y^{(i)} - f(x^{(i)}; w) \right)^2 \]
- 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
  - the performance of the selected model is finally evaluated on the test set
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.
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

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

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

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

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

5-fold CV
Average on 100 runs
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 \( m \).

<table>
<thead>
<tr>
<th></th>
<th>( M = 0 )</th>
<th>( M = 1 )</th>
<th>( M = 6 )</th>
<th>( M = 9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0^* )</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>( w_1^* )</td>
<td></td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>( w_2^* )</td>
<td></td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
</tr>
<tr>
<td>( w_3^* )</td>
<td></td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
</tr>
<tr>
<td>( w_4^* )</td>
<td></td>
<td></td>
<td></td>
<td>-231639.30</td>
</tr>
<tr>
<td>( w_5^* )</td>
<td></td>
<td></td>
<td></td>
<td>640042.26</td>
</tr>
<tr>
<td>( w_6^* )</td>
<td></td>
<td></td>
<td></td>
<td>-1061800.52</td>
</tr>
<tr>
<td>( w_7^* )</td>
<td></td>
<td></td>
<td></td>
<td>1042400.18</td>
</tr>
<tr>
<td>( w_8^* )</td>
<td></td>
<td></td>
<td></td>
<td>-557682.99</td>
</tr>
<tr>
<td>( w_9^* )</td>
<td></td>
<td></td>
<td></td>
<td>125201.43</td>
</tr>
</tbody>
</table>

[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>0.35</td>
<td>232.37</td>
<td>-5321.83</td>
<td>48568.31</td>
<td>-231639.30</td>
<td>640042.26</td>
<td>-1061800.52</td>
<td>1042400.18</td>
<td>-557682.99</td>
<td>125201.43</td>
</tr>
<tr>
<td>0.35</td>
<td>4.74</td>
<td>-0.77</td>
<td>-31.97</td>
<td>-3.89</td>
<td>55.28</td>
<td>41.32</td>
<td>-45.95</td>
<td>-91.53</td>
<td>72.68</td>
</tr>
<tr>
<td>0.13</td>
<td>-0.05</td>
<td>-0.06</td>
<td>-0.05</td>
<td>-0.03</td>
<td>-0.02</td>
<td>-0.01</td>
<td>-0.00</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

$\ln \lambda = -\infty$, $\ln \lambda = -18$, $\ln \lambda = 0$

[30] Bishop
Regularization parameter

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

![Graph showing $E_{RMS}$ vs $\ln \lambda$ with two curves for training and test sets](image)
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
  
  
  \[ J_v(\mathbf{w}) = \frac{1}{n_v} \sum_{i \in v\_set} (y^{(i)} - f(x^{(i)}; \mathbf{w}))^2 \]

- Select the model with the best $J_v(\hat{\mathbf{w}})$ (or $J_{cv}(\hat{\mathbf{w}})$)
Bias-Variance: Example

This example has been adapted from: Prof. Andrew Ng’s slides
Bias-Variance: Example

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

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

- **Bias:** $J_{\text{train}}(\hat{\mathbf{w}}) \approx J_v(\hat{\mathbf{w}})$ and $J_{\text{train}}(\hat{\mathbf{w}})$ is very high

- **Variance:** $J_{\text{train}}(\hat{\mathbf{w}}) \ll J_v(\hat{\mathbf{w}})$ and $J_{\text{train}}(\hat{\mathbf{w}})$ is low

![Graph showing the relationship between degree of polynomial $m$ and error for training and validation cost functions $J_{\text{train}}(\hat{\mathbf{w}})$ and $J_v(\hat{\mathbf{w}})$]
Bias-Variance (with regularization): Example

\[ f(x; w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_4 x^4 \]

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

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

\[
J_v \quad J_{\text{train}}
\]

\[
\lambda
\]

error
Size of training set

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

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

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

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 help much.

$$f(x; w) = w_0 + w_1 x$$
If a learning algorithm is suffering from high variance, getting more training data is likely to help.
Formal discussion on bias, variance, and noise

- Best unrestricted regression function
- Noise
- Bias and variance
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 $L(y, h(x)) = (y - h(x))^2$):

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

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

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

- For each \( x \) separately minimize loss since \( h(x) \) can be chosen independently for each different \( x \): \n
\[ \frac{\delta E_{x,y} \left[ (y - h(x))^2 \right]}{\delta h(x)} = \int 2(y - h(x))p(x, y)dy = 0 \]

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

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

\( \begin{align*}
\text{Expected loss} &= E_{x,y}[(f(x; \hat{w}) - y)^2] \\
&= E_{x,y}[(f(x; \hat{w}) - h(x) + h(x) - y)^2] \\
&= E_x [(f(x; \hat{w}) - h(x))^2] + E_{x,y}[(h(x) - y)^2] \\
&\quad + 2E_{x,y}[(f(x; \hat{w}) - h(x))(h(x) - y)] \\
&= E_x [(f(x; \hat{w}) - h(x))E_y[(h(x) - y)]] \\
&\quad + 0 \\
\delta E_{x,y} \left[ (y - h(x))^2 \right] = 0
\end{align*} \)
Error decomposition

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

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

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

+ 0

- Noise shows the irreducible minimum value of the loss function
Best unrestricted regression function:

\[ f(x; w) = w^T x \]

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

\[ w^* = \arg\min_w E_{x,y} [(y - f(x; w))^2] \]

Best regression:

\[ h(x) = E_{y|x} [y] \]

\[ h \text{ minimizes } E_{x,y} [(y - h(x))^2] \]

\[ \hat{w} = w_D, D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n} \]
**Bias, variance, and noise**

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

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

\[
\hat{w} = w_D
\]

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

\[
\text{bias}^2 = \int (E_D[f(x; w_D)] - h(x))^2 p(x)dx
\]

\[
\text{variance} = \int E_D[(f(x; w_D) - E_D[f(x; w_D)])^2] p(x)dx
\]
Regularization and bias/variance

\[ L = 100 \text{ data sets} \]
\[ n = 25 \]
24 Gaussians as basis functions

\( \lambda \) is

large

\( \ln \lambda = 2.6 \)

intermediate

\( \ln \lambda = -0.31 \)

small

\( \ln \lambda = -2.4 \)

[Bishop]
Learning curves of bias and variance

[Bishop]
Curve fitting: probabilistic perspective

- Describing uncertainty over value of target variable as a probability distribution
- Example:

\[ h(x_0; w) \]

\[ h(x; w) \]
Curve fitting: probabilistic perspective (Example)

- Special case:
  
  Observed output = function + noise

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

  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 selected probabilistic distribution for \( p(y|x) \) (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:

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

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \quad X = \begin{bmatrix} 1 & x^{(1)}_1 & \cdots & x^{(1)}_d \\ 1 & x^{(2)}_1 & \cdots & x^{(2)}_d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(n)}_1 & \cdots & x^{(n)}_d \end{bmatrix}$$
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?
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):

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

\[
\hat{w} = \arg\max_w L(D; w, \sigma^2)
= \arg\max_w p(y|X, w, \sigma^2)
\]
Maximum likelihood estimation (Cont’d)

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

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

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

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

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
Maximum likelihood estimation (Cont’d)

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

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

- 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

\[
\max_w p(w|X, y) \propto p(y|X, w)p(w)
\]

\[
p(w) = \mathcal{N}(0, \alpha^2 I) = \left(\frac{1}{2\pi\alpha}\right)^{(m+1)/2} \exp\left\{-\frac{1}{2\alpha^2} w^T w\right\}
\]
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

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

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

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

- Given observations $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$

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

  $$p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$$

- Example of prior distribution
  $$p(w) = \mathcal{N}(0, \alpha^2 I)$$
Bayesian approach

- Given observations $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$

Predictive distribution

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

$$p(y^{(i)}|f(x^{(i)}, w), \theta) = \mathcal{N}(y^{(i)}|f(x^{(i)}; w), \sigma^2)$$

$$p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$$

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

$$p(\mathcal{D}|w) = L(\mathcal{D}; w, \theta) = \prod_{i=1}^n p(y^{(i)}|f(x^{(i)}, w), \theta)$$
Predictive distribution: example

- 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
Predictive distribution: example

function $f(x; \mathbf{w})$ using samples from the posterior distributions over $\mathbf{w}$

[Graphs of function $f(x; \mathbf{w})$ with samples from posterior distributions over $\mathbf{w}$]