Today’s Class: Practical Issues with Using Linear Regression and How to Address Them
1. Review of Linear Regression

2. Gradient Descent Methods

3. Feature Scaling

4. Ridge regression

5. Non-linear Basis Functions

6. Overfitting
Review of Linear Regression
Sale price $\approx$ price_per_sqft $\times$ square_footage + fixed_expense
Our model:
Sale_price =
price_per_sqft \times \text{square_footage} + \text{fixed_expense} + \text{unexplainable_stuff}

Training data:

<table>
<thead>
<tr>
<th>sqft</th>
<th>sale_price</th>
<th>prediction</th>
<th>error</th>
<th>squared error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>810K</td>
<td>720K</td>
<td>90K</td>
<td>8100</td>
</tr>
<tr>
<td>2100</td>
<td>907K</td>
<td>800K</td>
<td>107K</td>
<td>107^2</td>
</tr>
<tr>
<td>1100</td>
<td>312K</td>
<td>350K</td>
<td>38K</td>
<td>38^2</td>
</tr>
<tr>
<td>5500</td>
<td>2,600K</td>
<td>2,600K</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>8100 + 107^2 + 38^2 + 0 + \cdots</td>
</tr>
</tbody>
</table>

Aim:
Adjust price_per_sqft and fixed_expense such that the sum of the squared error is minimized — i.e., the unexplainable_stuff is minimized.
Linear regression

Setup:

- **Input**: $x \in \mathbb{R}^D$ (covariates, predictors, features, etc)
- **Output**: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- **Model**: $f : x \rightarrow y$, with $f(x) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \mathbf{w}^\top \mathbf{x}$.
  
  - $\mathbf{w} = [w_1 \ w_2 \ \cdots \ w_D]^\top$: weights, parameters, or parameter vector
  
  - $w_0$ is called bias.

  - Sometimes, we also call $\mathbf{w} = [w_0 \ w_1 \ w_2 \ \cdots \ w_D]^\top$ parameters.

- **Training data**: $\mathcal{D} = \{(x_n, y_n), n = 1, 2, \ldots, N\}$

Minimize the Residual sum of squares:

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} [y_n - f(x_n)]^2 = \sum_{n=1}^{N} [y_n - (w_0 + \sum_{d=1}^{D} w_d x_{nd})]^2$$
A simple case: x is just one-dimensional ($D=1$)

Residual sum of squares:

$$RSS(w) = \sum_n [y_n - f(x_n)]^2 = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$
A simple case: x is just one-dimensional ($D=1$)

**Residual sum of squares:**

$$RSS(w) = \sum_{n} [y_n - f(x_n)]^2 = \sum_{n} [y_n - (w_0 + w_1 x_n)]^2$$

**Stationary points:**

Take derivative with respect to parameters and set it to zero

$$\frac{\partial RSS(w)}{\partial w_0} = 0 \Rightarrow -2 \sum_{n} [y_n - (w_0 + w_1 x_n)] = 0,$$

$$\frac{\partial RSS(w)}{\partial w_1} = 0 \Rightarrow -2 \sum_{n} [y_n - (w_0 + w_1 x_n)] x_n = 0.$$
A simple case: \( x \) is just one-dimensional (\( D=1 \))

\[
\frac{\partial \text{RSS}(w)}{\partial w_0} = 0 \Rightarrow -2 \sum_n \left[ y_n - (w_0 + w_1 x_n) \right] = 0
\]

\[
\frac{\partial \text{RSS}(w)}{\partial w_1} = 0 \Rightarrow -2 \sum_n \left[ y_n - (w_0 + w_1 x_n) \right] x_n = 0
\]

Simplify these expressions to get the “Normal Equations”:

\[
\sum y_n = N w_0 + w_1 \sum x_n
\]

\[
\sum x_n y_n = w_0 \sum x_n + w_1 \sum x_n^2
\]

Solving the system we obtain the least squares coefficient estimates:

\[
w_1 = \frac{\sum (x_n - \bar{x})(y_n - \bar{y})}{\sum (x_i - \bar{x})^2}
\]

and

\[
w_0 = \bar{y} - w_1 \bar{x}
\]

where \( \bar{x} = \frac{1}{N} \sum_n x_n \) and \( \bar{y} = \frac{1}{N} \sum_n y_n \).
Least Mean Squares when \(x\) is \(D\)-dimensional

**RSS\((w)\) in matrix form:**

\[
\text{RSS}(w) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - w^\top x_n]^2,
\]

where we have redefined some variables (by augmenting)

\[
x \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_D]^\top, \quad w \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_D]^\top
\]

**Design matrix and target vector:**

\[
X = \begin{pmatrix}
X_1^\top \\
X_2^\top \\
\vdots \\
X_N^\top
\end{pmatrix} \in \mathbb{R}^{N \times (D+1)}, \quad y = \begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix} \in \mathbb{R}^N
\]

**Compact expression:**

\[
\text{RSS}(w) = \|Xw - y\|_2^2 = \left\{ w^\top X^\top Xw - 2 (X^\top y)^\top w \right\} + \text{const}
\]
Example: \( RSS(w) \) in compact form

<table>
<thead>
<tr>
<th>sqft (1000’s)</th>
<th>bedrooms</th>
<th>bathrooms</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4</td>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Design matrix and target vector:

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

Compact expression:

\[
RSS(w) = \|Xw - y\|_2^2 = \left\{ w^TX^TXw - 2(X^Ty)^T \right\} + \text{const}
\]
Example: $RSS(w)$ in compact form

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>bedrooms</th>
<th>bathrooms</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4</td>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Design matrix and target vector:

$$X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_N^\top \end{pmatrix} = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 1.5 & 3 & 2 \\ 1 & 2.5 & 4 & 2.5 \end{bmatrix}, \quad y = \begin{bmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{bmatrix}$$

Compact expression:

$$RSS(w) = \|Xw - y\|_2^2 = \left\{ w^\top X^\top Xw - 2 (X^\top y)^\top w \right\} + \text{const}$$
Three Optimization Methods

Want to Minimize

\[ RSS(w) = \|Xw - y\|^2_2 = \left\{ w^\top X^\top Xw - 2 (X^\top y)^\top w \right\} + \text{const} \]

- Least-Squares Solution; taking the derivative and setting it to zero
- Batch Gradient Descent
- Stochastic Gradient Descent
Least-Squares Solution

Compact expression

\[ \text{RSS}(\mathbf{w}) = \| \mathbf{Xw} - \mathbf{y} \|^2_2 = \left\{ \mathbf{w}^\top \mathbf{X}^\top \mathbf{Xw} - 2 (\mathbf{X}^\top \mathbf{y})^\top \mathbf{w} \right\} + \text{const} \]

Gradients of Linear and Quadratic Functions

- \( \nabla_x (\mathbf{b}^\top \mathbf{x}) = \mathbf{b} \)
- \( \nabla_x (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = 2\mathbf{A} \mathbf{x} \) (symmetric \( \mathbf{A} \))

Normal equation

\[ \nabla_w \text{RSS}(\mathbf{w}) \propto \mathbf{X}^\top \mathbf{Xw} - \mathbf{X}^\top \mathbf{y} = 0 \]

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

\[ \mathbf{w}^{\text{LMS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \]
Gradient Descent Methods
Outline

Review of Linear Regression

Gradient Descent Methods

Feature Scaling

Ridge regression

Non-linear Basis Functions

Overfitting
Want to Minimize

\[ RSS(w) = \|Xw - y\|^2_2 = \left\{ w^\top X^\top Xw - 2 (X^\top y)^\top w \right\} + \text{const} \]

- Least-Squares Solution; taking the derivative and setting it to zero
- **Batch Gradient Descent**
- Stochastic Gradient Descent
Computational complexity

Bottleneck of computing the solution?

\[ w = \left( X^\top X \right)^{-1} X y \]

Matrix multiply of \( X^\top X \in \mathbb{R}^{(D+1) \times (D+1)} \)
Inverting the matrix \( X^\top X \)

How many operations do we need?

- \( O(ND^2) \) for matrix multiplication
- \( O(D^3) \) (e.g., using Gauss-Jordan elimination) or \( O(D^{2.373}) \) (recent theoretical advances) for matrix inversion
- Impractical for very large \( D \) or \( N \)
(Batch) Gradient descent

- Initialize $\mathbf{w}$ to $\mathbf{w}^{(0)}$ (e.g., randomly); set $t = 0$; choose $\eta > 0$
- Loop until convergence
  1. Compute the gradient
     \[ \nabla \text{RSS}(\mathbf{w}) = \mathbf{X}^\top \mathbf{X} \mathbf{w}^{(t)} - \mathbf{X}^\top \mathbf{y} \]
  2. Update the parameters
     \[ \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla \text{RSS}(\mathbf{w}) \]
  3. $t \leftarrow t + 1$

What is the complexity of each iteration? $O(ND)$
Why would this work?

If gradient descent converges, it will converge to the same solution as using matrix inversion.

This is because $RSS(w)$ is a convex function in its parameters $w$

Hessian of $RSS$

$$RSS(w) = w^\top X^\top Xw - 2 (X^\top y)^\top w + \text{const}$$

$$\Rightarrow \frac{\partial^2 RSS(w)}{\partial w w^\top} = 2X^\top X$$

$X^\top X$ is positive semidefinite, because for any $v$

$$v^\top X^\top X v = \|X^\top v\|_2^2 \geq 0$$
Three Optimization Methods

Want to Minimize

$$RSS(w) = \|Xw - y\|_2^2 = \left\{ w^T X^T X w - 2 (X^T y)^T w \right\} + \text{const}$$

- Least-Squares Solution; taking the derivative and setting it to zero
- Batch Gradient Descent
- **Stochastic Gradient Descent**
Stochastic gradient descent (SGD)

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

- Initialize $w$ to some $w^{(0)}$; 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
     \[
     g_t = (x_t^\top w^{(t)} - y_t)x_t
     \]
  3. Update the parameters
     \[
     w^{(t+1)} = w^{(t)} - \eta g_t
     \]
  4. $t \leftarrow t + 1$

How does the complexity per iteration compare with gradient descent?

- $O(ND)$ for gradient descent versus $O(D)$ for SGD
SGD versus Batch GD

- SGD reduces per-iteration complexity from $O(ND)$ to $O(D)$
- But it is noisier and can take longer to converge
### Example: Comparing the Three Methods

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

![Graph showing the relationship between house size and price.](image-url)
The $w_0$ and $w_1$ that minimize this are given by:

$$w^{LMS} = (X^\top X)^{-1} X^\top y$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 2.5 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1.5 \\ 1 & 2.5 \end{pmatrix}^{-1} \begin{pmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{pmatrix}$$
Example: Least Squares Solution

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

The $w_0$ and $w_1$ that minimize this are given by:

$$\mathbf{w}^{LMS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

\[
\begin{bmatrix}
  w_0 \\
  w_1
\end{bmatrix} = \begin{bmatrix}
  1 & 1 & 1 & 1 \\
  1 & 2 & 1.5 & 2.5
\end{bmatrix}^{-1} \begin{bmatrix}
  1 \\
  1 \\
  1 \\
  1
\end{bmatrix} \begin{bmatrix}
  2 \\
  3.5 \\
  3 \\
  4.5
\end{bmatrix}
\]

\[
\begin{bmatrix}
  w_0 \\
  w_1
\end{bmatrix} = \begin{bmatrix}
  0.45 \\
  1.6
\end{bmatrix}
\]

Minimum RSS is $\text{RSS}^* = 0.2236$
Example: Batch Gradient Descent

<table>
<thead>
<tr>
<th>sqft (1000’s)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

\[ \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla \text{RSS}(\mathbf{w}) = \mathbf{w}^{(t)} - \eta \left( \mathbf{X}^\top \mathbf{X} \mathbf{w}^{(t)} - \mathbf{X}^\top \mathbf{y} \right) \]

\[ \eta = 0.01 \]
Larger $\eta$ gives faster convergence

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

$$w^{(t+1)} = w^{(t)} - \eta \nabla RSS(w) = w^{(t)} - \eta \left( X^T X w^{(t)} - X^T y \right)$$

![Graph showing RSS Value vs Number of Iterations for different $\eta$ values. The graph has two curves, one for $\eta = 0.01$ and another for $\eta = 0.1$. The curve for $\eta = 0.01$ converges faster.]
But too large $\eta$ makes GD unstable

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

$$w^{(t+1)} = w^{(t)} - \eta \nabla RSS(w) = w^{(t)} - \eta \left(X^T X w^{(t)} - X^T y \right)$$

![Graph showing the effect of different learning rates on the RSS value over iterations. The graph compares the RSS values for learning rates $\eta = 0.01$, $\eta = 0.1$, and $\eta = 0.12$.](image-url)
Example: Stochastic Gradient Descent

<table>
<thead>
<tr>
<th>sqft (1000’s)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

\[
\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla RSS(\mathbf{w}) = \mathbf{w}^{(t)} - \eta \left( x_t^\top \mathbf{w}^{(t)} - y \right) x_t
\]

![Graph showing the decrease in RSS value over number of iterations for \( \eta = 0.05 \).](image)
Larger $\eta$ gives faster convergence

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

\[ w^{(t+1)} = w^{(t)} - \eta \nabla RSS(w) = w^{(t)} - \eta (x_t^\top w^{(t)} - y)x_t \]
But too large \( \eta \) makes SGD unstable

\[
\begin{array}{|c|c|}
\hline
\text{sqft (1000’s)} & \text{sale price (100k)} \\
\hline
1 & 2 \\
2 & 3.5 \\
1.5 & 3 \\
2.5 & 4.5 \\
\hline
\end{array}
\]

\[
w^{(t+1)} = w^{(t)} - \eta \nabla \text{RSS}(w) = w^{(t)} - \eta \left(x_t^\top w^{(t)} - y\right)x_t
\]
How to Choose Learning Rate $\eta$ in practice?

- Try 0.0001, 0.001, 0.01, 0.1 etc. on a validation dataset (more on this later) and choose the one that gives fastest, stable convergence.
- Reduce $\eta$ by a constant factor (eg. 10) when learning saturates so that we can reach closer to the true minimum.
- More advanced learning rate schedules such as AdaGrad, Adam, AdaDelta are used in practice.
• Batch gradient descent computes the exact gradient.
• Stochastic gradient descent approximates the gradient with a single data point; its expectation equals the true gradient.
• Mini-batch variant: set the batch size to trade-off between accuracy of estimating gradient and computational cost
• Similar ideas extend to other ML optimization problems.
Feature Scaling
Outline

Review of Linear Regression

Gradient Descent Methods

Feature Scaling

Ridge regression

Non-linear Basis Functions

Overfitting
Batch Gradient Descent: Scaled Features

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

\[ w^{(t+1)} = w^{(t)} - \eta \nabla RSS(w) = w^{(t)} - \eta \left( X^T X w^{(t)} - X^T y \right) \]

Number of Iterations

<table>
<thead>
<tr>
<th>RSS Value</th>
<th>η = 0.01</th>
<th>η = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of Iterations

0.5
1.0
1.5
RSS Value

η = 0.01
η = 0.1
Batch Gradient Descent: Without Feature Scaling

<table>
<thead>
<tr>
<th>sqft</th>
<th>sale price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>200,000</td>
</tr>
<tr>
<td>2000</td>
<td>350,000</td>
</tr>
<tr>
<td>1500</td>
<td>300,000</td>
</tr>
<tr>
<td>2500</td>
<td>450,000</td>
</tr>
</tbody>
</table>

- Least-squares solution is \((w_0^*, w_1^*) = (45000, 160)\)
- \(\nabla RSS(\mathbf{w})\) becomes HUGE, causing instability
- We need a tiny \(\eta\) to compensate, but this leads to slow convergence
Batch Gradient Descent: Without Feature Scaling

<table>
<thead>
<tr>
<th>sqft</th>
<th>sale price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>200,000</td>
</tr>
<tr>
<td>2000</td>
<td>350,000</td>
</tr>
<tr>
<td>1500</td>
<td>300,000</td>
</tr>
<tr>
<td>2500</td>
<td>450,000</td>
</tr>
</tbody>
</table>

- Least-squares solution is \((w_0^*, w_1^*) = (45000, 160)\)
- \(\nabla RSS(w)\) becomes HUGE, causing instability
- We need a tiny \(\eta\) to compensate, but this leads to slow convergence

![Graph showing RSS value over iterations for different \(\eta\) values]
How to Scale Features?

Goal: Make sure that feature values are $O(1)$:

- Divide feature $x_d$ by its largest possible value in the dataset $x_d^{(1)}, \ldots x_d^{(N)}$
- OR, Replace $x_d$ by $(x_d - \mu)/(\text{max value} - \text{min value})$. This will result in all scaled features $-1 \leq x_d \leq 1$

The labels $y^{(1)}, \ldots y^{(N)}$ should be similarly re-scaled
Ridge regression
Outline

- Review of Linear Regression
- Gradient Descent Methods
- Feature Scaling
- Ridge regression
- Non-linear Basis Functions
- Overfitting
What if $X^\top X$ is not invertible?

$$w^{LMS} = (X^\top X)^{-1} X^\top y$$

Why might this happen?

- **Answer 1**: $N < D$. Not enough data to estimate all parameters.
- **Answer 2**: Columns of $X$ are not linearly independent, e.g., some features are linear functions of other features. In this case, solution is not unique. Examples:
  - A feature is a re-scaled version of another, for example, having two features correspond to length in meters and feet respectively
  - Same feature is repeated twice – could happen when there are many features
  - A feature has the same value for all data points
  - Sum of two features is equal to a third feature
Example: Matrix $X^\top X$ is not invertible

<table>
<thead>
<tr>
<th>sqft (1000's)</th>
<th>bathrooms</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>2</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Design matrix and target vector:

$$X = \begin{bmatrix}
1 & 1 & 2 \\
1 & 2 & 2 \\
1 & 1.5 & 2 \\
1 & 2.5 & 2
\end{bmatrix}, \quad w = \begin{bmatrix}
w_0 \\
w_1 \\
w_2
\end{bmatrix}, \quad y = \begin{bmatrix}
2 \\
3.5 \\
3 \\
4.5
\end{bmatrix}$$

The 'bathrooms' feature is redundant, so we don't need $w_2$

$$y = w_0 + w_1 x_1 + w_2 x_2$$

$$= w_0 + w_1 x_1 + w_2 \times 2, \quad \text{since } x_2 \text{ is always 2!}$$

$$= w_{0,\text{eff}} + w_1 x_1, \quad \text{where } w_{0,\text{eff}} = (w_0 + 2w_2)$$
**Intuition:** what does a non-invertible $X^\top X$ mean? Consider the SVD of this matrix:

$$X^\top X = V \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \lambda_r & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} V^\top$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r > 0$ and $r < D$. We will have a divide by zero issue when computing $(X^\top X)^{-1}$.

**Fix the problem:** ensure all singular values are non-zero:

$$X^\top X + \lambda I = V \text{diag}(\lambda_1 + \lambda, \lambda_2 + \lambda, \cdots, \lambda) V^\top$$

where $\lambda > 0$ and $I$ is the identity matrix.
Regularized least square (ridge regression)

**Solution**

\[ w = \left( X^\top X + \lambda I \right)^{-1} X^\top y \]

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

\[
\frac{1}{2} \left\{ w^\top X^\top X w - 2 \left( X^\top y \right)^\top w \right\} + \frac{1}{2} \lambda \|w\|^2_2
\]

**Benefits**

- Numerically more stable, invertible matrix
- Force \( w \) to be small
- Prevent overfitting — more on this later
Applying this to our example

<table>
<thead>
<tr>
<th>sqft (1000’s)</th>
<th>bathrooms</th>
<th>sale price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>2</td>
<td>4.5</td>
</tr>
</tbody>
</table>

The 'bathrooms' feature is redundant, so we don’t need \( w_2 \)

\[
y = w_0 + w_1 x_1 + w_2 x_2
\]

\[
= w_0 + w_1 x_1 + w_2 \times 2,
\]

since \( x_2 \) is always 2!

\[
= w_{0, \text{eff}} + w_1 x_1,
\]

where \( w_{0, \text{eff}} = (w_0 + 2w_2) \)

\[
= 0.45 + 1.6x_1
\]

Should get this
Applying this to our example

The 'bathrooms' feature is redundant, so we don’t need \( w_2 \)

\[
y = w_0 + w_1 x_1 + w_2 x_2
\]

\[
= w_0 + w_1 x_1 + w_2 \times 2, \quad \text{since } x_2 \text{ is always 2!}
\]

\[
= w_{0,\text{eff}} + w_1 x_1, \quad \text{where } w_{0,\text{eff}} = (w_0 + 2w_2)
\]

\[
= 0.45 + 1.6x_1 \quad \text{Should get this}
\]

Compute the solution for \( \lambda = 0.5 \)

\[
\begin{bmatrix}
w_0 \\
w_1 \\
w_2
\end{bmatrix}
= \left( X^\top X + \lambda I \right)^{-1} X^\top y
\]

\[
\begin{bmatrix}
w_0 \\
w_1 \\
w_2
\end{bmatrix}
= \begin{bmatrix}
0.208 \\
1.247 \\
0.4166
\end{bmatrix}
\]
How does $\lambda$ affect the solution?

\[
\begin{bmatrix}
  w_0 \\
  w_1 \\
  w_2
\end{bmatrix} = \left( X^\top X + \lambda I \right)^{-1} X^\top y
\]

Let us plot $w'_0 = w_0 + 2w_2$ and $w_1$ for different $\lambda \in [0.01, 20]$

Setting small $\lambda$ gives almost the least-squares solution, but it can cause numerical instability in the inversion.
How to choose $\lambda$?

$\lambda$ is referred as *hyperparameter*

- Associated with the estimation method, not the dataset
- In contrast $w$ is the parameter vector
- Use validation set or cross-validation to find good choice of $\lambda$ (more on this in the next lecture)
Add a term to the objective function.

- Choose the parameters to not just minimize risk, but avoid being too large.

$$\frac{1}{2} \left\{ w^\top X^\top X w - 2 \left( X^\top y \right)^\top w \right\} + \frac{1}{2} \lambda \| w \|_2^2$$

Probabilistic interpretation: Place a prior on our weights

- Interpret $w$ as a random variable
- Assume that each $w_d$ is centered around zero
- Use observed data $D$ to update our prior belief on $w$

Gaussian priors lead to ridge regression.
Review: Probabilistic interpretation of Linear Regression

Linear Regression model: \( Y = \mathbf{w}^\top \mathbf{X} + \eta \)

\( \eta \sim N(0, \sigma_0^2) \) is a Gaussian random variable and \( Y \sim N(\mathbf{w}^\top \mathbf{X}, \sigma_0^2) \)

**Frequentist interpretation:** We assume that \( \mathbf{w} \) is fixed.

- The likelihood function maps parameters to probabilities

\[
L : \mathbf{w}, \sigma_0^2 \mapsto p(D|\mathbf{w}, \sigma_0^2) = p(y|\mathbf{X}, \mathbf{w}, \sigma_0^2) = \prod_n p(y_n|x_n, \mathbf{w}, \sigma_0^2)
\]

- Maximizing the likelihood with respect to \( \mathbf{w} \) minimizes the RSS and yields the LMS solution:

\[
\mathbf{w}^{\text{LMS}} = \mathbf{w}^{\text{ML}} = \arg \max_{\mathbf{w}} L(\mathbf{w}, \sigma_0^2)
\]
Probabilistic interpretation of Ridge Regression

Ridge Regression model: \( Y = \mathbf{w}^\top \mathbf{X} + \eta \)

- \( Y \sim N(\mathbf{w}^\top \mathbf{X}, \sigma_0^2) \) is a Gaussian random variable (as before)
- \( w_d \sim N(0, \sigma^2) \) are i.i.d. Gaussian random variables (\textit{unlike before})
- Note that all \( w_d \) share the same variance \( \sigma^2 \)

- To find \( \mathbf{w} \) given data \( \mathcal{D} \), compute the posterior distribution of \( \mathbf{w} \):

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

- Maximum a posterior (MAP) estimate:

\[
\mathbf{w}^{\text{MAP}} = \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})p(\mathbf{w})
\]
Estimating $w$

Let $x_1, \ldots, x_N$ be i.i.d. with $y \mid w, x \sim N(w^\top x, \sigma_0^2)$; $w_d \sim N(0, \sigma^2)$.

**Joint likelihood of data and parameters (given $\sigma_0$, $\sigma$):**

$$p(D, w) = p(D \mid w)p(w) = \prod_n p(y_n \mid x_n, w) \prod_d p(w_d)$$

Plugging in the Gaussian PDF, we get:

$$\log p(D, w) = \sum_n \log p(y_n \mid x_n, w) + \sum_d \log p(w_d)$$

$$= -\frac{\sum_n (w^\top x_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const}$$

**MAP estimate:** $w_{\text{MAP}} = \arg \max_w \log p(D, w)$

$$w_{\text{MAP}} = \arg \min_w \frac{\sum_n (w^\top x_n - y_n)^2}{2\sigma_0^2} + \frac{1}{2\sigma^2} \|w\|_2^2$$
Maximum a posterior (MAP) estimate

$$\mathcal{E}(w) = \sum_n (wx_n - y_n)^2 + \lambda \|w\|_2^2$$

where $\lambda > 0$ is used to denote $\sigma_0^2 / \sigma^2$. This extra term $\|w\|_2^2$ is called regularization/regularizer and controls the magnitude of $w$.

Intuitions

- If $\lambda \to +\infty$, then $\sigma_0^2 \gg \sigma^2$: the variance of noise is far greater than what our prior model can allow for $w$. In this case, our prior model on $w$ will force $w$ to be close to zero. Numerically,
  $$w^{MAP} \to 0$$

- If $\lambda \to 0$, then we trust our data more. Numerically,
  $$w^{MAP} \to w^{LMS} = \arg\min \sum_n (w^\top x_n - y_n)^2$$
Outline

1. Review of Linear Regression

2. Gradient Descent Methods

3. Feature Scaling

4. Ridge regression

5. Non-linear Basis Functions

6. Overfitting
Non-linear Basis Functions
Outline

Review of Linear Regression

Gradient Descent Methods

Feature Scaling

Ridge regression

Non-linear Basis Functions

Overfitting
Is a linear modeling assumption always a good idea?

**Figure 1:** Sale price can saturate as sq. footage increases

**Figure 2:** Temperature has cyclic variations over each year
We can use a nonlinear mapping:

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

- \( M \) is dimensionality of new features \( z \) (or \( \phi(x) \))
- \( M \) could be greater than, less than, or equal to \( D \)

We can apply existing learning methods on the transformed data:

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

Residual sum of 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: Lot of Flexibility in Designing New Features!

<table>
<thead>
<tr>
<th>(x_1), Area (1k sqft)</th>
<th>(x_1^2), Area(^2)</th>
<th>Price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3.5</td>
</tr>
<tr>
<td>1.5</td>
<td>2.25</td>
<td>3</td>
</tr>
<tr>
<td>2.5</td>
<td>6.25</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Figure 3: Add \(x_1^2\) as a feature to allow us to fit quadratic, instead of linear functions of the house area \(x_1\)
Example: Lot of Flexibility in Designing New Features!

<table>
<thead>
<tr>
<th>$x_1$, front (100ft)</th>
<th>$x_2$ depth (100ft)</th>
<th>$10x_1x_2$, Lot (1k sqft)</th>
<th>Price (100k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>2.5</td>
<td>2</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>5</td>
<td>3.5</td>
</tr>
<tr>
<td>0.8</td>
<td>1.5</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>1.0</td>
<td>1.5</td>
<td>15</td>
<td>4.5</td>
</tr>
</tbody>
</table>

**Figure 4:** Instead of having frontage and depth as two separate features, it may be better to consider the lot-area, which is equal to frontage $\times$ depth.
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:

**underfitting** since \( f(x) \) is too simple
Adding high-order terms

More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!
Overfitting
Outline

Review of Linear Regression

Gradient Descent Methods

Feature Scaling

Ridge regression

Non-linear Basis Functions

Overfitting
Overfitting

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 large $M$:

Predicted price goes to zero (and is ultimately negative) if you buy a big enough house!

This is called poor generalization/overfitting.
Detecting overfitting

Plot model complexity versus objective function:

- X axis: model complexity, e.g., $M$
- Y axis: error, e.g., RSS, RMS (square root of RSS), 0-1 loss

Compute the objective on a training and test dataset.

As a model increases in complexity:

- Training error keeps improving
- Test error may first improve but eventually will deteriorate
Dealing with overfitting

Try to use more training data

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

Intuition: Give preference to ‘simpler’ models

- How do we define a simple linear regression model — $\mathbf{w}^\top \mathbf{x}$?
- Intuitively, the weights should not be “too 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></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>