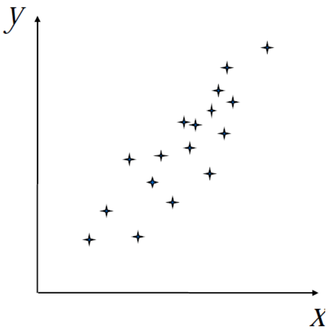
Learning Models by Fitting Parameters: Linear and Ridge Regression

Piyush Rai

CS5350/6350: Machine Learning

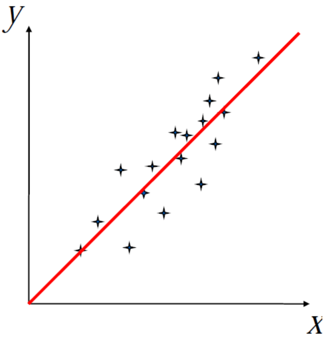
September 6, 2011

Linear Regression: One-Dimensional Case



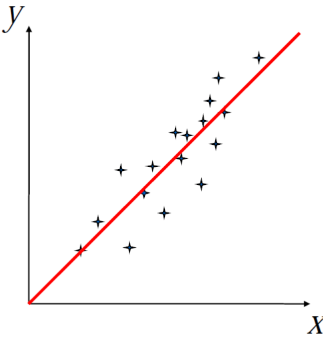
- **Given:** a set of N input-response pairs
- The inputs (x) and the responses (y) are one dimensional scalars
- **Goal:** Model the relationship between x and y

Linear Regression: One-Dimensional Case



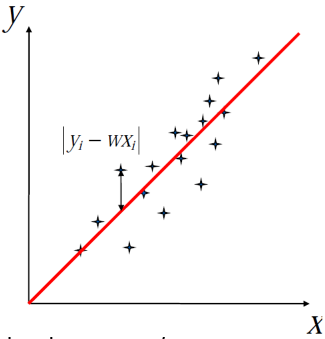
- Let's assume the relationship between x and y is linear

Linear Regression: One-Dimensional Case



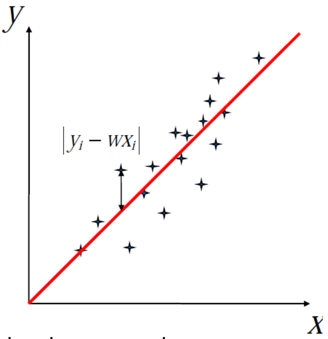
- Let's assume the relationship between x and y is linear
- Linear relationship can be defined by a straight line with *parameter* w
- Equation of the straight line: $y = wx$

Linear Regression: One-Dimensional Case



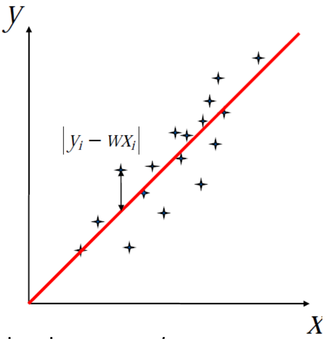
- The line may not fit the data *exactly*

Linear Regression: One-Dimensional Case



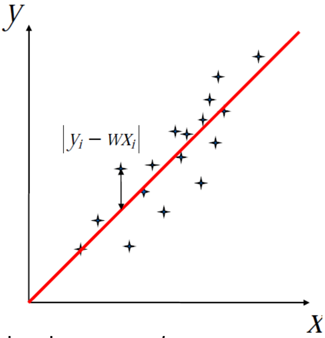
- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation

Linear Regression: One-Dimensional Case



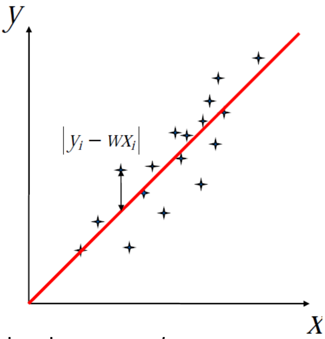
- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation
- Error for the pair (x_i, y_i) pair: $e_i = y_i - wx_i$

Linear Regression: One-Dimensional Case



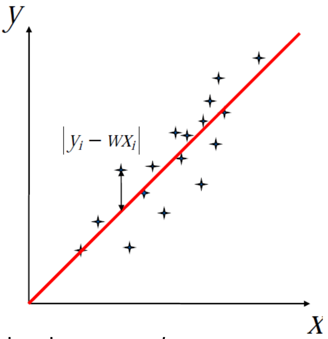
- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation
- Error for the pair (x_i, y_i) pair: $e_i = y_i - wx_i$
- The total squared error: $E = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (y_i - wx_i)^2$

Linear Regression: One-Dimensional Case



- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation
- Error for the pair (x_i, y_i) pair: $e_i = y_i - wx_i$
- The total squared error: $E = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N}(y_i - wx_i)^2$
- The best fitting line is defined by w minimizing the total error E

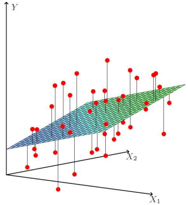
Linear Regression: One-Dimensional Case



- The line may not fit the data *exactly*
- But we can try making the line a reasonable approximation
- Error for the pair (x_i, y_i) pair: $e_i = y_i - wx_i$
- The total squared error: $E = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N}(y_i - wx_i)^2$
- The best fitting line is defined by w minimizing the total error E
- Just requires a little bit of calculus to find it (take derivative, equate to zero..)

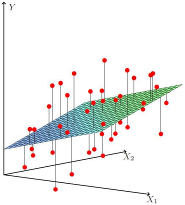
Linear Regression: In Higher Dimensions

- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



Linear Regression: In Higher Dimensions

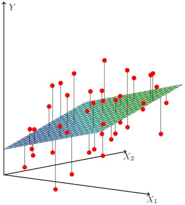
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?

Linear Regression: In Higher Dimensions

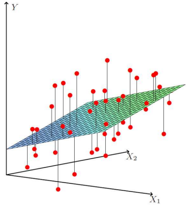
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y

Linear Regression: In Higher Dimensions

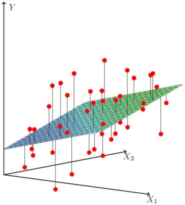
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y
  - Linear regression uses the sum-of-squared error notion of closeness

Linear Regression: In Higher Dimensions

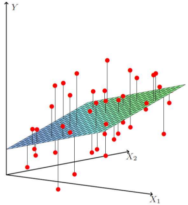
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too

Linear Regression: In Higher Dimensions

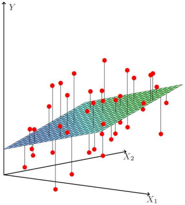
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too
  - Fitting a D-dimensional hyperplane to the data

Linear Regression: In Higher Dimensions

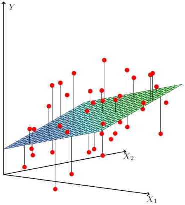
- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too
  - Fitting a D-dimensional hyperplane to the data
  - Hard to visualize in pictures though..

Linear Regression: In Higher Dimensions

- **Analogy to line fitting:** In higher dimensions, we will fit hyperplanes
- For 2-dim. inputs, linear regression fits a 2-dim. plane to the data



- Many planes are possible. Which one is the best?
- **Intuition:** Choose the one which is (on average) closest to the responses Y
  - Linear regression uses the sum-of-squared error notion of closeness
- Similar intuition carries over to higher dimensions too
  - Fitting a D-dimensional hyperplane to the data
  - Hard to visualize in pictures though..
- The hyperplane is defined by parameters $\mathbf{w}$ (a $D \times 1$ weight vector)

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Inputs $\mathbf{x}_i$: D-dimensional vectors ($\mathbb{R}^D$), responses y_i: scalars ($\mathbb{R}$)

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Inputs $\mathbf{x}_i$: D-dimensional vectors ($\mathbb{R}^D$), responses y_i: scalars ($\mathbb{R}$)
- The linear model: response is a linear function of the model parameters

$$y = f(\mathbf{x}, \mathbf{w}) = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x})$$

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Inputs $\mathbf{x}_i$: D-dimensional vectors ($\mathbb{R}^D$), responses y_i: scalars ($\mathbb{R}$)
- The linear model: response is a linear function of the model parameters

$$y = f(\mathbf{x}, \mathbf{w}) = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x})$$

- w_j's and b are the model parameters (b is an offset)
  - Parameters define the mapping from the inputs to responses

Linear Regression: In Higher Dimensions (Formally)

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Inputs $\mathbf{x}_i$: D-dimensional vectors ($\mathbb{R}^D$), responses y_i: scalars ($\mathbb{R}$)
- The linear model: response is a linear function of the model parameters

$$y = f(\mathbf{x}, \mathbf{w}) = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x})$$

- w_j's and b are the model parameters (b is an offset)
  - Parameters define the mapping from the inputs to responses
- Each ϕ_j is called a basis function
  - Allows change of representation of the input $\mathbf{x}$ (often desired)

Linear Regression: In Higher Dimensions

The linear model:

$$y = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = b + \mathbf{w}^T \phi(\mathbf{x})$$

- $\phi = [\phi_1, \ldots . \phi_M]$
- $\mathbf{w} = [w_1, \ldots, w_M]$, the **weight vector** (to learn using the training data)

Linear Regression: In Higher Dimensions

The linear model:

$$y = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = b + \mathbf{w}^T \phi(\mathbf{x})$$

- $\phi = [\phi_1, \ldots . \phi_M]$
- $\mathbf{w} = [w_1, \ldots, w_M]$, the **weight vector** (to learn using the training data)
- We consider the simplest case: $\phi(\mathbf{x}) = \mathbf{x}$
  - $\phi_j(\mathbf{x})$ is the j-th feature of the data (total D features, so $M = D$)

Linear Regression: In Higher Dimensions

The linear model:

$$y = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = b + \mathbf{w}^T \phi(\mathbf{x})$$

- $\phi = [\phi_1, \ldots . \phi_M]$
- $\mathbf{w} = [w_1, \ldots, w_M]$, the **weight vector** (to learn using the training data)
- We consider the simplest case: $\phi(\mathbf{x}) = \mathbf{x}$
  - $\phi_j(\mathbf{x})$ is the j-th feature of the data (total D features, so $M = D$)
- The linear model becomes

$$\boxed{y = b + \sum_{j=1}^{D} w_j x_j = b + \mathbf{w}^T \mathbf{x}}$$

Linear Regression: In Higher Dimensions

The linear model:

$$y = b + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = b + \mathbf{w}^T \phi(\mathbf{x})$$

- $\phi = [\phi_1, \ldots . \phi_M]$
- $\mathbf{w} = [w_1, \ldots, w_M]$, the **weight vector** (to learn using the training data)
- We consider the simplest case: $\phi(\mathbf{x}) = \mathbf{x}$
  - $\phi_j(\mathbf{x})$ is the j-th feature of the data (total D features, so $M = D$)
- The linear model becomes

$$\boxed{y = b + \sum_{j=1}^{D} w_j x_j = b + \mathbf{w}^T \mathbf{x}}$$

- **Note:** Nonlinear relationships between $\mathbf{x}$ and y can be modeled using suitably chosen ϕ_j's (more when we cover Kernel Methods)

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Switching to matrix notation, the relationship becomes: $\mathbf{Y} = \mathbf{X}\mathbf{w}$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Switching to matrix notation, the relationship becomes: $\mathbf{Y} = \mathbf{X}\mathbf{w}$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix},$$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Switching to matrix notation, the relationship becomes: $\mathbf{Y} = \mathbf{X}\mathbf{w}$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 \; \mathbf{x}_1 \\ \vdots \; \vdots \\ 1 \; \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} 1 & x_{11} \cdots x_{1D} \\ \vdots & \ddots \quad \vdots \\ 1 & x_{N1} \cdots x_{ND} \end{pmatrix},$$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Switching to matrix notation, the relationship becomes: $\mathbf{Y} = \mathbf{X}\mathbf{w}$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 \ \mathbf{x}_1 \\ \vdots \ \vdots \\ 1 \ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1D} \\ \vdots & & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{ND} \end{pmatrix}, \mathbf{w} = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

Linear Regression: In Higher Dimensions

- Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- Fit each training example $(\mathbf{x}_i, y_i)$ using the linear model

$$y_i = b + \mathbf{w}^T \mathbf{x}_i$$

- A bit of notation abuse: write $\mathbf{w} = [b, \mathbf{w}]$, write $\mathbf{x}_i = [1, \mathbf{x}_i]$

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Switching to matrix notation, the relationship becomes: $\mathbf{Y} = \mathbf{X}\mathbf{w}$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & \mathbf{x}_1 \\ \vdots & \vdots \\ 1 & \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} 1 & x_{11} \cdots x_{1D} \\ \vdots & \ddots & \vdots \\ 1 & x_{N1} \cdots x_{ND} \end{pmatrix}, \mathbf{w} = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

- $\mathbf{Y}$: $N \times 1$, $\mathbf{X}$: $N \times (D+1)$, $\mathbf{w}$: $(D+1) \times 1$

Linear Regression: The Objective Function

- Parameter $\mathbf{w}$ that satisfies $y_i = \mathbf{w}^T\mathbf{x}_i$ *exactly* for each i may not exist

Linear Regression: The Objective Function

- Parameter $\mathbf{w}$ that satisfies $y_i = \mathbf{w}^T \mathbf{x}_i$ *exactly* for each i may not exist
- So we look for the closest approximation

Linear Regression: The Objective Function

- Parameter $\mathbf{w}$ that satisfies $y_i = \mathbf{w}^T\mathbf{x}_i$ *exactly* for each i may not exist
- So we look for the closest approximation
- Specifically, $\mathbf{w}$ that minimizes the following sum-of-squared-differences between the truth (y_i) and the predictions ($\mathbf{w}^T\mathbf{x}_i$), just as we did for the one-dimensional case:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

Linear Regression: The Objective Function

- Parameter $\mathbf{w}$ that satisfies $y_i = \mathbf{w}^T \mathbf{x}_i$ *exactly* for each i may not exist
- So we look for the closest approximation
- Specifically, $\mathbf{w}$ that minimizes the following sum-of-squared-differences between the truth (y_i) and the predictions ($\mathbf{w}^T \mathbf{x}_i$), just as we did for the one-dimensional case:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- Following the matrix notation, we can write the above as:

$$E(\mathbf{w}) = \frac{1}{2} (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w})$$

Linear Regression: Least-Squares Solution

- Taking derivative w.r.t $\mathbf{w}$, and equating to zero, we get

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \\ \Longrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \end{aligned}$$

Linear Regression: Least-Squares Solution

- Taking derivative w.r.t $\mathbf{w}$, and equating to zero, we get

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \\ \Longrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \end{aligned}$$

- Taking inverse on both sides, we get the solution

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}}$$

- The above is also called the least-squares solution (since we minimized a sum-of-squared-differences objective)

Linear Regression: Least-Squares Solution

- Taking derivative w.r.t $\mathbf{w}$, and equating to zero, we get

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \\ \Longrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \end{aligned}$$

- Taking inverse on both sides, we get the solution

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}}$$

- The above is also called the least-squares solution (since we minimized a sum-of-squared-differences objective)
- **Note:** The same solution holds even if the responses are vector-valued (assume K responses per input)
  - $\mathbf{Y}$ will be an $N \times K$ matrix (assuming K responses per input)
  - $\mathbf{w}$ will be a $D \times K$ matrix (k-th column is the weight vector for the k-th response variable)

Linear Regression: Complexity Control

- We minimized the sum-of-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$$

- There is no control on the values the elements of **w** can take

Linear Regression: Complexity Control

- We minimized the sum-of-squares objective for linear regression
$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$$
- There is no control on the values the elements of **w** can take
- Problem: The w_i's can get very large trying to fit training data
  - Implications: The model becomes complex
  - Result: The model may lead to overfitting

Linear Regression: Complexity Control

- We minimized the sum-of-squares objective for linear regression
$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$$
- There is no control on the values the elements of **w** can take
- Problem: The w_i's can get very large trying to fit training data
  - Implications: The model becomes complex
  - Result: The model may lead to overfitting
- Solution: Penalize large values of the parameters/coefficients w_i's
  - Penalizing amounts to doing complexity control (also called regularization)
  - Leads to better generalization

Linear Regression: Complexity Control

- We minimized the sum-of-squares objective for linear regression
$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$$
- There is no control on the values the elements of $\mathbf{w}$ can take
- Problem: The w_i's can get very large trying to fit training data
  - Implications: The model becomes complex
  - Result: The model may lead to overfitting
- Solution: Penalize large values of the parameters/coefficients w_i's
  - Penalizing amounts to doing complexity control (also called regularization)
  - Leads to better generalization
- Penalizing the squared norm $\mathbf{w}^T\mathbf{w}$ is a common choice (called ℓ_2 norm)
$$\mathbf{w}^T\mathbf{w} = \sum_{j=1}^{D} w_j^2$$

Linear Regression: Complexity Control

- We minimized the sum-of-squares objective for linear regression
$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$$
- There is no control on the values the elements of $\mathbf{w}$ can take
- Problem: The w_i's can get very large trying to fit training data
  - Implications: The model becomes complex
  - Result: The model may lead to overfitting
- Solution: Penalize large values of the parameters/coefficients w_i's
  - Penalizing amounts to doing complexity control (also called regularization)
  - Leads to better generalization
- Penalizing the squared norm $\mathbf{w}^T\mathbf{w}$ is a common choice (called ℓ_2 norm)
$$\mathbf{w}^T\mathbf{w} = \sum_{j=1}^{D} w_j^2$$
- **Note:** other form of penalization are also possible. For example:
  - Sum of absolute values of the coefficients: $\sum_{j=1}^{D} |w_j|$ (called ℓ_1 norm)

Linear Regression: The Regularized Objective Function

- The modified objective becomes

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- We minimize the sum of a loss function and a regularizer term

Linear Regression: The Regularized Objective Function

- The modified objective becomes

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- We minimize the sum of a loss function and a regularizer term
- The hyperparameter λ controls the amount of regularization
- **Important:** It's a standard way to control overfitting in supervised learning
- Common form of a penalized loss function in supervised learning looks like:

$$\boxed{E(\mathbf{w}) = \ell(\mathbf{X}, \mathbf{Y}, \mathbf{w}) + R(\mathbf{w})}$$

Linear Regression: The Regularized Objective Function

- The modified objective becomes

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- We minimize the sum of a loss function and a regularizer term
- The hyperparameter λ controls the amount of regularization
- **Important:** It's a standard way to control overfitting in supervised learning
- Common form of a penalized loss function in supervised learning looks like:

$$\boxed{E(\mathbf{w}) = \ell(\mathbf{X}, \mathbf{Y}, \mathbf{w}) + R(\mathbf{w})}$$

- The loss function $\ell(\mathbf{X}, \mathbf{Y}, \mathbf{w})$ is a measure of model-fit on the training data

Linear Regression: The Regularized Objective Function

- The modified objective becomes

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- We minimize the sum of a loss function and a regularizer term
- The hyperparameter λ controls the amount of regularization
- **Important:** It's a standard way to control overfitting in supervised learning
- Common form of a penalized loss function in supervised learning looks like:

$$\boxed{E(\mathbf{w}) = \ell(\mathbf{X}, \mathbf{Y}, \mathbf{w}) + R(\mathbf{w})}$$

- The loss function $\ell(\mathbf{X}, \mathbf{Y}, \mathbf{w})$ is a measure of model-fit on the training data
- The regularizer $R(\mathbf{w})$ prevents the model from becoming too complex

Linear Regression: The Regularized Objective Function

- The modified objective becomes

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- We minimize the sum of a loss function and a regularizer term
- The hyperparameter λ controls the amount of regularization
- **Important:** It's a standard way to control overfitting in supervised learning
- Common form of a penalized loss function in supervised learning looks like:

$$\boxed{E(\mathbf{w}) = \ell(\mathbf{X}, \mathbf{Y}, \mathbf{w}) + R(\mathbf{w})}$$

- The loss function $\ell(\mathbf{X}, \mathbf{Y}, \mathbf{w})$ is a measure of model-fit on the training data
- The regularizer $R(\mathbf{w})$ prevents the model from becoming too complex
- Regularization is particularly important for small N, large D

Linear Regression: The Regularized Objective Function

Coming back to the penalized least-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Linear Regression: The Regularized Objective Function

Coming back to the penalized least-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Taking derivative w.r.t. $\mathbf{w}$ and equating to zero gives:

$$\begin{aligned}
\nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = 0 \\
\implies \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \\
\implies (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{Y}
\end{aligned}$$

Linear Regression: The Regularized Objective Function

Coming back to the penalized least-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Taking derivative w.r.t. **w** and equating to zero gives:

$$\begin{aligned}
\nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = 0 \\
\implies \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \\
\implies (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{Y}
\end{aligned}$$

- Taking inverse on both sides, we get the solution

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}}$$

Linear Regression: The Regularized Objective Function

Coming back to the penalized least-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Taking derivative w.r.t. $\mathbf{w}$ and equating to zero gives:

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = 0 \\ \Longrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \\ \Longrightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \end{aligned}$$

- Taking inverse on both sides, we get the solution

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}}$$

- Penalized linear regression is also known as ridge regression

Linear Regression: The Regularized Objective Function

Coming back to the penalized least-squares objective for linear regression

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Taking derivative w.r.t. $\mathbf{w}$ and equating to zero gives:

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = 0 \\ \Longrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \\ \Longrightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{Y} \end{aligned}$$

- Taking inverse on both sides, we get the solution

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}}$$

- Penalized linear regression is also known as ridge regression

- Ridge regression also useful when $\mathbf{X}^T\mathbf{X}$ is not invertible
  - Standard least-squares solution $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ will not be valid
  - Adding the $\lambda\mathbf{I}$ makes $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ invertibe

Linear Regression: Gradient Descent Solution

- Recall: solving for $\mathbf{w}$ requires inverting $D \times D$ matrices $\mathbf{X}^T\mathbf{X}$ or $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$
- Matrix inversion can be expensive if data dimensionality D is large

Linear Regression: Gradient Descent Solution

- Recall: solving for $\mathbf{w}$ requires inverting $D \times D$ matrices $\mathbf{X}^T\mathbf{X}$ or $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$
- Matrix inversion can be expensive if data dimensionality D is large
- One solution: *Iterative* minimization of the loss function
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$: Linear Regression

Linear Regression: Gradient Descent Solution

- Recall: solving for $\mathbf{w}$ requires inverting $D \times D$ matrices $\mathbf{X}^T\mathbf{X}$ or $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$
- Matrix inversion can be expensive if data dimensionality D is large
- One solution: *Iterative* minimization of the loss function
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$: Linear Regression
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$: Ridge Regression

Linear Regression: Gradient Descent Solution

- Recall: solving for $\mathbf{w}$ requires inverting $D \times D$ matrices $\mathbf{X}^T\mathbf{X}$ or $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$
- Matrix inversion can be expensive if data dimensionality D is large
- One solution: *Iterative* minimization of the loss function
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$: Linear Regression
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$: Ridge Regression
- How: Using Gradient Descent (GD)
- A *general recipe* for iteratively optimizing similar loss functions

Linear Regression: Gradient Descent Solution

- Recall: solving for $\mathbf{w}$ requires inverting $D \times D$ matrices $\mathbf{X}^T\mathbf{X}$ or $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$
- Matrix inversion can be expensive if data dimensionality D is large
- One solution: *Iterative* minimization of the loss function
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w})$: Linear Regression
  - $E(\mathbf{w}) = \frac{1}{2}(\mathbf{Y} - \mathbf{X}\mathbf{w})^T(\mathbf{Y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$: Ridge Regression
- How: Using Gradient Descent (GD)
- A *general recipe* for iteratively optimizing similar loss functions
- Gradient Descent rule:
  - Initialize the weight vector $\mathbf{w} = \mathbf{w}^0$
  - Update $\mathbf{w}$ by moving along the direction of negative gradient $-\frac{\partial \mathbf{E}}{\partial \mathbf{w}}$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}}$$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \\
&= \mathbf{w} - \alpha \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{Y})
\end{aligned}
$$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$
\begin{aligned}
\mathbf{w} &= \mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \\
&= \mathbf{w} - \alpha \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{Y}) \\
&= \mathbf{w} - \alpha \sum_{i=1}^{N} \mathbf{x}_i (\mathbf{w}^T \mathbf{x}_i - y_i)
\end{aligned}
$$

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$\begin{aligned} \mathbf{w} &= \mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \alpha \mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{Y}) \\ &= \mathbf{w} - \alpha \sum_{i=1}^{N} \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i) \end{aligned}$$

- α is the learning rate
- **Stop:** When some criteria is met (e.g., max. # of iterations), or the rate of decrease of **E** falls below some threshold
  - Small α: slow convergence but small residual error
  - Large α: fast convergence but large residual error

Linear Regression: Gradient Descent Solution

- Initialize $\mathbf{w} = \mathbf{w}^0$
- Repeat until convergence:

$$\begin{aligned}\mathbf{w} &= \mathbf{w} - \alpha \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \alpha \mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{Y}) \\ &= \mathbf{w} - \alpha \sum_{i=1}^{N} \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)\end{aligned}$$

- α is the learning rate
- **Stop:** When some criteria is met (e.g., max. # of iterations), or the rate of decrease of **E** falls below some threshold
  - Small α: slow convergence but small residual error
  - Large α: fast convergence but large residual error
- Note that convergence rate depends on the *error* at each iteration
  - Error over all examples: $\sum_{i=1}^{N} \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)$

Linear Regression: Gradient Descent Solution

- The least-squares linear regression objective is a convex function
  - It has a unique minimum

Linear Regression: Gradient Descent Solution

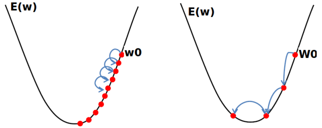
- The least-squares linear regression objective is a convex function
  - It has a unique minimum
  - Gradient descent will find the unique minimum (or get very close it to, depending in the learning rate α)

Linear Regression: Gradient Descent Solution

- The least-squares linear regression objective is a convex function
    - It has a unique minimum
    - Gradient descent will find the unique minimum (or get very close it to, depending in the learning rate α)
    - For general functions, GD can only find a local minimum

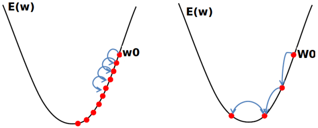
Linear Regression: Gradient Descent Solution

- The least-squares linear regression objective is a convex function
  - It has a unique minimum
  - Gradient descent will find the unique minimum (or get very close it to, depending in the learning rate α)
  - For general functions, GD can only find a local minimum
- Effect of the learning rate α (left: small α, right: large α)



Linear Regression: Gradient Descent Solution

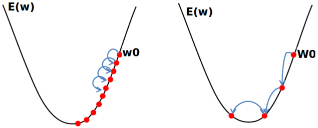
- The least-squares linear regression objective is a convex function
  - It has a unique minimum
  - Gradient descent will find the unique minimum (or get very close it to, depending in the learning rate α)
  - For general functions, GD can only find a local minimum
- Effect of the learning rate α (left: small α, right: large α)



- **Stochastic Gradient Descent (SGD):** Variant of GD which computes the gradient of $E(\mathbf{w})$ w.r.t. a single training example and thus allows updating $\mathbf{w}$ using one example at a time (unlike GD which uses *all the data* to make each update of $\mathbf{w}$). SGD for linear regression looks like:
  - `repeat-while-converged` {for i=1:N $\{\mathbf{w} - \alpha \mathbf{x}_i(\mathbf{w}^T \mathbf{x}_i - y_i)\}$}

Linear Regression: Gradient Descent Solution

- The least-squares linear regression objective is a convex function
  - It has a unique minimum
  - Gradient descent will find the unique minimum (or get very close it to, depending in the learning rate α)
  - For general functions, GD can only find a local minimum
- Effect of the learning rate α (left: small α, right: large α)



- **Stochastic Gradient Descent (SGD):** Variant of GD which computes the gradient of $E(\mathbf{w})$ w.r.t. a single training example and thus allows updating $\mathbf{w}$ using one example at a time (unlike GD which uses *all the data* to make each update of $\mathbf{w}$). SGD for linear regression looks like:
  - `repeat-while-converged` {for i=1:N $\{\mathbf{w} - \alpha \mathbf{x}_i(\mathbf{w}^T \mathbf{x}_i - y_i)\}$}
- Note: SGD is usually more efficient than GD and also converges faster

Next class..

- Linear Classifiers
  - Hyperplane based class separators
  - The Perceptron algorithm
  - Maximum Margin Hyperplanes: Introduction to Support Vector Machines