Linear Models for Regression
Greg Mori - CMPT 419/726

Bishop PRML Ch. 3
Outline

Regression
Linear Basis Function Models
Loss Functions for Regression
Finding Optimal Weights
Regularization
Bayesian Linear Regression
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Regression

- Given **training set** \( \{(x_1, t_1), \ldots, (x_N, t_N)\} \)
- \( t_i \) is continuous: **regression**
- For now, assume \( t_i \in \mathbb{R}, x_i \in \mathbb{R}^D \)
- E.g. \( t_i \) is stock price, \( x_i \) contains company profit, debt, cash flow, gross sales, number of spam emails sent, \ldots
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Linear Functions

- A function $f(\cdot)$ is **linear** if
  \[ f(\alpha u + \beta v) = \alpha f(u) + \beta f(v) \]

- Linear functions will lead to simple algorithms, so let’s see what we can do with them
Linear Regression

- Simplest linear model for regression

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

- Remember, we’re learning \( w \)
- Set \( w \) so that \( y(x, w) \) aligns with target value in training data
Linear Regression

- Simplest linear model for regression

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

- Remember, we’re learning \( w \)
- Set \( w \) so that \( y(x, w) \) aligns with target value in training data
**Linear Regression**

- Simplest linear model for regression

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

- Remember, we’re learning \( w \)
- Set \( w \) so that \( y(x, w) \) aligns with target value in training data
- This is a very simple model, limited in what it can do
Linear Basis Function Models

- Simplest linear model

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

was linear in \( x \) (*) and \( w \)

- Linear in \( w \) is what will be important for simple algorithms
- Extend to include fixed non-linear functions of data

\[ y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x) \]

- Linear combinations of these basis functions also linear in parameters
Linear Basis Function Models

- Simplest linear model

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

was linear in \( x \) (*) and \( w \)

- Linear in \( w \) is what will be important for simple algorithms

- Extend to include fixed non-linear functions of data

\[ y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x) \]

- Linear combinations of these basis functions also linear in parameters
Linear Basis Function Models

- Simplest linear model

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \]

was linear in \( x \) (*) and \( w \)

- Linear in \( w \) is what will be important for simple algorithms
- Extend to include fixed non-linear functions of data

\[ y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x) \]

- Linear combinations of these basis functions also linear in parameters
Linear Basis Function Models

- **Bias** parameter allows fixed offset in data

\[
y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x)
\]

- Think of simple 1-D \( x \):

\[
y(x, w) = w_0 + w_1 x
\]

- For notational convenience, define \( \phi_0(x) = 1 \):

\[
y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)
\]
Linear Basis Function Models

- **Bias** parameter allows fixed offset in data

\[
y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x)
\]

- Think of simple 1-D \( x \):

\[
y(x, w) = \underbrace{w_0}_{\text{intercept}} + \underbrace{w_1 x}_{\text{slope}}
\]

- For notational convenience, define \( \phi_0(x) = 1 \):

\[
y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)
\]
Linear Basis Function Models

- **Bias** parameter allows fixed offset in data

\[ y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x) \]

- Think of simple 1-D \( x \):

\[ y(x, w) = w_0 + w_1 x \]

- For notational convenience, define \( \phi_0(x) = 1 \):

\[ y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x) \]
Linear Basis Function Models

- Function for regression $y(x, w)$ is non-linear function of $x$, but linear in $w$:

  $$y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

- Polynomial regression is an example of this
- Order $M$ polynomial regression, $\phi_j(x) =$?
Linear Basis Function Models

- Function for regression $y(x, w)$ is non-linear function of $x$, but linear in $w$:

$$y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

- Polynomial regression is an example of this
- Order $M$ polynomial regression, $\phi_j(x) =$?
Regression \textbf{Linear Basis Function Models} \quad \text{Loss Functions for Regression} \quad \text{Finding Optimal Weights} \quad \text{Regularization} \quad \text{Bayesian Linear Regression}

\section*{Linear Basis Function Models}

- Function for regression $y(x, w)$ is non-linear function of $x$, but linear in $w$:

$$y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

- Polynomial regression is an example of this
- Order $M$ polynomial regression, $\phi_j(x) =$?
- $\phi_j(x) = x^j$:

$$y(x, w) = w_0 x^0 + w_1 x^1 + \ldots + w_M x^M$$
Basis Functions: Feature Functions

- Often we extract features from $x$
  - An intuitive way to think of $\phi_j(x)$ is as feature functions
- E.g. Automatic CMPT726 project report grading system
  - $x$ is text of report: In this project we apply the algorithm of Mori [2] to recognizing blue objects. We test this algorithm on pictures of you and I from my holiday photo collection. ...

- $\phi_1(x)$ is count of occurrences of Mori [2]
- $\phi_2(x)$ is count of occurrences of you and I
- Regression grade $y(x, w) = 20\phi_1(x) - 10\phi_2(x)$
Basis Functions: Feature Functions

- Often we extract features from $x$
  - An intuitive way to think of $\phi_j(x)$ is as feature functions
- E.g. Automatic CMPT726 project report grading system
  - $x$ is text of report: In this project we apply the algorithm of Mori [2] to recognizing blue objects. We test this algorithm on pictures of you and I from my holiday photo collection. ...
- $\phi_1(x)$ is count of occurrences of Mori [2]
- $\phi_2(x)$ is count of occurrences of you and I
- Regression grade $y(x, w) = 20\phi_1(x) - 10\phi_2(x)$
Basis Functions: Feature Functions

- Often we extract features from $x$
  - An intuitive way to think of $\phi_j(x)$ is as feature functions
- E.g. Automatic CMPT726 project report grading system
  - $x$ is text of report: In this project we apply the algorithm of Mori [2] to recognizing blue objects. We test this algorithm on pictures of you and I from my holiday photo collection. ...
  - $\phi_1(x)$ is count of occurrences of Mori [2]
  - $\phi_2(x)$ is count of occurrences of of you and I
- Regression grade $y(x, w) = 20\phi_1(x) - 10\phi_2(x)$
Basis Functions: Feature Functions

- Often we extract features from $x$
  - An intuitive way to think of $\phi_j(x)$ is as feature functions
- E.g. Automatic CMPT726 project report grading system
  - $x$ is text of report: In this project we apply the algorithm of Mori [2] to recognizing blue objects. We test this algorithm on pictures of you and I from my holiday photo collection. ...
  - $\phi_1(x)$ is count of occurrences of Mori [2]
  - $\phi_2(x)$ is count of occurrences of of you and I
- Regression grade $y(x, w) = 20\phi_1(x) - 10\phi_2(x)$
Other Non-linear Basis Functions

- **Polynomial** $\phi_j(x) = x^j$
- **Gaussians** $\phi_j(x) = \exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$
- **Sigmoidal** $\phi_j(x) = \frac{1}{1+\exp((\mu_j-x)/s)}$
Example - Gaussian Basis Functions: Temperature

- Use Gaussian basis functions, regression on temperature
Example - Gaussian Basis Functions: Temperature

- $\mu_1 = \text{Vancouver}$, $\mu_2 = \text{San Francisco}$, $\mu_3 = \text{Oakland}$
Example - Gaussian Basis Functions: Temperature

• $\mu_1 =$ Vancouver, $\mu_2 =$ San Francisco, $\mu_3 =$ Oakland

• Temperature in $x =$ Seattle? $y(x, w) =$

$$w_1 \exp\left\{ -\frac{(x-\mu_1)^2}{2s^2} \right\} + w_2 \exp\left\{ -\frac{(x-\mu_2)^2}{2s^2} \right\} + w_3 \exp\left\{ -\frac{(x-\mu_3)^2}{2s^2} \right\}$$
Example - Gaussian Basis Functions: Temperature

- $\mu_1 = \text{Vancouver}, \mu_2 = \text{San Francisco}, \mu_3 = \text{Oakland}$
- Temperature in $x = \text{Seattle}$? 
  $$y(x, w) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}$$
- Compute distances to all $\mu$, $y(x, w) \approx w_1$
Example - Gaussian Basis Functions: 726 Report Grading

• Define:
  • $\mu_1 = \text{Crime and Punishment}$
  • $\mu_2 = \text{Animal Farm}$
  • $\mu_3 = \text{Some paper by Mori}$

• Learn weights:
  • $w_1 = ?$
  • $w_2 = ?$
  • $w_3 = ?$

• Grade a project report $x$:
  • Measure similarity of $x$ to each $\mu$, Gaussian, with weights:
    $$y(x, w) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}$$

• The Gaussian basis function models end up similar to template matching
Example - Gaussian Basis Functions: 726 Report Grading

- Define:
  - \( \mu_1 = \) Crime and Punishment
  - \( \mu_2 = \) Animal Farm
  - \( \mu_3 = \) Some paper by Mori

- Learn weights:
  - \( w_1 = ? \)
  - \( w_2 = ? \)
  - \( w_3 = ? \)

- Grade a project report \( x \):
  - Measure similarity of \( x \) to each \( \mu \), Gaussian, with weights:
    \[
    y(x, w) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}
    \]
  - The Gaussian basis function models end up similar to template matching
Example - Gaussian Basis Functions: 726 Report Grading

- Define:
  - $\mu_1 = \text{Crime and Punishment}$
  - $\mu_2 = \text{Animal Farm}$
  - $\mu_3 = \text{Some paper by Mori}$

- Learn weights:
  - $w_1 = ?$
  - $w_2 = ?$
  - $w_3 = ?$

- Grade a project report $x$:
  - Measure similarity of $x$ to each $\mu$, Gaussian, with weights:
    \[
    y(x, w) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}
    \]
  - The Gaussian basis function models end up similar to template matching
Example - Gaussian Basis Functions

- Could define $\phi_j(x) = \exp\left\{-\frac{(x-x_j)^2}{2s^2}\right\}$
  - Gaussian around each training data point $x_j$, $N$ of them
- Could use for modelling temperature or resource availability at spatial location $x$
- Overfitting - interpolates data
- Example of a kernel method, more later
Example - Gaussian Basis Functions

- Could define $\phi_j(x) = \exp\left\{-\frac{(x-x_j)^2}{2s^2}\right\}$
  - Gaussian around each training data point $x_j$, $N$ of them
- Could use for modelling temperature or resource availability at spatial location $x$
- Overfitting - interpolates data
- Example of a kernel method, more later
Example - Gaussian Basis Functions

- Could define \( \phi_j(x) = \exp\left\{-\frac{(x-x_j)^2}{2s^2}\right\} \)
  - Gaussian around each training data point \( x_j \), \( N \) of them
- Could use for modelling temperature or resource availability at spatial location \( x \)
- Overfitting - interpolates data
- Example of a kernel method, more later
Example - Gaussian Basis Functions

- Could define \( \phi_j(x) = \exp\left\{-\frac{(x-x_j)^2}{2s^2}\right\} \)
  - Gaussian around each training data point \( x_j \), \( N \) of them
- Could use for modelling temperature or resource availability at spatial location \( x \)
- Overfitting - interpolates data
- Example of a kernel method, more later
**Example - Gaussian Basis Functions**

- Could define $\phi_j(x) = \exp\left\{-\frac{(x-x_j)^2}{2s^2}\right\}$
  - Gaussian around each training data point $x_j$, $N$ of them
- Could use for modelling temperature or resource availability at spatial location $x$
- Overfitting - interpolates data
- Example of a kernel method, more later
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Loss Functions for Regression

- We want to find the “best” set of coefficients $w$
- Recall, one way to define “best” was minimizing squared error:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{ y(x_n, w) - t_n \}^2$$

- We will now look at another way, based on maximum likelihood
Gaussian Noise Model for Regression

- We are provided with a training set \( \{(x_i, t_i)\} \)
- Let’s assume \( t \) arises from a deterministic function plus Gaussian distributed (with precision \( \beta \)) noise:

\[
t = y(x, w) + \epsilon
\]

- The probability of observing a target value \( t \) is then:

\[
p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})
\]

- Notation: \( \mathcal{N}(x|\mu, \sigma^2) \); \( x \) drawn from Gaussian with mean \( \mu \), variance \( \sigma^2 \)
Gaussian Noise Model for Regression

- We are provided with a training set \( \{(x_i, t_i)\} \)
- Let's assume \( t \) arises from a deterministic function plus Gaussian distributed (with precision \( \beta \)) noise:

\[
t = y(x, w) + \epsilon
\]

- The probability of observing a target value \( t \) is then:

\[
p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})
\]

- Notation: \( \mathcal{N}(x|\mu, \sigma^2); x \) drawn from Gaussian with mean \( \mu \), variance \( \sigma^2 \)
Maximum Likelihood for Regression

- The likelihood of data $t = \{t_i\}$ using this Gaussian noise model is:

$$p(t|w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})$$

- The log-likelihood is:

$$\ln p(t|w, \beta) = \ln \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left(-\frac{\beta}{2} (t_n - w^T \phi(x_n))^2\right)$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$$

- Sum of squared errors is maximum likelihood under a Gaussian noise model
Maximum Likelihood for Regression

- The likelihood of data $t = \{ t_i \}$ using this Gaussian noise model is:

$$ p(t|w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1}) $$

- The log-likelihood is:

$$ \ln p(t|w, \beta) = \ln \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left(-\frac{\beta}{2}(t_n - w^T \phi(x_n))^2\right) $$

$$ = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 $$

- Sum of squared errors is maximum likelihood under a Gaussian noise model
Maximum Likelihood for Regression

- The likelihood of data $t = \{t_i\}$ using this Gaussian noise model is:

$$p(t|w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})$$

- The log-likelihood is:

$$\ln p(t|w, \beta) = \ln \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left(-\frac{\beta}{2}(t_n - w^T \phi(x_n))^2\right)$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$$

- Sum of squared errors is maximum likelihood under a Gaussian noise model
Maximum Likelihood for Regression

- The likelihood of data $t = \{t_i\}$ using this Gaussian noise model is:

$$p(t|w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})$$

- The log-likelihood is:

$$\ln p(t|w, \beta) = \ln \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left(-\frac{\beta}{2} (t_n - w^T \phi(x_n))^2\right)$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$$

- Sum of squared errors is maximum likelihood under a Gaussian noise model
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Finding Optimal Weights

• How do we maximize likelihood wrt \( w \) (or minimize squared error)?

• Take gradient of log-likelihood wrt \( w \):

\[
\frac{\partial}{\partial w_i} \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi_i(x_n)
\]

• In vector form:

\[
\nabla \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi(x_n)^T
\]
Finding Optimal Weights

• How do we maximize likelihood wrt $w$ (or minimize squared error)?

• Take gradient of log-likelihood wrt $w$:

$$
\frac{\partial}{\partial w_i} \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi_i(x_n)
$$

• In vector form:

$$
\nabla \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi(x_n)^T
$$
Finding Optimal Weights

- How do we maximize likelihood wrt \( w \) (or minimize squared error)?
- Take gradient of log-likelihood wrt \( w \):

\[
\frac{\partial}{\partial w_i} \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi_i(x_n)
\]

- In vector form:

\[
\nabla \ln p(t|w, \beta) = \beta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi(x_n)^T
\]
Finding Optimal Weights

- Set gradient to 0:

\[
\mathbf{0}^T = \sum_{n=1}^{N} t_n \phi(x_n)^T - \mathbf{w}^T \left( \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \right)
\]

- Maximum likelihood estimate for \( \mathbf{w} \):

\[
\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}
\]

\[
\Phi = \begin{pmatrix}
\phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_{M-1}(x_1) \\
\phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_{M-1}(x_2) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_0(x_N) & \phi_1(x_N) & \ldots & \phi_{M-1}(x_N)
\end{pmatrix}
\]

- \( \Phi^\dagger = (\Phi^T \Phi)^{-1} \Phi^T \) known as the pseudo-inverse

(numpy.linalg.pinv in python)
Finding Optimal Weights

• Set gradient to 0:

\[ 0^T = \sum_{n=1}^{N} t_n \phi(x_n)^T - w^T \left( \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \right) \]

• Maximum likelihood estimate for \( w \):

\[ w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t \]

\[ \Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \ldots & \phi_{M-1}(x_N) \end{pmatrix} \]

• \( \Phi^\dagger = (\Phi^T \Phi)^{-1} \Phi^T \) known as the pseudo-inverse
  (numpy.linalg.pinv in python)
Geometry of Least Squares

- \( t = (t_1, \ldots, t_N) \) is the target value vector
- \( S \) is space spanned by \( \varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)) \)
- Solution \( y \) lies in \( S \)
- Least squares solution is orthogonal projection of \( t \) onto \( S \)
- Can verify this by looking at \( y = \Phi w_{ML} = \Phi \Phi^\dagger t = Pt \)
  - \( P^2 = P, \ P = P^T \)
Geometry of Least Squares

- \( t = (t_1, \ldots, t_N) \) is the target value vector
- \( S \) is space spanned by \( \varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)) \)
- Solution \( y \) lies in \( S \)
- Least squares solution is orthogonal projection of \( t \) onto \( S \)
- Can verify this by looking at \( y = \Phi w_{ML} = \Phi \Phi^\dagger t = Pt \)
  - \( P^2 = P, \ P = P^T \)
Geometry of Least Squares

- \( t = (t_1, \ldots, t_N) \) is the target value vector
- \( S \) is space spanned by \( \varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)) \)
- Solution \( y \) lies in \( S \)
- Least squares solution is orthogonal projection of \( t \) onto \( S \)
- Can verify this by looking at \( y = \Phi w_{ML} = \Phi \Phi^\dagger t = Pt \)
  - \( P^2 = P, \quad P = P^T \)
• \( t = (t_1, \ldots, t_N) \) is the target value vector
• \( S \) is space spanned by \( \varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)) \)
• Solution \( y \) lies in \( S \)
• Least squares solution is orthogonal projection of \( t \) onto \( S \)
• Can verify this by looking at \( y = \Phi w_{ML} = \Phi \Phi^\dagger t = Pt \)
  • \( P^2 = P \), \( P = P^T \)
Geometry of Least Squares

- \( t = (t_1, \ldots, t_N) \) is the target value vector
- \( S \) is space spanned by \( \varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)) \)
- Solution \( y \) lies in \( S \)
- Least squares solution is orthogonal projection of \( t \) onto \( S \)
- Can verify this by looking at \( y = \Phi w_{ML} = \Phi \Phi^\dagger t = Pt \)
  - \( P^2 = P, P = P^T \)
Sequential Learning

- In practice $N$ might be huge, or data might arrive online
- Can use a gradient descent method:
  - Start with initial guess for $w$
  - Update by taking a step in gradient direction $\nabla E$ of error function
- Modify to use stochastic / sequential gradient descent:
  - If error function $E = \sum_n E_n$ (e.g. least squares)
  - Update by taking a step in gradient direction $\nabla E_n$ for one example
  - Details about step size are important – decrease step size at the end
Sequential Learning

- In practice, \( N \) might be huge, or data might arrive online.
- Can use a gradient descent method:
  - Start with initial guess for \( w \)
  - Update by taking a step in gradient direction \( \nabla E \) of error function
- Modify to use stochastic / sequential gradient descent:
  - If error function \( E = \sum_n E_n \) (e.g. least squares)
  - Update by taking a step in gradient direction \( \nabla E_n \) for one example
  - Details about step size are important – decrease step size at the end
Sequential Learning

- In practice $N$ might be huge, or data might arrive online.
- Can use a gradient descent method:
  - Start with initial guess for $w$
  - Update by taking a step in gradient direction $\nabla E$ of error function
- Modify to use stochastic / sequential gradient descent:
  - If error function $E = \sum_n E_n$ (e.g. least squares)
  - Update by taking a step in gradient direction $\nabla E_n$ for one example
- Details about step size are important – decrease step size at the end.
Sequential Learning

- In practice $N$ might be huge, or data might arrive online.
- Can use a gradient descent method:
  - Start with initial guess for $w$
  - Update by taking a step in gradient direction $\nabla E$ of error function
- Modify to use stochastic / sequential gradient descent:
  - If error function $E = \sum_n E_n$ (e.g. least squares)
  - Update by taking a step in gradient direction $\nabla E_n$ for one example
  - Details about step size are important – decrease step size at the end
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Last week we discussed regularization as a technique to avoid over-fitting:

\[
\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \left( y(x_n, w) - t_n \right)^2 + \frac{\lambda}{2} \| w \|^2
\]

Next on the menu:
- Other regularizers
- Bayesian learning and quadratic regularizer
Other Regularizers

- Can use different norms for regularizer:

\[
\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \sum_{j=1}^{M} |w_j|^q
\]

- e.g. \(q = 2\) – ridge regression
- e.g. \(q = 1\) – lasso
- math is easiest with ridge regression
Optimization with a Quadratic Regularizer

- With $q = 2$, total error still a nice quadratic:

$$
\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} w^T w
$$

- Calculus ...

$$
w = \left(\lambda I + \Phi^T \Phi\right)^{-1} \Phi^T t
$$

- Similar to unregularized least squares
- Advantage $(\lambda I + \Phi^T \Phi)$ is well conditioned so inversion is stable
Ridge Regression vs. Lasso

- Ridge regression aka parameter shrinkage
  - Weights $w$ shrink back towards origin

\[ w_1 \quad w_2 \quad w^* \]

\[ w_1 \quad w_2 \]

Intuitively, once minimum achieved at large radius, minimum is on $w_1^* = 0$.
Ridge Regression vs. Lasso

- Ridge regression aka parameter shrinkage
  - Weights $w$ shrink back towards origin
- Lasso leads to sparse models
  - Components of $w$ tend to 0 with large $\lambda$ (strong regularization)
  - Intuitively, once minimum achieved at large radius, minimum is on $w_1 = 0$
Outline

Regression

Linear Basis Function Models

Loss Functions for Regression

Finding Optimal Weights

Regularization

Bayesian Linear Regression
Bayesian Linear Regression

- Last week we saw an example of a Bayesian approach
  - Coin tossing - prior on parameters
- We will now do the same for linear regression
  - Prior on parameter $w$
- There will turn out to be a connection to regularization
Bayesian Linear Regression

- Start with a prior over parameters $w$
  - **Conjugate prior** is a Gaussian:

  $$p(w) = \mathcal{N}(w|0, \alpha^{-1}I)$$

  - This simple form will make math easier; can be done for arbitrary Gaussian too

- Data likelihood, Gaussian model as before:

  $$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$$
Bayesian Linear Regression

- Start with a prior over parameters $w$
  - **Conjugate prior** is a Gaussian:
    \[
    p(w) = \mathcal{N}(w|\mathbf{0}, \alpha^{-1}\mathbf{I})
    \]
  - This simple form will make math easier; can be done for arbitrary Gaussian too
- Data likelihood, Gaussian model as before:
  \[
  p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})
  \]
Bayesian Linear Regression

• Posterior distribution on \( w \):

\[
p(w|t) \propto \left( \prod_{n=1}^{N} p(t_n|x_n, w, \beta) \right) p(w)
\]

\[
= \left[ \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp \left( -\frac{\beta}{2} (t_n - w^T \phi(x_n))^2 \right) \right] \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp \left( -\frac{\alpha}{2} w^T w \right)
\]

• Take the log:

\[-\ln p(w|t) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 + \frac{\alpha}{2} w^T w + \text{const}\]

• \( L_2 \) regularization is maximum a posteriori (MAP) with a Gaussian prior.

• \( \lambda = \alpha / \beta \)
Bayesian Linear Regression

- Posterior distribution on $w$:

$$p(w|t) \propto \left( \prod_{n=1}^{N} p(t_n|x_n, w, \beta) \right) p(w)$$

$$= \left[ \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left( -\frac{\beta}{2} (t_n - w^T \phi(x_n))^2 \right) \right] \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp\left( -\frac{\alpha}{2} w^T w \right)$$

- Take the log:

$$-\ln p(w|t) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 + \frac{\alpha}{2} w^T w + \text{const}$$

- $L_2$ regularization is maximum a posteriori (MAP) with a Gaussian prior.

- $\lambda = \alpha/\beta$
Bayesian Linear Regression

- Posterior distribution on $w$:

$$p(w|t) \propto \left( \prod_{n=1}^{N} p(t_n|x_n, w, \beta) \right) p(w)$$

$$= \left[ \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp \left( -\frac{\beta}{2} (t_n - w^T \phi(x_n))^2 \right) \right] \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp(-\frac{\alpha}{2} w^T w)$$

- Take the log:

$$-\ln p(w|t) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 + \frac{\alpha}{2} w^T w + \text{const}$$

- $L_2$ regularization is maximum a posteriori (MAP) with a Gaussian prior.

- $\lambda = \alpha/\beta$
Bayesian Linear Regression

- Posterior distribution on $w$:

$$p(w|t) \propto \left( \prod_{n=1}^{N} p(t_n|x_n, w, \beta) \right) p(w)$$

$$= \left[ \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp \left( -\frac{\beta}{2} (t_n - w^T \phi(x_n))^2 \right) \right] \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp \left( -\frac{\alpha}{2} w^T w \right)$$

- Take the log:

$$-\ln p(w|t) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 + \frac{\alpha}{2} w^T w + \text{const}$$

- $L_2$ regularization is maximum a posteriori (MAP) with a Gaussian prior.
  - $\lambda = \alpha/\beta$
Bayesian Linear Regression - Intuition

- Simple example $x, t \in \mathbb{R}$,
  \[ y(x, w) = w_0 + w_1 x \]
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Bayesian Linear Regression - Intuition

- Simple example $x, t \in \mathbb{R}$, $y(x, w) = w_0 + w_1x$
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Bayesian Linear Regression - Intuition

- Simple example $x, t \in \mathbb{R}$,
  $y(x, \mathbf{w}) = w_0 + w_1 x$
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Bayesian Linear Regression - Intuition

- Simple example $x, t \in \mathbb{R}$, $y(x, \mathbf{w}) = w_0 + w_1 x$
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Bayesian Linear Regression - Intuition

- Simple example $x, t \in \mathbb{R}$,
  $y(x, \mathbf{w}) = w_0 + w_1x$
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Bayesian Linear Regression - Intuition

- Simple example \( x, t \in \mathbb{R} \),
  \( y(x, \mathbf{w}) = w_0 + w_1 x \)
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood
Predictive Distribution

- Single estimate of \( w \) (ML or MAP) doesn’t tell whole story
- We have a distribution over \( w \), and can use it to make predictions
- Given a new value for \( x \), we can compute a distribution over \( t \):

\[
p(t|t, \alpha, \beta) = \int p(t, w|t, \alpha, \beta) dw
\]

\[
p(t|t, \alpha, \beta) = \int p(t|w, \beta) p(w|t, \alpha, \beta) \underbrace{dw}_{\text{sum}}
\]

- i.e. For each value of \( w \), let it make a prediction, multiply by its probability, sum over all \( w \)
- For arbitrary models as the distributions, this integral may not be computationally tractable
Predictive Distribution

- Single estimate of $w$ (ML or MAP) doesn’t tell whole story
- We have a distribution over $w$, and can use it to make predictions
- Given a new value for $x$, we can compute a *distribution* over $t$:

$$p(t|t, \alpha, \beta) = \int p(t, w|t, \alpha, \beta)dw$$

$$p(t|t, \alpha, \beta) = \int p(t|w, \beta) p(w|t, \alpha, \beta) dw$$

  - i.e. For each value of $w$, let it make a prediction, multiply by its probability, sum over all $w$
  - For arbitrary models as the distributions, this integral may not be computationally tractable
Predictive Distribution

- Single estimate of $w$ (ML or MAP) doesn’t tell whole story
- We have a distribution over $w$, and can use it to make predictions
- Given a new value for $x$, we can compute a distribution over $t$:

$$p(t|t, \alpha, \beta) = \int p(t, w|t, \alpha, \beta)dw$$

$$p(t|t, \alpha, \beta) = \int p(t|w, \beta) p(w|t, \alpha, \beta) dw$$

- i.e. For each value of $w$, let it make a prediction, multiply by its probability, sum over all $w$
- For arbitrary models as the distributions, this integral may not be computationally tractable
• With the Gaussians we’ve used for these distributions, the predictive distribution will also be Gaussian
  • (math on convolutions of Gaussians)
• Green line is true (unobserved) curve, blue data points, red line is mean, pink one standard deviation
  • Uncertainty small around data points
  • Pink region shrinks with more data
Bayesian Model Selection

- So what do the Bayesians say about model selection?
  - **Model selection** is choosing model $\mathcal{M}_i$ e.g. degree of polynomial, type of basis function $\phi$
  - Don’t select, just integrate
    
    $$p(t|x, \mathcal{D}) = \sum_{i=1}^{L} p(t|x, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D})$$

  - Average together the results of all models
  - Could choose most likely model a posteriori $p(\mathcal{M}_i|\mathcal{D})$
    - More efficient, approximation
Bayesian Model Selection

- So what do the Bayesians say about model selection?
  - **Model selection** is choosing model $\mathcal{M}_i$ e.g. degree of polynomial, type of basis function $\phi$
- Don’t select, just integrate

$$p(t|x, \mathcal{D}) = \sum_{i=1}^{L} \left( p(t|x, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D}) \right)$$

- Average together the results of all models
- Could choose most likely model a posteriori $p(\mathcal{M}_i|\mathcal{D})$
  - More efficient, approximation
Bayesian Model Selection

- So what do the Bayesians say about model selection?
  - **Model selection** is choosing model $\mathcal{M}_i$ e.g. degree of polynomial, type of basis function $\phi$

- Don’t select, just integrate

$$p(t|x, \mathcal{D}) = \sum_{i=1}^{L} p(t|x, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D})$$

- Average together the results of all models
- Could choose most likely model a posteriori $p(\mathcal{M}_i|\mathcal{D})$
  - More efficient, approximation
Bayesian Model Selection

• So what do the Bayesians say about model selection?
  • **Model selection** is choosing model $\mathcal{M}_i$ e.g. degree of polynomial, type of basis function $\phi$

• Don’t select, just integrate

$$p(t|x, \mathcal{D}) = \sum_{i=1}^{L} \underbrace{p(t|x, \mathcal{M}_i, \mathcal{D})}_{\text{predictive dist.}} p(\mathcal{M}_i|\mathcal{D})$$

• Average together the results of all models
• Could choose most likely model a posteriori $p(\mathcal{M}_i|\mathcal{D})$
  • More efficient, approximation
Bayesian Model Selection

- How do we compute the posterior over models?

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

- Another likelihood + prior combination

- Likelihood:

\[ p(\mathcal{D}|M_i) = \int p(\mathcal{D}|w, M_i)p(w|M_i)dw \]
Bayesian Model Selection

- How do we compute the posterior over models?

\[ p(M_i|D) \propto p(D|M_i)p(M_i) \]

- Another likelihood + prior combination

- Likelihood:

\[ p(D|M_i) = \int p(D|w, M_i)p(w|M_i)dw \]
Conclusion

• Readings: Ch. 3.1, 3.1.1-3.1.4, 3.3.1-3.3.2, 3.4
• Linear Models for Regression
  • Linear combination of (non-linear) basis functions
• Fitting parameters of regression model
  • Least squares
  • Maximum likelihood (can be = least squares)
• Controlling over-fitting
  • Regularization
  • Bayesian, use prior (can be = regularization)
• Model selection
  • Cross-validation (use held-out data)
  • Bayesian (use model evidence, likelihood)