18-661 Introduction to Machine Learning

Linear Regression – II

Fall 2020

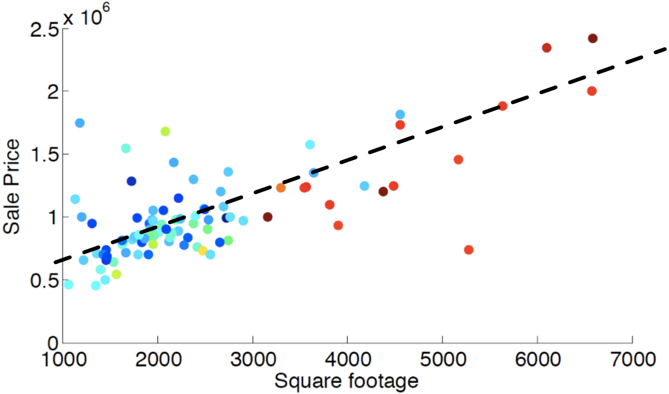
ECE – Carnegie Mellon University

Today's Class: Practical Issues with Using Linear Regression and How to Address Them

Outline

Review of Linear Regression

Example: Predicting house prices



Sale price $\approx$ price_per_sqft $\times$ square_footage $+$ fixed_expense

Minimize squared errors

Our model:

Sale_price $=$

price_per_sqft $\times$ square_footage $+$ fixed_expense $+$ unexplainable_stuff

Training data:

sqft	sale price	prediction	error	squared error
2000	810K	720K	90K	8100
2100	907K	800K	107K	107^2
1100	312K	350K	38K	38^2
5500	2,600K	2,600K	0	0
$\cdots$	$\cdots$			
Total				$8100 + 107^2 + 38^2 + 0 + \cdots$

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**: $\mathbf{x} \in \mathbb{R}^D$ (covariates, predictors, features, etc)
- **Output**: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- **Model**: $f : \mathbf{x} \to y$, with $f(\mathbf{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \mathbf{w}^\top \mathbf{x}$.
  - $\mathbf{w} = [w_1 \; w_2 \; \cdots \; w_D]^\top$: *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**: $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, 2, \ldots, N\}$

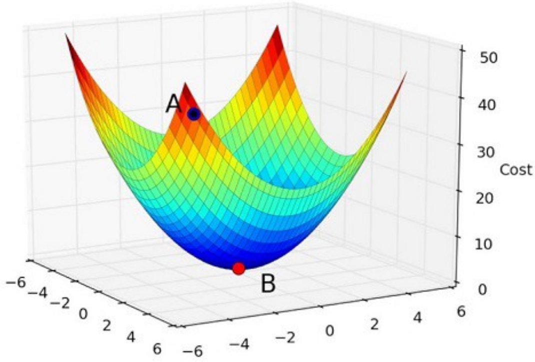
Minimize the Residual sum of squares:

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} [y_n - f(\mathbf{x}_n)]^2 = \sum_{n=1}^{N} [y_n - (w_0 + \sum_{d=1}^{D} w_d x_{nd})]^2$$

A simple case: x is just one-dimensional (D=1)

Residual sum of squares:

$$RSS(\mathbf{w}) = \sum_n [y_n - f(\mathbf{x}_n)]^2 = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$



A simple case: x is just one-dimensional (D=1)

Residual sum of squares:

$$RSS(\mathbf{w}) = \sum_n [y_n - f(\mathbf{x}_n)]^2 = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$

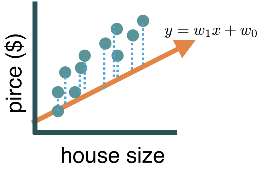


Figure 1: RSS is the sum of squares of the dotted lines

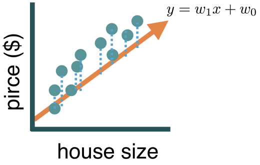


Figure 2: Adjust (w_0, w_1) to reduce RSS

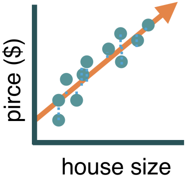


Figure 3: RSS minimized at (w_o^*, w_1^*)

A simple case: x is just one-dimensional (D=1)

Residual sum of squares:

$$RSS(\mathbf{w}) = \sum_n [y_n - f(\mathbf{x}_n)]^2 = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$

Stationary points:

Take derivative with respect to parameters and set it to zero

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

$$\frac{\partial RSS(\mathbf{w})}{\partial w_1} = 0 \Rightarrow -2 \sum_n [y_n - (w_0 + w_1 x_n)] x_n = 0.$$

A simple case: x is just one-dimensional (D=1)

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

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

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:

$$w_1 = \frac{\sum (x_n - \bar{x})(y_n - \bar{y})}{\sum (x_i - \bar{x})^2} \qquad \text{and} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

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

$RSS(\mathbf{w})$ in matrix form:

$$RSS(\mathbf{w}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \mathbf{w}^\top \mathbf{x}_n]^2,$$

where we have redefined some variables (by augmenting)

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

Design matrix and target vector:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} \in \mathbb{R}^{N \times (D+1)}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{R}^N$$

Compact expression:

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

Example: $RSS(\mathbf{w})$ in compact form

sqft (1000's)	bedrooms	bathrooms	sale price (100k)
1	2	1	2
2	2	2	3.5
1.5	3	2	3
2.5	4	2.5	4.5

Design matrix and target vector:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} \in \mathbb{R}^{N\times(D+1)}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{R}^N$$

. Compact expression:

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

Example: $RSS(\mathbf{w})$ in compact form

sqft (1000's)	bedrooms	bathrooms	sale price (100k)
1	2	1	2
2	2	2	3.5
1.5	3	2	3
2.5	4	2.5	4.5

Design matrix and target vector:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 1.5 & 3 & 2 \\ 1 & 2.5 & 4 & 2.5 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{bmatrix}$$

. Compact expression:

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

Three Optimization Methods

Want to Minimize

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

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

Least-Squares Solution

Compact expression

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

Gradients of Linear and Quadratic Functions

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

Normal equation

$$\nabla_{\mathbf{w}} RSS(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y} = 0$$

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

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

Gradient Descent Methods

Outline

Three Optimization Methods

Want to Minimize

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

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

Computational complexity

Bottleneck of computing the solution?

$$\boldsymbol{w} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

How many operations do we need?

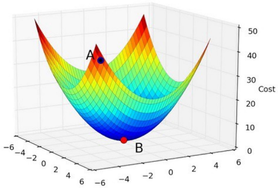
- $O(\mathrm{ND}^2)$ for matrix multiplication $\mathbf{X}^\top\mathbf{X}$
- $O(\mathrm{D}^3)$ (e.g., using Gauss-Jordan elimination) or $O(\mathrm{D}^{2.373})$ (recent theoretical advances) for matrix inversion of $\mathbf{X}^\top\mathbf{X}$
- $O(\mathrm{ND})$ for matrix multiplication $\mathbf{X}^\top\boldsymbol{y}$
- $O(\mathrm{D}^2)$ for $\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1}$ times $\boldsymbol{X}^\top \boldsymbol{y}$

$O(\mathrm{ND}^2) + O(\mathrm{D}^3)$ – Impractical for very large D or N

Alternative method: Batch Gradient Descent

(Batch) Gradient descent

- Initialize **w** to $\boldsymbol{w}^{(0)}$ (e.g., randomly);
  set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Compute the gradient
     $\nabla RSS(\boldsymbol{w}) = \mathbf{X}^\top(\mathbf{X}\boldsymbol{w}^{(t)} - \boldsymbol{y})$
  2. Update the parameters
     $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w})$
  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 $RSS(\boldsymbol{w})$ is a convex function in its parameters $\boldsymbol{w}$

Hessian of RSS

$$RSS(\boldsymbol{w}) = \boldsymbol{w}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{w} - 2\left(\mathbf{X}^\top \boldsymbol{y}\right)^\top \boldsymbol{w} + \text{const}$$
$$\Rightarrow \frac{\partial^2 RSS(\boldsymbol{w})}{\partial \boldsymbol{w} \boldsymbol{w}^\top} = 2\mathbf{X}^\top \mathbf{X}$$

$\mathbf{X}^\top \mathbf{X}$ is positive semidefinite, because for any $\boldsymbol{v}$

$$\boldsymbol{v}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{v} = \|\mathbf{X}^\top \boldsymbol{v}\|_2^2 \geq 0$$

Three Optimization Methods

Want to Minimize

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

- 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 $\boldsymbol{w}$ to some $\boldsymbol{w}^{(0)}$; set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient

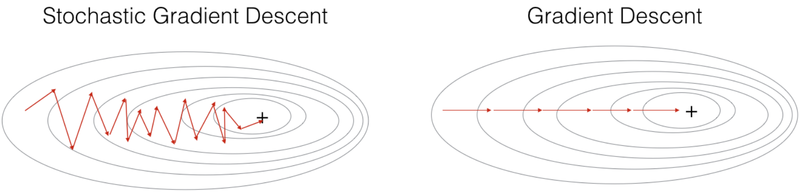
$$\boldsymbol{g}_t = (\boldsymbol{x}_t^\top \boldsymbol{w}^{(t)} - y_t)\mathbf{x}_t$$

  3. Update the parameters
     $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \boldsymbol{g}_t$
  4. $t \leftarrow t + 1$

How does the complexity per iteration compare with gradient descent?

- $O(\mathrm{ND})$ for gradient descent versus $O(\mathrm{D})$ for SGD

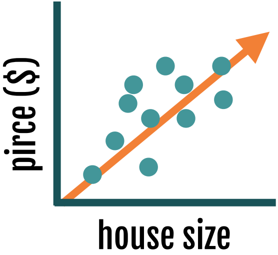
SGD versus Batch GD



- SGD reduces per-iteration complexity from $O(\mathrm{ND})$ to $O(\mathrm{D})$
- But it is noisier and can take longer to converge

Example: Comparing the Three Methods

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5



Example: Least Squares Solution

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

The w_0 and w_1 that minimize this are given by:

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

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1.5 \\ 1 & 2.5 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{bmatrix}$$

Example: Least Squares Solution

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

The w_0 and w_1 that minimize this are given by:

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

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1.5 \\ 1 & 2.5 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{bmatrix}$$

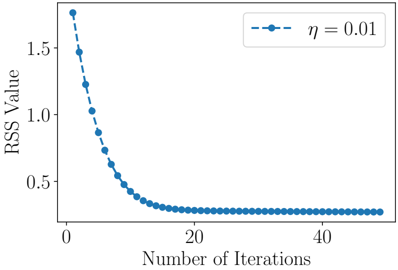
$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 1.6 \end{bmatrix}$$

Minimum RSS is $RSS^* = ||\mathbf{X}\mathbf{w}^{LMS} - \mathbf{y}||_2^2 = 0.2236$

Example: Batch Gradient Descent

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

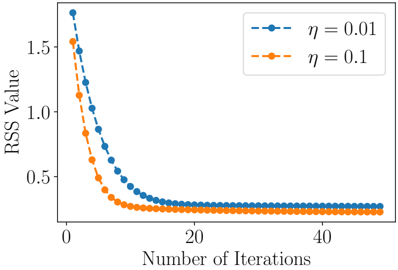
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \mathbf{X}^\top \left(\mathbf{X} \boldsymbol{w}^{(t)} - \boldsymbol{y} \right)$$



Larger η gives faster convergence

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

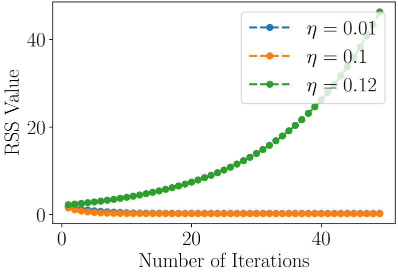
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \mathbf{X}^\top \left(\mathbf{X} \boldsymbol{w}^{(t)} - \boldsymbol{y} \right)$$



But too large η makes GD unstable

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

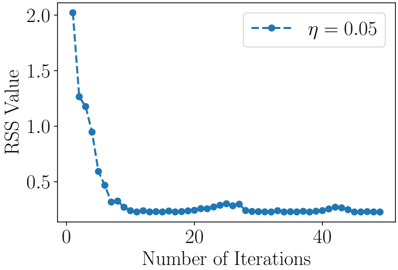
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \mathbf{X}^{\top} \left(\mathbf{X} \boldsymbol{w}^{(t)} - \boldsymbol{y} \right)$$



Example: Stochastic Gradient Descent

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

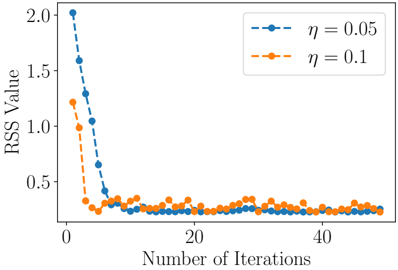
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \left(\mathbf{x}_t^\top \boldsymbol{w}^{(t)} - \boldsymbol{y} \right) \mathbf{x}_t$$



Larger η gives faster convergence

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

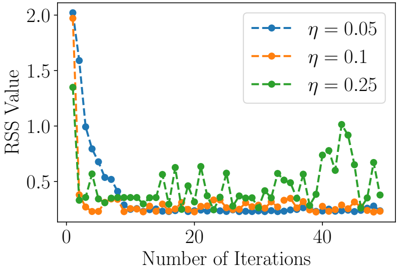
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \left(\mathbf{x}_t^\top \boldsymbol{w}^{(t)} - \boldsymbol{y} \right) \mathbf{x}_t$$



But too large η makes SGD unstable

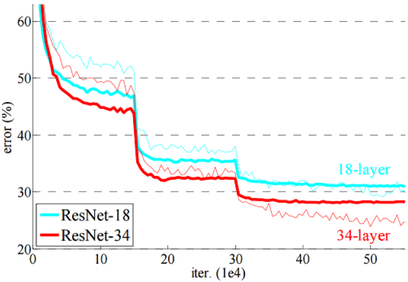
sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \left(\mathbf{x}_t^\top \boldsymbol{w}^{(t)} - \boldsymbol{y} \right) \mathbf{x}_t$$



How to Choose Learning Rate η 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 η 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.



Summary of Gradient Descent Methods

- 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.

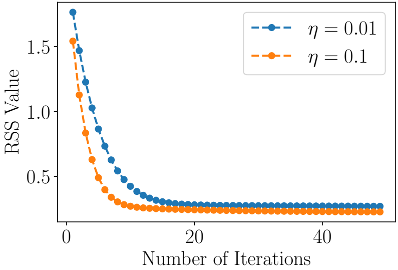
Feature Scaling

Outline

Batch Gradient Descent: Scaled Features

sqft (1000's)	sale price (100k)
1	2
2	3.5
1.5	3
2.5	4.5

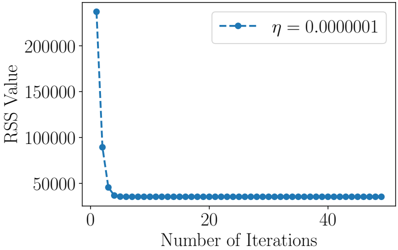
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w}) = \boldsymbol{w}^{(t)} - \eta \mathbf{X}^\top \left(\mathbf{X}\boldsymbol{w}^{(t)} - \boldsymbol{y} \right)$$



Batch Gradient Descent: Without Feature Scaling

sqft	sale price
1000	200,000
2000	350,000
1500	300,000
2500	450,000

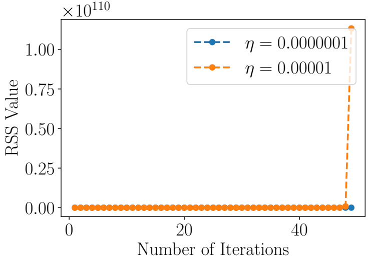
- Least-squares solution is $(w_0^*, w_1^*) = (45000, 160)$
- $\nabla RSS(\boldsymbol{w}^{(t)}) = \mathbf{X}^\top \left(\mathbf{X}\boldsymbol{w}^{(t)} - \boldsymbol{y}\right)$ becomes HUGE, causing instability
- We need a tiny η to compensate, but this leads to slow convergence



Batch Gradient Descent: Without Feature Scaling

sqft	sale price
1000	200,000
2000	350,000
1500	300,000
2500	450,000

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



How to Scale Features?

- **Min-max normalization**

$$x'_d = \frac{x_d - \min_n(x_d)}{\max_n x_d - \min_n x_d}$$

The min and max are taken over the possible values $x_d^{(1)}, \dots x_d^{(N)}$ of x_d in the dataset. This will result in all scaled features $0 \leq x_d \leq 1$

- **Mean normalization**

$$x'_d = \frac{x_d - \text{avg}(x_d)}{\max_n x_d - \min_n x_d}$$

This will result in all scaled features $-1 \leq x_d \leq 1$

Labels $y^{(1)}, \dots y^{(N)}$ should be similarly re-scaled
Several other methods: eg. dividing by standard deviation (Z-score normalization)

Ridge regression

Outline

What if $X^\top X$ is not invertible?

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

Why might this happen?

- **Answer 1:** N < D. Not enough data to estimate all parameters. $\mathbf{X}^\top \mathbf{X}$ is not full-rank
- **Answer 2:** Columns of $\boldsymbol{X}$ 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

sqft (1000's)	bathrooms	sale price (100k)
1	2	2
2	2	3.5
1.5	2	3
2.5	2	4.5

Design matrix and target vector:

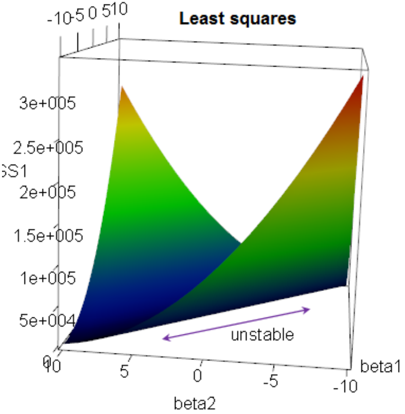
$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 1.5 & 2 \\ 1 & 2.5 & 2 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 3.5 \\ 3 \\ 4.5 \end{bmatrix}$$

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

$$\begin{aligned} y &= w_0 + w_1 x_1 + w_2 x_2 \\ &= w_0 + w_1 x_1 + w_2 \times 2, \quad \text{since } x_2 \text{ is always 2!} \\ &= w_{0,eff} + w_1 x_1, \quad \text{where } w_{0,eff} = (w_0 + 2w_2) \end{aligned}$$

What does the RSS loss function look like?

- When $\boldsymbol{X}^\top\boldsymbol{X}$ is not invertible, the RSS objective function has a ridge, that is, the minimum is a line instead of a single point



In our example, this line is $w_{0,\mathit{eff}} = (w_0 + 2w_2)$

How do you fix this issue?

sqft (1000's)	bathrooms	sale price (100k)
1	2	2
2	2	3.5
1.5	2	3
2.5	2	4.5

- Manually remove redundant features
- But this can be tedious and non-trivial, especially when a feature is a linear combination of several other features

Need a general way that doesn't require manual feature engineering
SOLUTION: Ridge Regression

Ridge regression

Intuition: what does a non-invertible $\boldsymbol{X}^\top \boldsymbol{X}$ mean?
Consider the SVD of this matrix:

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

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

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

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

where $\lambda > 0$ and $\boldsymbol{I}$ is the identity matrix.

Regularized least square (ridge regression)

Solution

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

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

$$\overbrace{\frac{1}{2}\left\{\boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w} - 2\left(\boldsymbol{X}^\top \boldsymbol{y}\right)^\top \boldsymbol{w}\right\}}^{RSS(\boldsymbol{w})} + \underbrace{\frac{1}{2}\lambda \|w\|_2^2}_{\text{regularization}}$$

Benefits

- Numerically more stable, invertible matrix
- Force **w** to be small
- Prevent overfitting — more on this later

Applying this to our example

sqft (1000's)	bathrooms	sale price (100k)
1	2	2
2	2	3.5
1.5	2	3
2.5	2	4.5

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

$$
\begin{aligned}
y &= w_0 + w_1 x_1 + w_2 x_2 \\
&= w_0 + w_1 x_1 + w_2 \times 2, && \text{since } x_2 \text{ is always 2!} \\
&= w_{0,eff} + w_1 x_1, && \text{where } w_{0,eff} = (w_0 + 2w_2) \\
&= 0.45 + 1.6 x_1 && \text{Should get this}
\end{aligned}
$$

Applying this to our example

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

$$\begin{aligned} y &= w_0 + w_1 x_1 + w_2 x_2 \\ &= w_0 + w_1 x_1 + w_2 \times 2, \quad \text{since } x_2 \text{ is always 2!} \\ &= w_{0,eff} + w_1 x_1, \quad \text{where } w_{0,eff} = (w_0 + 2w_2) \\ &= 0.45 + 1.6 x_1 \quad \text{Should get this} \end{aligned}$$

Compute the solution for $\lambda = 0.5$

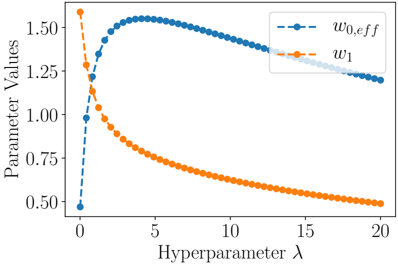
$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.208 \\ 1.247 \\ 0.4166 \end{bmatrix}$$

How does λ affect the solution?

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

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

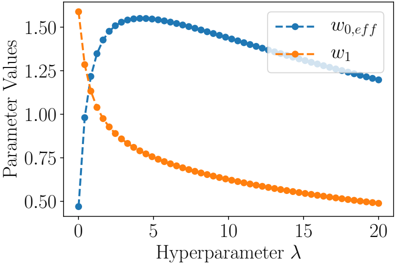


Setting small λ gives almost the least-squares solution, but it can cause numerical instability in the inversion

How to choose λ?

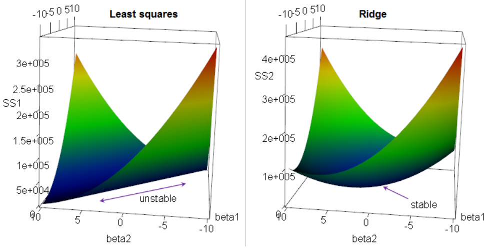
λ is referred as *hyperparameter*

- Associated with the estimation method, not the dataset
- In contrast $\boldsymbol{w}$ is the parameter vector
- Use validation set or cross-validation to find good choice of λ (more on this in the next lecture)



Why is it called Ridge Regression?

- When $\boldsymbol{X}^\top \boldsymbol{X}$ is not invertible, the RSS objective function has a ridge, that is, the minimum is a line instead of a single point
- Adding the regularizer term $\frac{1}{2}\lambda\|w\|_2^2$ yields a unique minimum, thus avoiding instability in matrix inversion



Probabilistic Interpretation of Ridge Regression

Add a term to the objective function.

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

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

Probabilistic interpretation: Place a prior on our weights

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

Gaussian priors lead to ridge regression.

Review: Probabilistic interpretation of Linear Regression

Linear Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$

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

Frequentist interpretation: We assume that $\boldsymbol{w}$ is fixed.

- The likelihood function maps parameters to probabilities

$$L : \boldsymbol{w}, \sigma_0^2 \mapsto p(\mathcal{D}|\boldsymbol{w}, \sigma_0^2) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}, \sigma_0^2) = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \sigma_0^2)$$

- Maximizing the likelihood with respect to $\boldsymbol{w}$ minimizes the RSS and yields the LMS solution:

$$\boldsymbol{w}^{\mathrm{LMS}} = \boldsymbol{w}^{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} L(\boldsymbol{w}, \sigma_0^2)$$

Probabilistic interpretation of Ridge Regression

Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$

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

- To find $\boldsymbol{w}$ given data $\mathcal{D}$, compute the posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})$$

Estimating w

Let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be i.i.d. with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$; $w_d \sim N(0, \sigma^2)$.

Joint likelihood of data and parameters (given σ_0, σ):

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Plugging in the Gaussian PDF, we get:

$$\begin{aligned}
\log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\
&= -\frac{\sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const}
\end{aligned}$$

MAP estimate: $\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log p(\mathcal{D}, \boldsymbol{w})$

$$\boldsymbol{w}^{\mathrm{MAP}} = \operatorname{argmin}_{\boldsymbol{w}} \frac{\sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} + \frac{1}{2\sigma^2} \|\boldsymbol{w}\|_2^2$$

Maximum a posterior (MAP) estimate

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the magnitude of $\boldsymbol{w}$.

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 $\boldsymbol{w}$. In this case, our prior model on $\boldsymbol{w}$ will force $\boldsymbol{w}$ to be close to zero. Numerically,

$$\boldsymbol{w}^{\text{MAP}} \to \mathbf{0}$$

- If $\lambda \to 0$, then we trust our data more. Numerically,

$$\boldsymbol{w}^{\text{MAP}} \to \boldsymbol{w}^{\text{LMS}} = \operatorname{argmin} \sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2$$

Outline

Non-linear Basis Functions

Outline

Is a linear modeling assumption always a good idea?

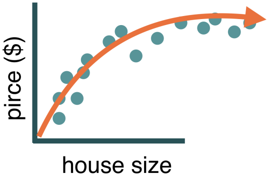


Figure 4: Sale price can saturate as sq.footage increases

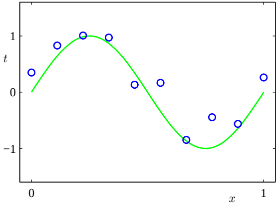


Figure 5: Temperature has cyclic variations over each year

General nonlinear basis functions

We can use a nonlinear mapping to a new feature vector:

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

- M is dimensionality of new features $\boldsymbol{z}$ (or $\phi(\boldsymbol{x})$)
- 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 $\boldsymbol{w}^\top \phi(\boldsymbol{x})$
- other methods: nearest neighbors, decision trees, etc

Regression with nonlinear basis

Residual sum of squares

$$\sum_n [\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) - y_n]^2$$

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

The LMS solution can be formulated with the new design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \phi(\boldsymbol{x}_2)^\top \\ \vdots \\ \phi(\boldsymbol{x}_N)^\top \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \boldsymbol{w}^{\mathrm{LMS}} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$$

Example: Lot of Flexibility in Designing New Features!

x_1, Area (1k sqft)	x_1^2, Area^2	Price (100k)
1	1	2
2	4	3.5
1.5	2.25	3
2.5	6.25	4.5

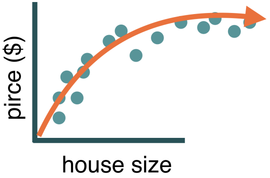


Figure 6: Add x_1^2 as a feature to allow us to fit quadratic, instead of linear functions of the house area x_1

Example: Lot of Flexibility in Designing New Features!

x_1, front (100ft)	x_2 depth (100ft)	$10x_1x_2$, Lot (1k sqft)	Price (100k)
0.5	0.5	2.5	2
0.5	1	5	3.5
0.8	1.5	12	3
1.0	1.5	15	4.5

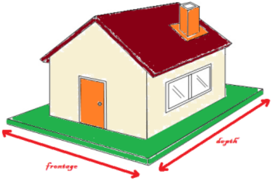


Figure 7: Instead of having frontage and depth as two separate features, it may be better to consider the lot-area, which is equal to frontage×depth

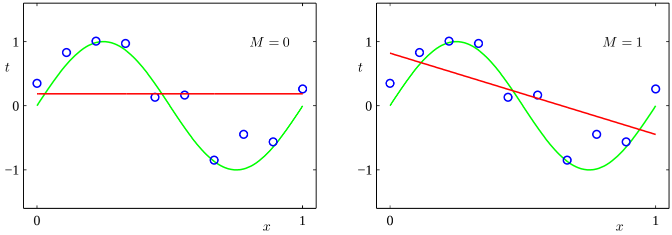
Example with regression

Polynomial basis functions

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

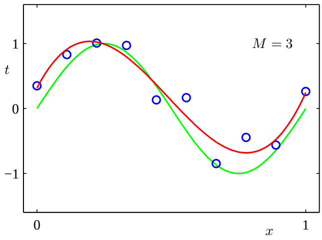
Fitting samples from a sine function:

underfitting since $f(x)$ is too simple

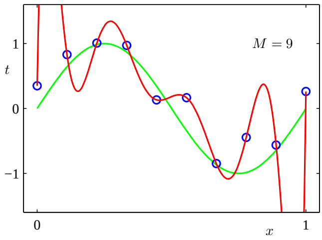


Adding high-order terms

M=3



M=9: overfitting



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

Overfitting

Outline

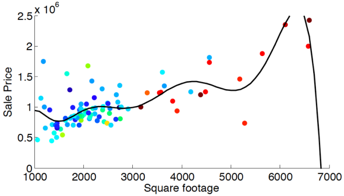
Overfitting

Parameters for higher-order polynomials are very large

	$M=0$	$M=1$	$M=3$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

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!

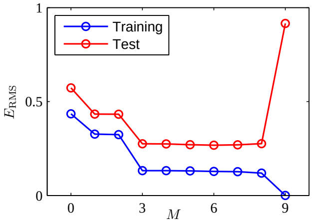
This is called poor generalization/overfitting.

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

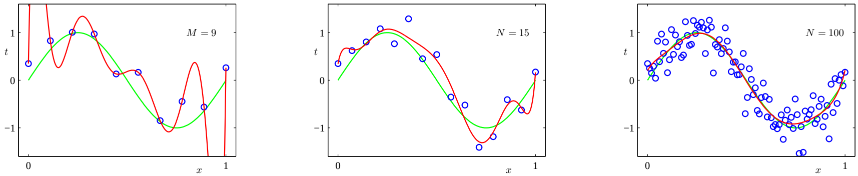
Compute the objective on a training and test dataset.



As a model increases in complexity:

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

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 — $\boldsymbol{w}^\top \boldsymbol{x}$?
- Intuitively, the weights should not be "too large"

	$M=0$	$M=1$	$M=3$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Next Class: Overfitting, Regularization and the Bias-variance trade-off