STA141C: Big Data & High Performance Statistical Computing

Lecture 7: Linear Regression, Linear System Solvers

Cho-Jui Hsieh
UC Davis

May 9, 2017

Linear Regression

Regression

- Input: training data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ and corresponding outputs $y_1, y_2, \ldots, y_n \in \mathbb{R}$
- Training: compute a function f such that $f(\boldsymbol{x}_i) \approx y_i$ for all i
- Prediction: given a testing sample $\tilde{\boldsymbol{x}}$, predict the output as $f(\tilde{\boldsymbol{x}})$
- Examples:
  - Income, number of children $\Rightarrow$ Consumer spending
  - Processes, memory $\Rightarrow$ Power consumption
  - Financial reports $\Rightarrow$ Risk
  - Atmospheric conditions $\Rightarrow$ Precipitation

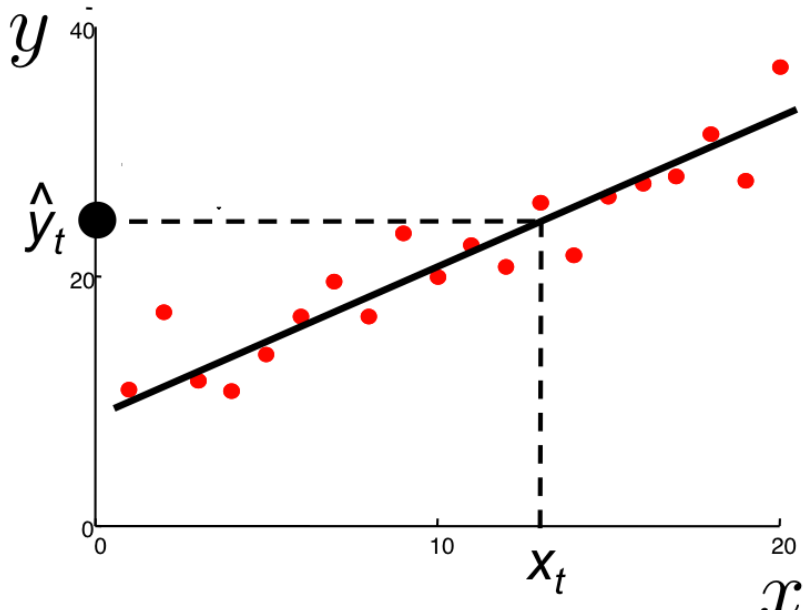
Linear Regression

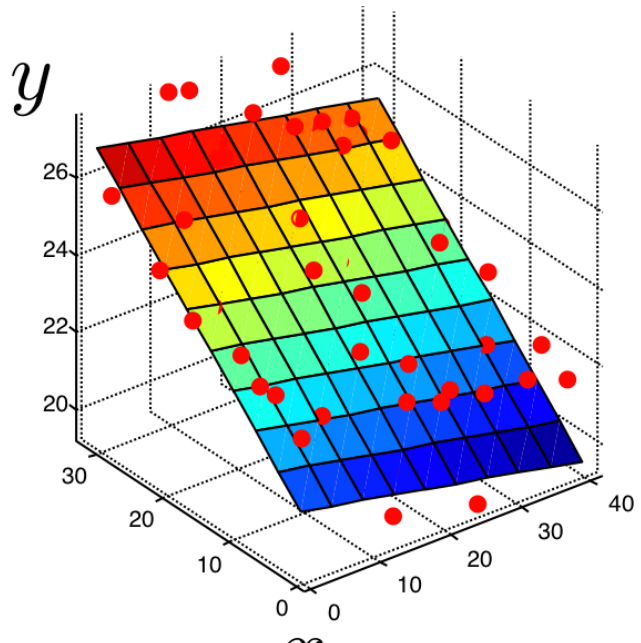
- Assume $f(\cdot)$ is a linear function parameterized by $\boldsymbol{w} \in \mathbb{R}^d$:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

- Training: compute the model $\boldsymbol{w} \in \mathbb{R}^d$ such that $\boldsymbol{w}^T \boldsymbol{x}_i \approx y_i$ for all i
- Prediction: given a testing sample $\tilde{\boldsymbol{x}}$, the prediction value is $\boldsymbol{w}^T \tilde{\boldsymbol{x}}$
- How to find $\boldsymbol{w}$?

$$\boldsymbol{w}^* = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$





Linear Regression: probability interpretation

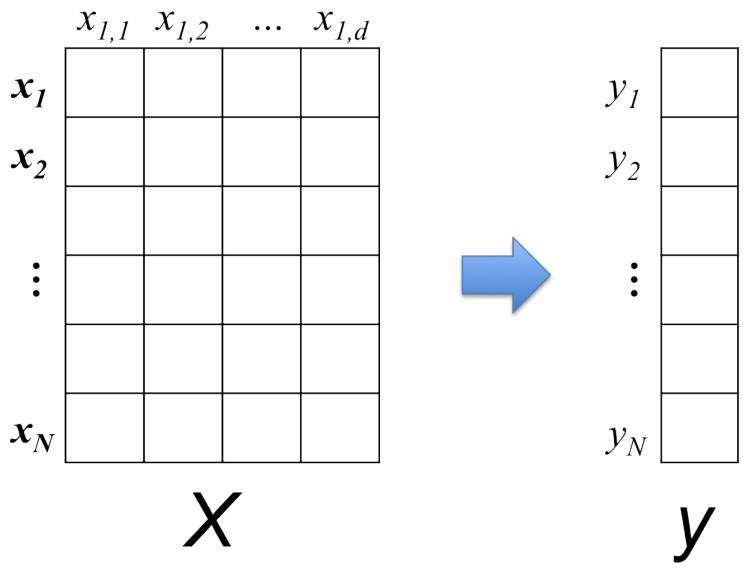
- Assume the data is generated from the probability model:

$$y_i \sim \boldsymbol{w}^T \boldsymbol{x}_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

- Maximum likelihood estimator:

$$\begin{aligned}
\boldsymbol{w}^* &= \operatorname*{argmax}_{\boldsymbol{w}} \log P(y_1, \ldots, y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{w}) \\
&= \operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{n} \log P(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \\
&= \operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}} e^{-(\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2/2}\right) \\
&= \operatorname*{argmax}_{\boldsymbol{w}} \sum_{i=1}^{n} -\frac{1}{2}(\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \text{constant} \\
&= \operatorname*{argmin}_{\boldsymbol{w}} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2
\end{aligned}$$

Linear Regression: written as a matrix form



- Linear regression: $\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$
- Matrix form: let $X \in \mathbb{R}^{n \times d}$ be the matrix where the i-th row is $\boldsymbol{x}_i$, $\mathbf{y} = [y_1, \ldots, y_n]^T$, then linear regression can be written as

$$\boldsymbol{w}^* = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\text{argmin}} \, \|X\boldsymbol{w} - \mathbf{y}\|_2^2$$

Solving Linear Regression

- Minimize the sum of squared error $J(\boldsymbol{w})$

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 \\ &= \frac{1}{2}(X\boldsymbol{w} - \mathbf{y})^T(X\boldsymbol{w} - \mathbf{y}) \\ &= \frac{1}{2}\boldsymbol{w}^T X^T X\boldsymbol{w} - \mathbf{y}^T X\boldsymbol{w} + \frac{1}{2}\mathbf{y}^T\mathbf{y} \end{aligned}$$

- Derivative: $\frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{w}) = X^T X\boldsymbol{w} - X^T\mathbf{y}$
- Setting the derivative equal to zero gives the normal equation

$$X^T X\boldsymbol{w}^* = X^T\mathbf{y}$$

- Therefore, $\boldsymbol{w}^* = (X^T X)^{-1} X^T\mathbf{y}$

Solving Linear Regression

- Minimize the sum of squared error $J(\boldsymbol{w})$

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 \\ &= \frac{1}{2}(X\boldsymbol{w} - \mathbf{y})^T(X\boldsymbol{w} - \mathbf{y}) \\ &= \frac{1}{2}\boldsymbol{w}^T X^T X\boldsymbol{w} - \mathbf{y}^T X\boldsymbol{w} + \frac{1}{2}\mathbf{y}^T\mathbf{y} \end{aligned}$$

- Derivative: $\frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{w}) = X^T X\boldsymbol{w} - X^T\mathbf{y}$
- Setting the derivative equal to zero gives the normal equation

$$X^T X\boldsymbol{w}^* = X^T\mathbf{y}$$

- Therefore, $\boldsymbol{w}^* = (X^T X)^{-1} X^T\mathbf{y}$

  but $X^T X$ may be non-invertible ...

Linear System Solver

- Linear System: given $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, solve (find a $\boldsymbol{x} \in \mathbb{R}^n$ such that)

$$A\boldsymbol{x} = \boldsymbol{b}$$

- Three cases:
  - A is invertible ($m = n$ and full rank)
    $\Rightarrow$ unique solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$

Linear System Solver

- Linear System: given $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, solve (find a $\boldsymbol{x} \in \mathbb{R}^n$ such that)

$$A\boldsymbol{x} = \boldsymbol{b}$$

- Three cases:
  - A is invertible ($m = n$ and full rank)
    $\Rightarrow$ unique solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$
  - Under-determined system: $\text{rank}(A) = m$ but $n > m$
    $\Rightarrow$ multiple solutions, usually want to find the "least norm" solution.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}.$$

Linear System Solver

- Linear System: given $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, solve (find a $\boldsymbol{x} \in \mathbb{R}^n$ such that)

$$A\boldsymbol{x} = \boldsymbol{b}$$

- Three cases:
  - A is invertible ($m = n$ and full rank)
    $\Rightarrow$ unique solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$
  - Under-determined system: $\text{rank}(A) = m$ but $n > m$
    $\Rightarrow$ multiple solutions, usually want to find the "least norm" solution.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}.$$

  - Over-determined system: $\text{rank}(A) < m$
    $\Rightarrow$ (usually) no solution
    $\Rightarrow$ Output $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{b}\|^2$

Linear System Solver

- Linear System: given $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, solve (find a $\boldsymbol{x} \in \mathbb{R}^n$ such that)

$$A\boldsymbol{x} = \boldsymbol{b}$$

- Three cases:
  - A is invertible ($m = n$ and full rank)
    $\Rightarrow$ unique solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$
  - Under-determined system: $\text{rank}(A) = m$ but $n > m$
    $\Rightarrow$ multiple solutions, usually want to find the "least norm" solution.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}.$$

  - Over-determined system: $\text{rank}(A) < m$
    $\Rightarrow$ (usually) no solution
    $\Rightarrow$ Output $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{b}\|^2$
- Do not compute the inverse of a matrix!
  - Numerical problem
  - Time consuming

Linear System Solver

- First case: $A \in \mathbb{R}^{m \times m}$, A is invertible (full rank)
- Use "Gaussian elimination" or equivalently "LU factorization"
- Call "numpy.linalg.solve"

```
>>> a = np.array([3,1], [1,2]])
>>> b = np.array([9,8])
>>> x = np.linalg.solve(a, b)
>>> x
array([ 2.,  3.])
```

Linear System Solver

- Under-determined system and over-determined system: can be solved by SVD

$$
\begin{aligned}
A\boldsymbol{x} &= \boldsymbol{b} \\
U\Sigma V^T\boldsymbol{x} &= \boldsymbol{b} \\
\boldsymbol{x} &= V\Sigma^{\dagger}U^T\boldsymbol{b} \text{ (psuedo-inverse)} \\
\Sigma^{\dagger} &= \text{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \cdots, \sigma_k^{-1}, 0, \cdots, 0) \\
&\quad \text{(assume } A \text{ has } k \text{ nonzero singular values)}
\end{aligned}
$$

- Call "numpy.linalg.lstsq"

Linear System Solver

```
>>> a = scipy.rand(5,10)
>>> b = scipy.rand(5,1)
>>> x = numpy.linalg.lstsq(a,b)
>>> numpy.linalg.norm(a.dot(x[0]) - b)
7.4320704251928296e-16

>>> a = scipy.rand(5,3)
>>> b = scipy.rand(5,1)
>>> x = numpy.linalg.lstsq(a,b)
>>> numpy.linalg.norm(a.dot(x[0]) - b)
0.35374284817556079
```

Solve multiple linear systems

- Many times we need to solve

$$A\boldsymbol{x}_i = \boldsymbol{b}_i \text{ for all } i = 1, \cdots, N$$

- They can be solved altogether (so only one SVD or other decomposition is needed)

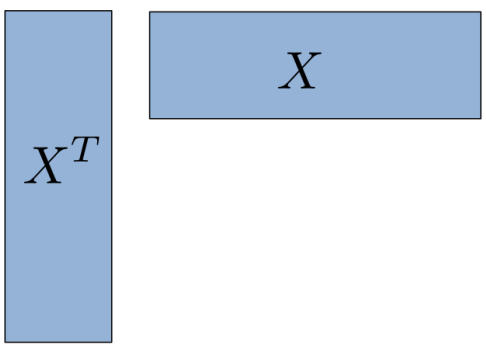
```
>>> a = scipy.rand(5,3)
>>> b = scipy.rand(5,4)
>>> x = numpy.linalg.lstsq(a,b)
>>> x[0]      ## solution for 5 linear systems
array([[ 0.15914526,  0.44365737,  0.31351924,  0.3476335 ],
       [ 0.30223114,  0.54325633,  0.22719821,  1.05852352],
       [ 0.07991735,  0.09856708,  0.08663738, -0.29111466]])
```

Solving Linear Regression

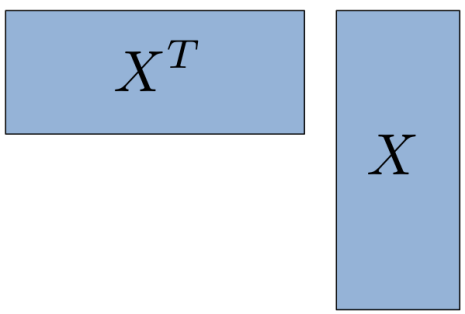
- Normal equation: $X^T X \boldsymbol{w}^* = X^T \mathbf{y}$
- If $X^T X$ is invertible (typically when # samples $>$ # features):

$$\boldsymbol{w}^* = (X^T X)^{-1} \mathbf{y}$$

- If $X^T X$ is low-rank (typically when # features $>$ # samples):
  infinite number of solutions
- In general, just use "numpy.linalg.lstsq"



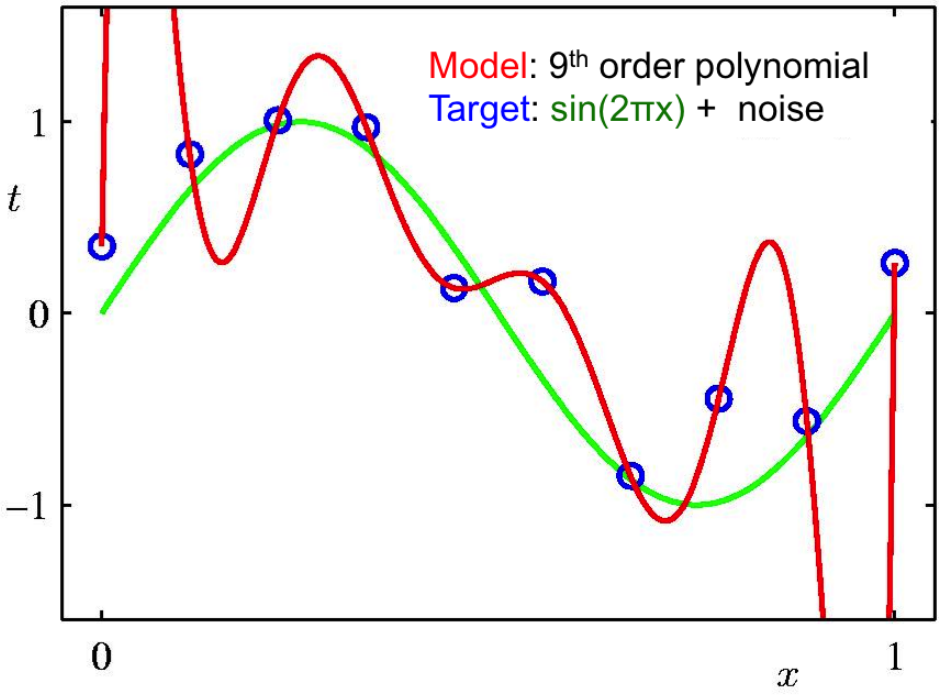
Underdetermined system
Infinite number of solutions

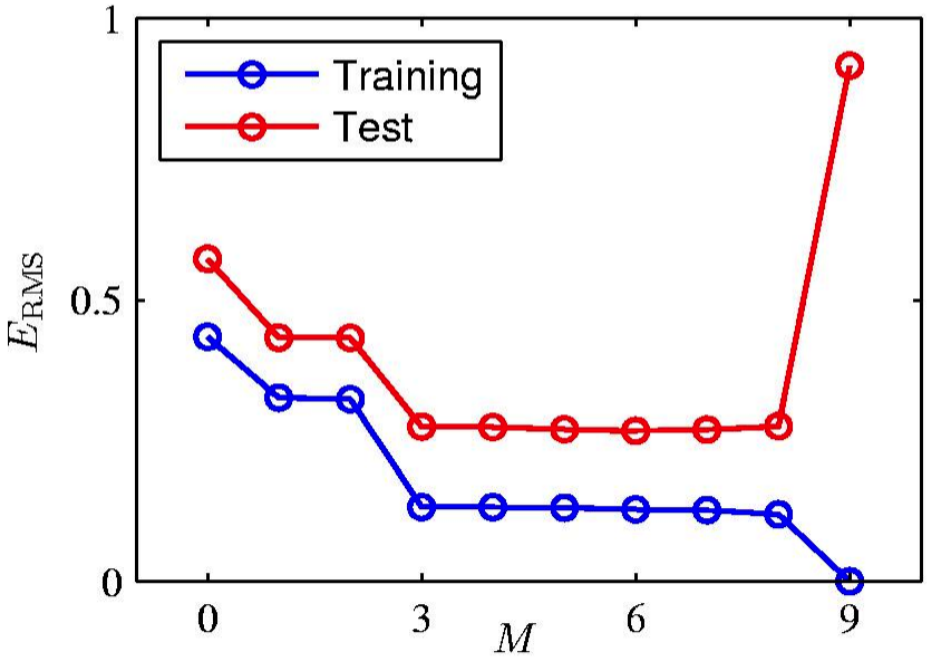


Overdetermined system
a unique solution

Regularized Linear Regression

Overfitting





- Overfitting: the model has low training error but high prediction error.
- Using too many features can lead to overfitting

Regularization to Avoid Overfitting

- Enforce the solution to have low L2-norm:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x}_i - y_i\|^2 \text{ s.t. } \|\boldsymbol{w}\|^2 \leq K$$

- Equivalent to the following problem with some λ

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x}_i - yi\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Regularized Linear Regression

- Regularized Linear Regression:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \|X\boldsymbol{w} - \mathbf{y}\|^2 + R(\boldsymbol{w})$$

  $R(\boldsymbol{w})$: regularization

- Ridge Regression (ℓ_2 regularization):

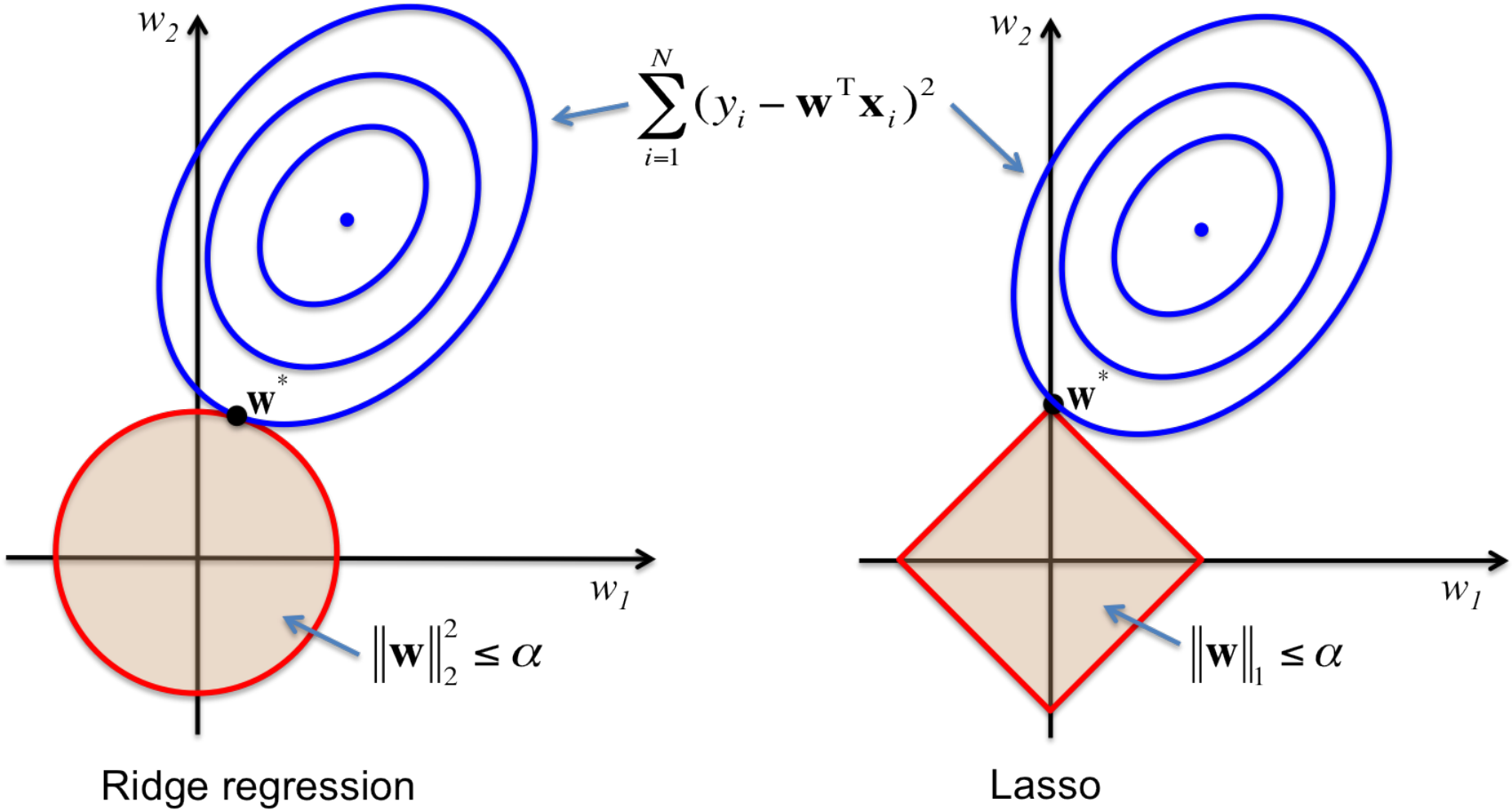
$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \|X\boldsymbol{w} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

- Lasso (ℓ_1 regularization):

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \|X\boldsymbol{w} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{w}\|_1$$

  Note that $\|\boldsymbol{w}\|_1 = \sum_{i=1}^{d} |w_i|$

Regularization



Lasso: the solution is sparse, but no closed form solution

Ridge Regression

- Ridge regression: $\operatorname{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \underbrace{\frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2}_{J(\boldsymbol{w})}$
- Closed form solution: optimal solution $\boldsymbol{w}^*$ satisfies $\nabla J(\boldsymbol{w}^*) = 0$:

$$X^T X \boldsymbol{w}^* - X^T \mathbf{y} + \lambda \boldsymbol{w}^* = 0$$
$$(X^T X + \lambda I)\boldsymbol{w}^* = X^T \mathbf{y}$$

- Optimial solution: $\boldsymbol{w}^* = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$
- Inverse always exists because $X^T X + \lambda I$ is positive definite
- What's the computational complexity?

Time Complexity

- When X is dense:
  - Closed form solution requires $O(nd^2 + d^3)$ if X is dense
  - Efficient if d is very small
  - Runs forever when $d > 100,000$
- Typical case for big data applications:
  - $X \in \mathbb{R}^{n \times d}$ is sparse with large n and large d
  - How can we solve the problem?

Coming up

- Optimization

Questions?