Review of last lecture

1. Linear regression

2. Nonlinear basis functions

3. Basic ideas of overcome overfitting
Estimating model parameters

Design matrix and target vector

\[
X = \begin{pmatrix}
    x_1^T \\
    x_2^T \\
    \vdots \\
    x_N^T
\end{pmatrix} \in \mathbb{R}^{N \times D},
\hat{X} = (1 \quad X) \in \mathbb{R}^{N \times (D+1)},
\ y = \begin{pmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_N
\end{pmatrix}
\]

Residual sum squares in matrix form

\[
RSS(\bar{w}) = \left\{ \bar{w}^T \hat{X}^T \hat{X} \bar{w} - 2 \left( \hat{X}^T y \right)^T \bar{w} \right\} + \text{const}
\]
Optimal solution

Normal equation
Take derivative with respect to $\tilde{w}$

$$\frac{\partial RSS(\tilde{w})}{\partial \tilde{w}} \propto \tilde{X}^T \tilde{X} \tilde{w} - \tilde{X}^T y = 0$$

This leads to the least-mean-square (LMS) solution

$$\tilde{w}^{LMS} = \left( \tilde{X}^T \tilde{X} \right)^{-1} \tilde{X}^T y$$
Practical concerns

Bottleneck of computing the LMS solution

$$w = \left( \tilde{X}^T \tilde{X} \right)^{-1} \tilde{X} y$$

is to invert the matrix $\tilde{X}^T \tilde{X} \in \mathbb{R}^{(D+1) \times (D+1)}$

Scalable methods

- Batch gradient descent
- Stochastic gradient descent
Stochastic gradient descent

**Widrow-Hoff rule**: update parameters using one example at a time

- Initialize $\tilde{w}$ to $\tilde{w}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $x_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

$$g_t = (\tilde{x}_t^T \tilde{w}^{(t)} - y_t)\tilde{x}_t$$

- Update the parameters
  $$\tilde{w}^{(t+1)} = \tilde{w}^{(t)} - \eta g_t$$
- $t \leftarrow t + 1$

*Scalable to large problems and often very effective*
Regularized least square/Ridge regression

For $\tilde{X}^T \tilde{X}$ that is not invertible

$$\tilde{w} = \left( \tilde{X}^T \tilde{X} + \lambda I \right)^{-1} \tilde{X}^T y$$

This is equivalent to adding an extra term to $RSS(\tilde{w})$

\[
\begin{align*}
\frac{1}{2} \left\{ \tilde{w}^T \tilde{X}^T \tilde{X} \tilde{w} - 2 \left( \tilde{X}^T y \right)^T \tilde{w} \right\} + \frac{1}{2} \lambda \| \tilde{w} \|_2^2
\end{align*}
\]

regularization
Things you should know about linear regression

- Linear regression is the linear combination of features. 
  \[ f(x) = w_0 + \sum_d w_d x_d = w_0 + w^T x \]

- If we minimize residual sum squares as our learning objective, we get a closed-form solution of parameters.

- You can either invert a matrix to obtain the parameters, or use gradient-based methods (batch or stochastic gradient descent)

- Probabilistic interpretation: maximum likelihood if assuming residual is Gaussian distributed

- Ridge regression: regularizing residual sum squares
Outline

1. Review of last lecture
2. Nonlinear basis functions
3. Basic ideas of overcome overfitting
What if data is not linearly separable or fits to a line

Example of nonlinear classification
What if data is not linearly separable or fits to a line

Example of nonlinear classification

Example of nonlinear regression
Nonlinear basis for classification

Transform the input/feature

$$\phi(x) : x \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$
Nonlinear basis for classification

Transform the input/feature

\[ \phi(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2 \]

Transformed training data: linearly separable!
Another example

How to transform the input/feature?

\[ \phi(x) : x \in \mathbb{R}^2 \rightarrow z = \begin{bmatrix} x_1^2 & x_1 & x_2 \end{bmatrix} \in \mathbb{R}^3 \]

Transformed training data: linearly separable

Intuition: \( w = [1 \ 0 \ 1]^T \) then, \( w^T z = \|x\|^2 \), i.e., the distance to the origin!

Drs. Sha & Liu \({\{feisha, yaniu.cs\}@usc.edu}\)
CSCI567 Machine Learning (Fall 2014)
October 2, 2014 12 / 24
How to transform the input/feature?

\[ \phi(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^2 \rightarrow z = \begin{bmatrix} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{bmatrix} \in \mathbb{R}^3 \]
Nonlinear basis functions

Another example

How to transform the input/feature?

\[ \phi(x) : x \in \mathbb{R}^2 \rightarrow z = \begin{bmatrix} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{bmatrix} \in \mathbb{R}^3 \]

Transformed training data: linearly separable

Intuition:
\[ \mathbf{w} = [1 \ 0 \ 1]^T \text{ then, } \mathbf{w}^T z = \|x\|_2^2, \text{ i.e., the distance to the origin!} \]
General nonlinear basis functions

We can use a nonlinear mapping

\[
\phi(x) : x \in \mathbb{R}^D \rightarrow z \in \mathbb{R}^M
\]

where \( M \) is the dimensionality of the new feature/input \( z \) (or \( \phi(x) \)). Note that \( M \) could be either greater than \( D \) or less than or the same.
General nonlinear basis functions

We can use a nonlinear mapping

$$\phi(x) : x \in \mathbb{R}^D \rightarrow z \in \mathbb{R}^M$$

where $M$ is the dimensionality of the new feature/input $z$ (or $\phi(x)$). Note that $M$ could be either greater than $D$ or less than or the same.

With the new features, we can apply our learning techniques

- linear methods: prediction is based on $w^T \phi(x)$
- other methods: nearest neighbors, decision trees, etc

to minimize our errors on the transformed training data
Regression with nonlinear basis

**Residual sum squares**

$$\sum_n \left[ w^T \phi(x_n) - y_n \right]^2$$

where $w \in \mathbb{R}^M$, the same dimensionality as the transformed features $\phi(x)$. 
Nonlinear basis functions

Regression with nonlinear basis

Residual sum squares

\[
\sum_n [w^T \phi(x_n) - y_n]^2
\]

where \( w \in \mathbb{R}^M \), the same dimensionality as the transformed features \( \phi(x) \).

The LMS solution can be formulated with the new design matrix

\[
\Phi = \begin{pmatrix}
\phi(x_1)^T \\
\phi(x_2)^T \\
\vdots \\
\phi(x_N)^T
\end{pmatrix} \in \mathbb{R}^{N \times M},
\quad
w^{\text{LMS}} = \left( \Phi^T \Phi \right)^{-1} \Phi^T y
\]
Example with regression

**Polynomial basis functions**

\[
\phi(x) = \begin{bmatrix}
1 \\
x \\
x^2 \\
\vdots \\
x^M
\end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m
\]
Nonlinear basis functions

Example with regression

Polynomial basis functions

\[ \phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m \]

Fitting samples from a sine function: **underrfitting** as \( f(x) \) is too simple.
Adding more high-order basis

M=3

M=9: overfitting

Being too adaptive leads to better results on the training data, but not so great on data that has not been seen!
Adding more high-order basis

\[ M = 3 \]

\[ M = 9: \text{overfitting} \]

Being too adaptive leads better results on the training data, but not so great on data that has not been seen!
Overfiting

Parameters for higher-order polynomials are very large

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 3$</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>-1.27</td>
<td>7.99</td>
<td></td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3$</td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
</tr>
<tr>
<td>$w_4$</td>
<td></td>
<td></td>
<td>-231639.30</td>
<td></td>
</tr>
<tr>
<td>$w_5$</td>
<td></td>
<td></td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$w_6$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$w_7$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$w_8$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$w_9$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>
Overfitting can be quite disastrous

Fitting the housing price data with $M = 3$

Note that the price would go to zero (or negative) if you buy bigger ones! This is called poor generalization/overfitting.
Detecting overfitting

Plot model complexity versus objective function

As model becomes more complex, performance on training keeps improving while on test data improve first and deteriorate later.

- Horizontal axis: *measure of model complexity*
  - In this example, we use the maximum order of the polynomial basis functions.
Detecting overfitting

Plot model complexity versus objective function

As model becomes more complex, performance on training keeps improving while on test data improve first and deteriorate later.

- Horizontal axis: *measure of model complexity*
  In this example, we use the maximum order of the polynomial basis functions.

- Vertical axis:
  1. For regression, the vertical axis would be RSS or RMS (squared root of RSS)
  2. For classification, the vertical axis would be classification error rate or cross-entropy error function
Basic ideas of overcome overfitting

Outline

1. Review of last lecture
2. Nonlinear basis functions
3. Basic ideas of overcome overfitting
   - Use more training data
   - Regularization methods
Use more training data to prevent over fitting

The more, the merrier
Use more training data to prevent over fitting

The more, the merrier

![Graph showing the effect of more training data](image-url)
Use more training data to prevent over fitting

The more, the merrier

What if we do not have a lot of data?
**Regularization methods**

**Intuition**: for a linear model for regression

\[ w^T x \]

what do we mean by *being simple*?
Regularity methods

**Intuition**: for a linear model for regression

\[ w^T x \]

what do we mean by *being simple*?

**Assumptions**

\[ p(w_d) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{w_d^2}{2\sigma^2}} \]

Namely, \textit{a priori}, we believe \( w_d \) is around zero, i.e., resulting in a simple model for prediction.

*Note that this line of thinking is to regard \( w \) as a random variable and we will use the observed data \( \mathcal{D} \) to update our prior belief on \( w \).*
Example: fitting data with polynomials

**Our regression model**

\[ y = \sum_{m=1}^{M} w_m x^m \]

Thus, smaller \( w_m \) will likely lead to a smaller order of polynomial, thus potentially preventing overfitting.
Setup for regularized linear regression

**Linear regression**

\[ y = w^T x + \eta \]

where \( \eta \) is a Gaussian noise, distributed according to \( N(0, \sigma_0^2) \).

**Prior distribution on the parameter**

\[
p(w) = \prod_{d=1}^{M} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{w^2_d}{2\sigma^2}}
\]

Note that all the dimensions share the same variance \( \sigma^2 \).