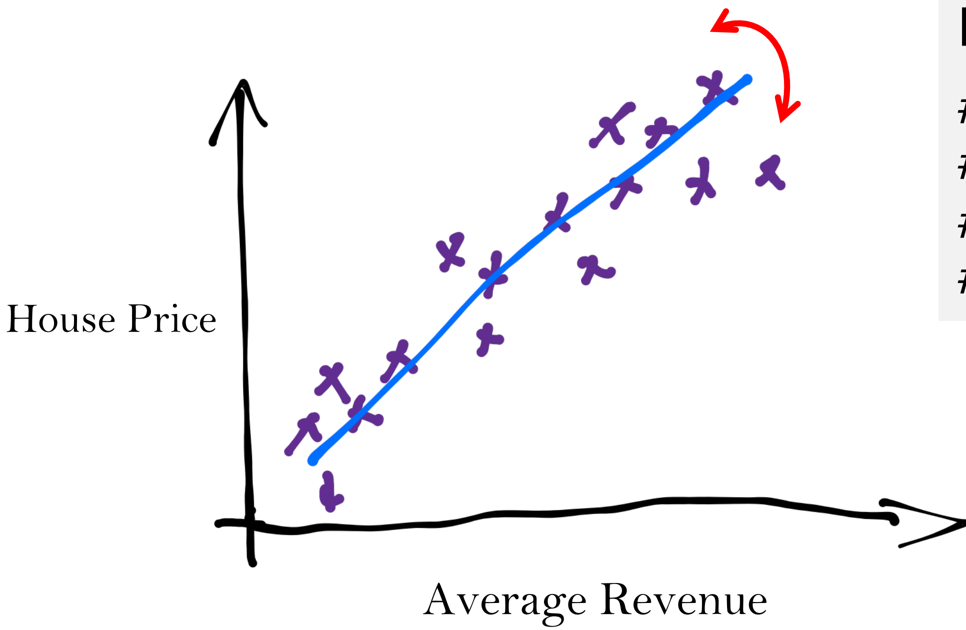
Machine Learning

Lecture 2: Linear Regression

Fall 2023

Instructor: Xiaodong Gu

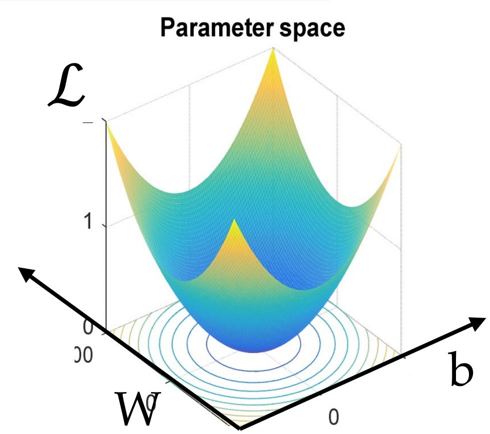
Recall: Key Elements of Machine Learning



Elements:

#1 Data (Experience)
#2 Model (Hypothesis)
#3 Loss Function (Objective)
#4 Optimization (Improve)

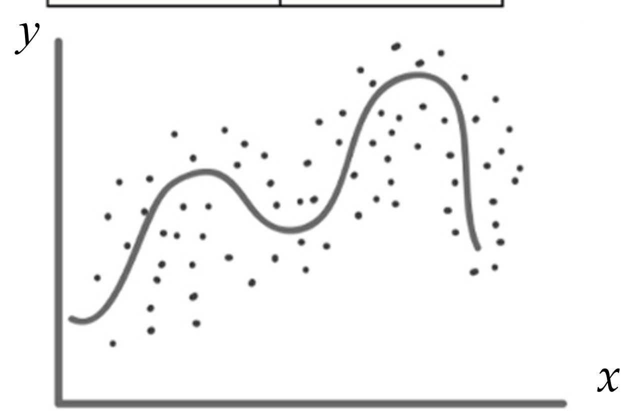
$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta|\mathcal{D})$$



Regression in Machine Learning

Machine Learning
- **Supervised Learning**
  - Regression (✓)
  - Classification
  - …
- Unsupervised Learning
- Reinforcement Learning

Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??



Regression: predicts real-valued labels

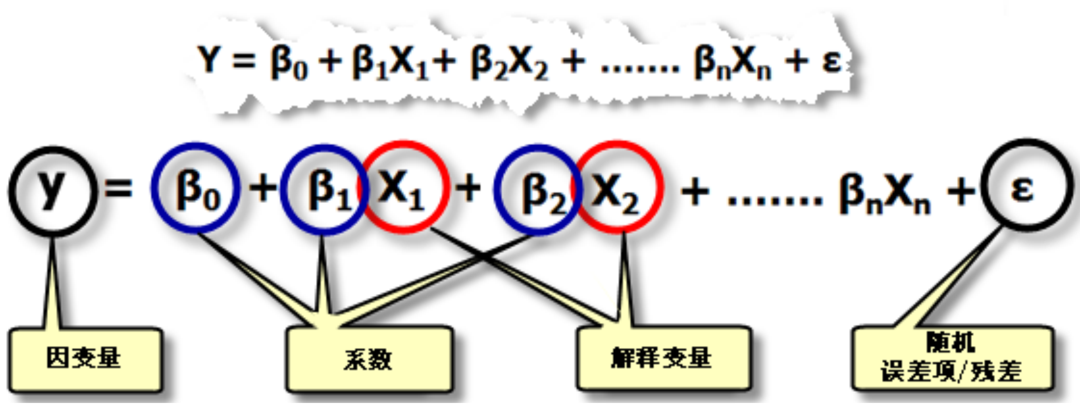
Regression

Why regression?

Regress the true value of a statistical variable through many experimentally observed values.

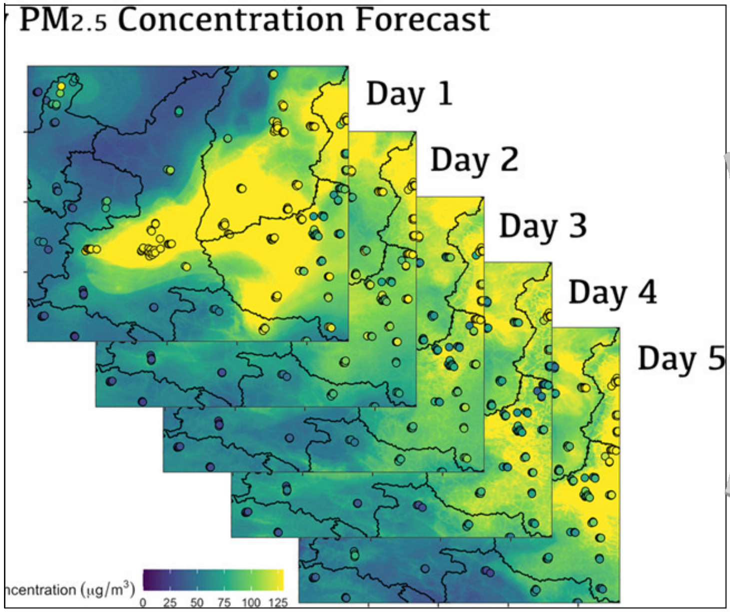
What is regression?

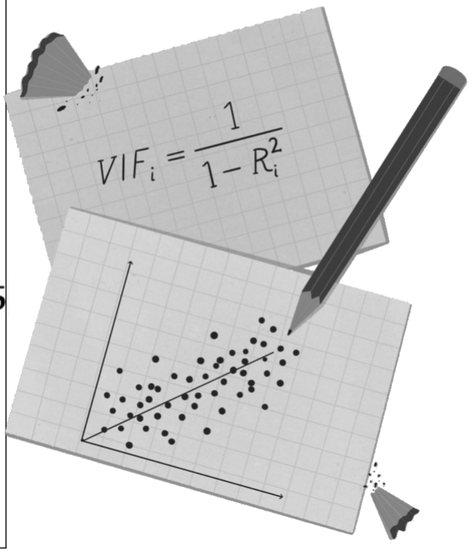
A function that describe the relationship between one **dependent** variable and a series of other (**independent**) variables.

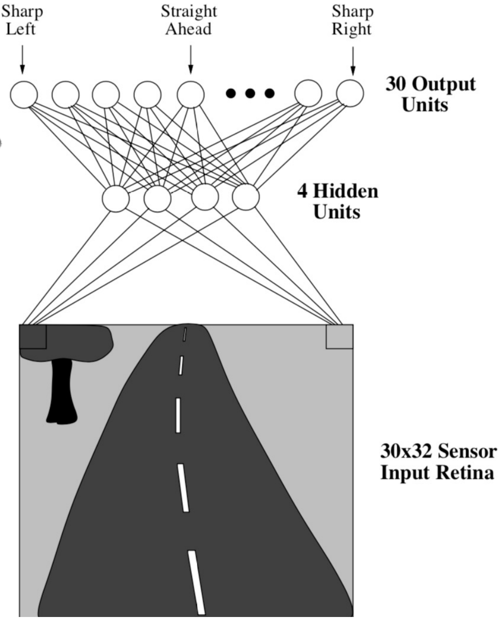


Applications of Regression

Forecasting | Factorization | Control | . . .



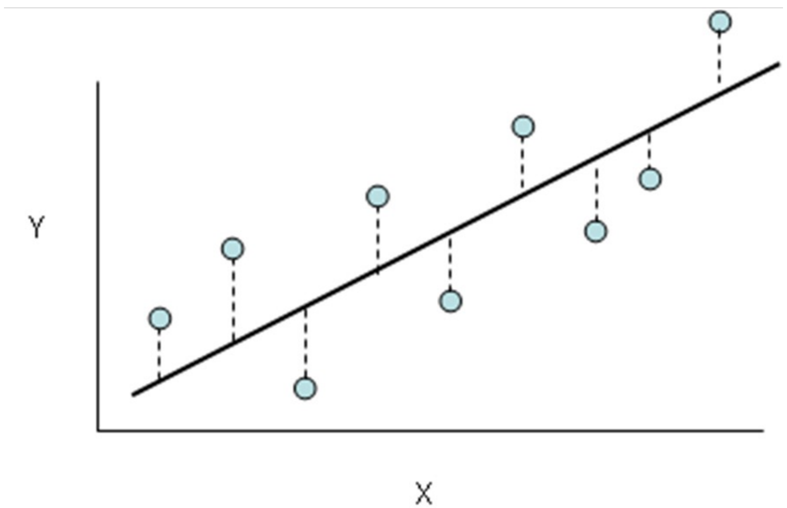


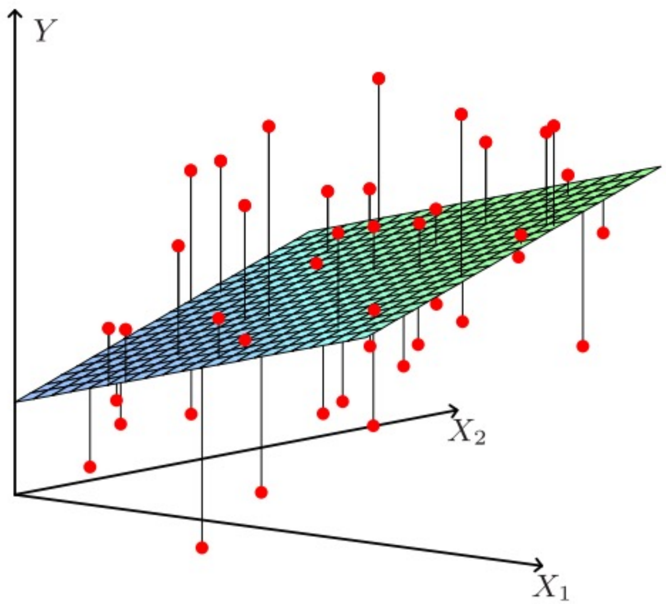


Linear Regression

A linear function for regression

$$y = f(x) = \boldsymbol{w}^T x + w_0$$

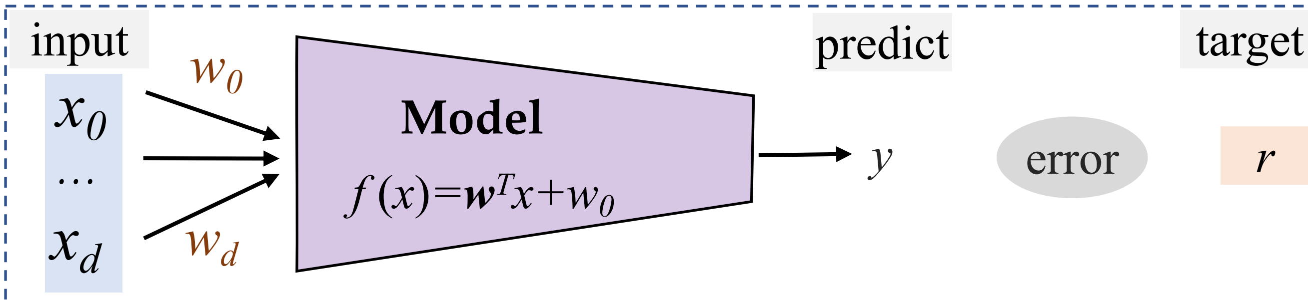




Linear model for regression is a (d+1)-dimensional hyperplane

Model Architecture

A simple linear function.



- Train:
  - estimate the parameters $\mathbf{w}$ and w_0 from data
- Test:
  - calculate $f(x) = \mathbf{w}^{\mathsf{T}} x + w_0$.

Loss Function

- For a given input x, the model outputs a real value y. Let $r \in \mathrm{R}$ be target value, the square error is :

$$l\,(\boldsymbol{w}, w_0 \mid x, r) = (r - y)^2$$

- Given: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(\mathrm{N})}, r^{(\mathrm{N})})\}$, the loss over the dataset is defined as the mean square error (MSE):

$$L(\boldsymbol{w}, w_0 \mid D) = \frac{1}{2\mathrm{N}} \sum_{\ell=1}^{N} (r^{(\ell)} - y^{(\ell)})^2$$

Optimization

Given: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(N)}, r^{(N)})\}$

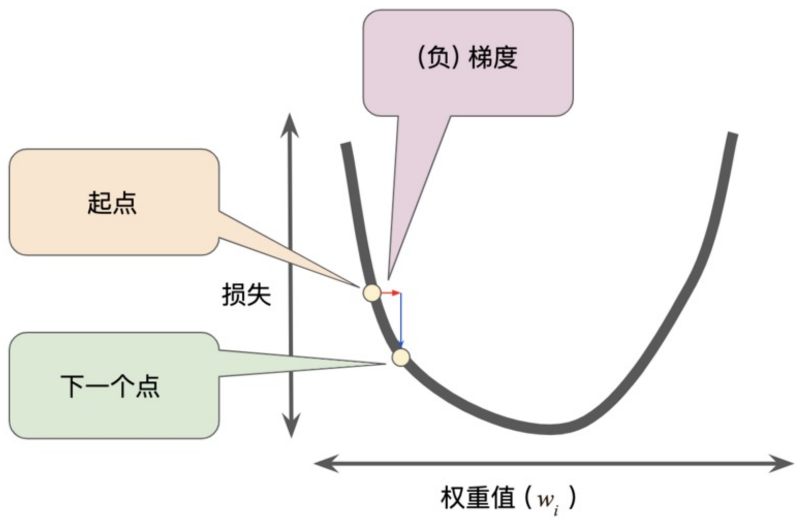
minimize the loss function using gradient descend:

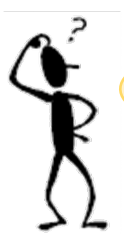
- Goal:

$$\min_w L(\boldsymbol{w})$$

- Iteration:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \frac{\partial L}{\partial w}$$





What is $\frac{\partial L}{\partial w}$?

Optimization – Gradient Descend

$$L(\boldsymbol{w}, w_0 \mid D) = -\,{}^1\!/_{2N} \sum_{\ell=1}^{N} (r^{(\ell)} - y^{(\ell)})^2$$

What is $\frac{\partial L}{\partial w}$?

For each $\boldsymbol{w}_j$ ($j = 1, \ldots, d$):

$$\frac{\partial L}{\partial w_j} = -\frac{1}{N} \underbrace{\sum_{\ell} \left(r^{(\ell)} - y^{(\ell)}\right) \frac{\partial y^{(\ell)}}{\partial w_j}}_{\text{Chain rule}} = -\frac{1}{N} \sum_{\ell} \left(r^{(\ell)} - y^{(\ell)}\right) x^{(\ell)}$$

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w}_{\text{old}} + \frac{1}{N} \sum_{\ell=1}^{N} (r^{(\ell)} - y^{(\ell)})\, x^{(\ell)}$$

The Algorithm

Gradient Descend for Liner Regression

Input: $D = \{(\mathrm{x}^{(l)}, \mathrm{r}^{(l)})\}$ $(l = 1{:}N)$

for $j = 0, \ldots, \mathrm{d}$

$\quad \boldsymbol{w}_{\mathrm{j}} \leftarrow rand\,(-0.01, 0.01)$

repeat

$\quad$ **for** $j = 0, \ldots, d$

$\quad\quad \Delta\boldsymbol{w}_{\mathrm{j}} \leftarrow 0$

$\quad$ **for** $l = 1, \ldots, N$

$\quad\quad y \leftarrow 0$

$\quad\quad$ **for** $j = 0, \ldots, d$

$\quad\quad\quad y \leftarrow y + \boldsymbol{w}_{\mathrm{j}} x_{\mathrm{j}}^{(l)}$

$\quad\quad \Delta\boldsymbol{w}_{\mathrm{j}} \leftarrow \Delta\boldsymbol{w}_{\mathrm{j}} + (r^{(l)} - y) x_{\mathrm{j}}^{(l)}$

$\quad \Delta\boldsymbol{w}_{\mathrm{j}} = \Delta\boldsymbol{w}_{\mathrm{j}} / \mathrm{N}$

$\quad$ **for** $j = 0, \ldots, d$

$\quad\quad w_{\mathrm{j}} \leftarrow w_{\mathrm{j}} + \eta \Delta w_{\mathrm{j}}$

until convergence

The Matrix Form

Let $X = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(N)} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(N)} & x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$ $\quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$ $\quad \boldsymbol{r} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix}$

- Prediction: $\boldsymbol{y} = X\boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}^{(1)}\boldsymbol{w} \\ \boldsymbol{x}^{(2)}\boldsymbol{w} \\ \vdots \\ \boldsymbol{x}^{(N)}\boldsymbol{w} \end{bmatrix}$
- Objective: $L(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{r}-\boldsymbol{y})^T(\boldsymbol{r}-\boldsymbol{y}) = \frac{1}{2}(\boldsymbol{r}-X\boldsymbol{w})^T(\boldsymbol{r}-X\boldsymbol{w})$

The Matrix Form

- Gradient

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{X}^T(\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w})$$

- Solution

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0 \;\Rightarrow \boldsymbol{X}^T(\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}) = 0$$

$$\Rightarrow \boldsymbol{X}^T\boldsymbol{r} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$

$$\Rightarrow \boldsymbol{w}^{\star} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{r}$$

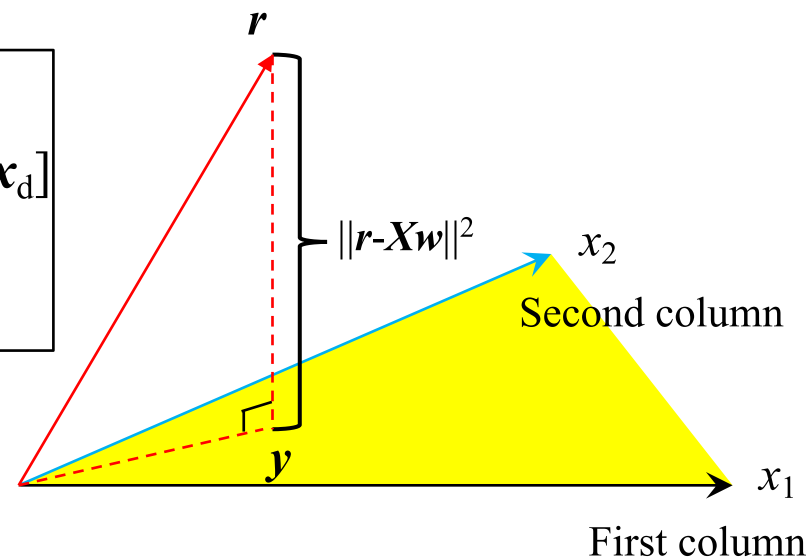
The Matrix Form

- Then the predicted values are

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\underbrace{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}}\boldsymbol{r} \\ &= \boldsymbol{H}\boldsymbol{r} \end{aligned}$$

Geometrical Explanation

- The column vectors $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\mathrm{d}}]$ form a subspace of $\mathbb{R}^{\mathrm{n}}$.
- **H** is a least square projection

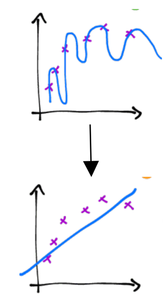


Matrix Form with Regularization

Problem: $X^{\mathrm{T}}X$ might be singular

When some column vectors are not independent (e.g., $\boldsymbol{x}_2=3\boldsymbol{x}_1$), then $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is singular, thus $\boldsymbol{w}^{\star} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{r}$ cannot be directly calculated.

Solution: Regularization



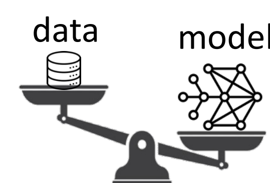
$$\boldsymbol{L}(\boldsymbol{w})=\frac{1}{2}(\boldsymbol{r}-\boldsymbol{y})^T(\boldsymbol{r}-\boldsymbol{y})=\frac{1}{2}(\boldsymbol{r}-\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{r}-\boldsymbol{X}\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

New gradient: $\dfrac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{X}^T(\boldsymbol{r}-\boldsymbol{X}\boldsymbol{w}) + \lambda\boldsymbol{w}$

New optimal solution: $\dfrac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0 \Rightarrow -\boldsymbol{X}^T(\boldsymbol{r}-\boldsymbol{X}\boldsymbol{w}) + \lambda\boldsymbol{w} = 0$

$$\Rightarrow \boldsymbol{X}^T\boldsymbol{r} = (\boldsymbol{X}^T\boldsymbol{X}+\lambda\mathbf{I})\boldsymbol{w}$$

$$\Rightarrow \boldsymbol{w}^{\star} = (\boldsymbol{X}^T\boldsymbol{X}+\lambda\mathbf{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{r}$$



Penalty to model turns out to be data augmentation (adding data prior)

Programming Time

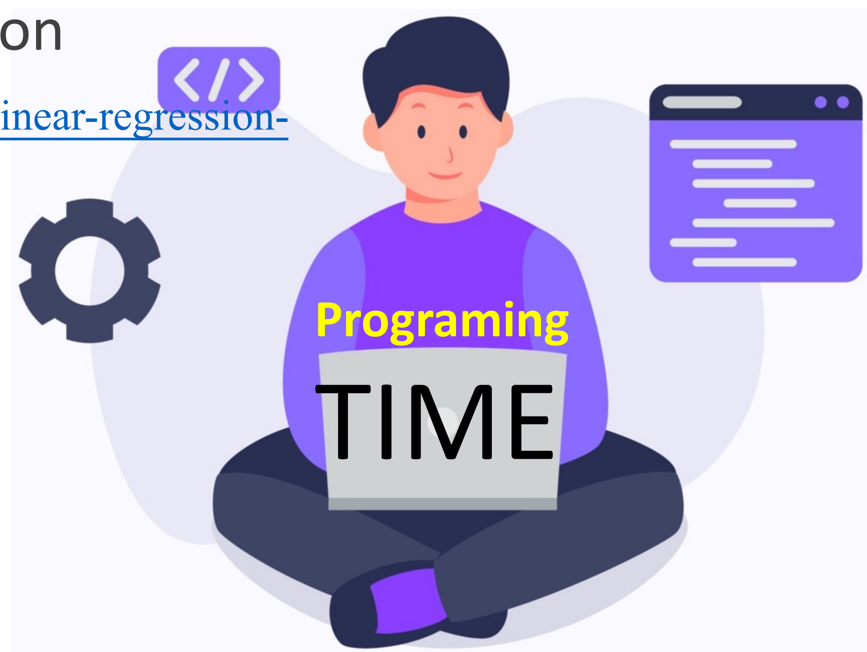
Tutorial:

Python

https://colab.research.google.com/github/cs231n/cs231n.github.io/blob/master/python-colab.ipynb

Linear regression with Python

https://www.kaggle.com/code/sudhirnl7/linear-regression-tutorial/data?select=insurance.csv



What's Next?

Classifications

Find a decision boundary that maximizes the margin between two classes.

Machine Learning

- **Supervised Learning**
  - Regression
  - Classification(✓)
  - …
- Unsupervised Learning
- Reinforcement Learning