https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 2: Linear Regression and Regularization

Shen Shen

Feb 7, 2025

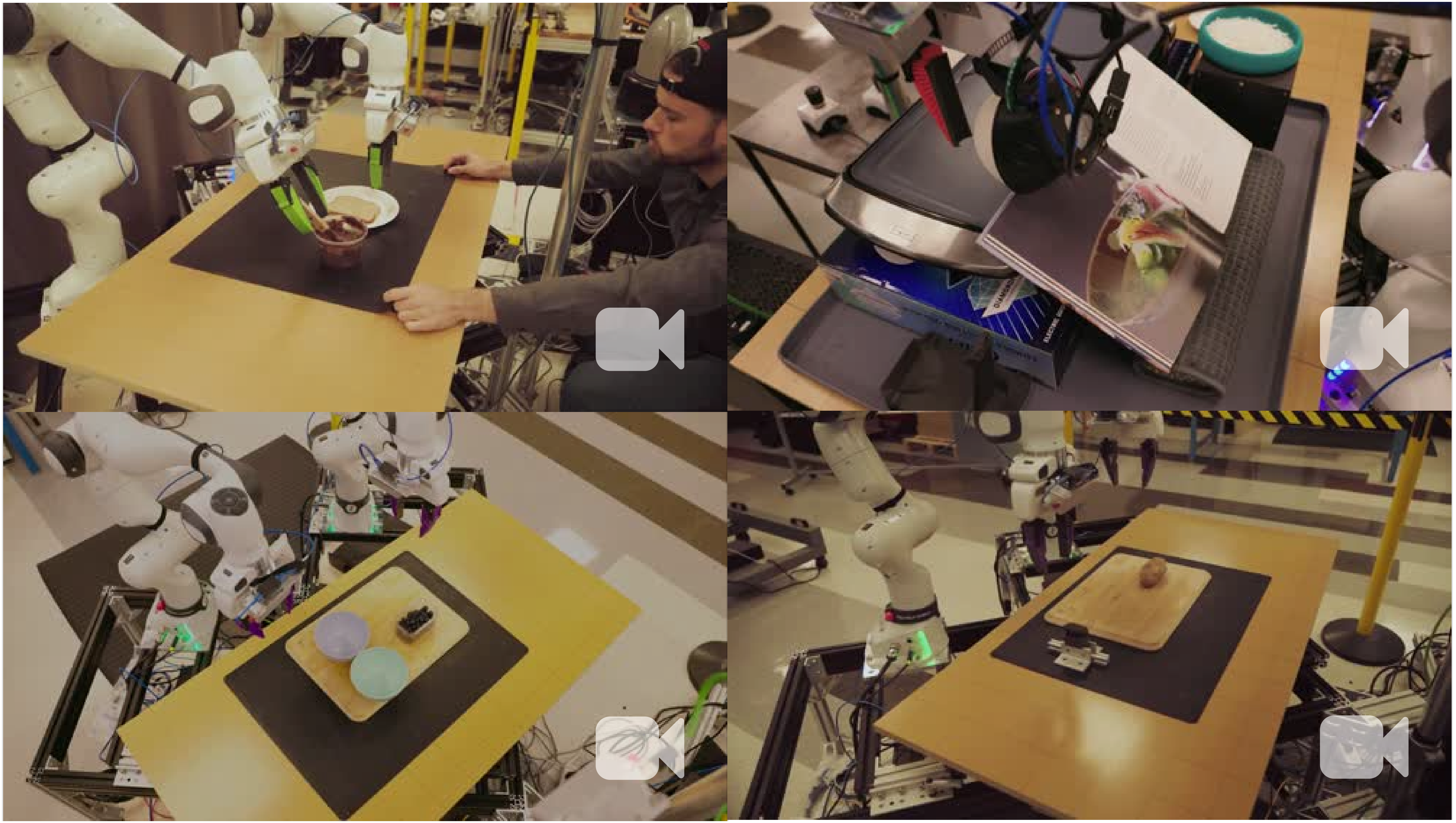
(11am, Room 10-250)

https://www.youtube.com/embed/kRPAGejCAeY?enablejsapi=1

DARPA Robotics Competition

2015

Optimization + first-principle physics



https://www.youtube.com/embed/fn3KWM1kuAw?start=1&enablejsapi=1

Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validation

Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validation

`Recall: pollution prediction example`

Training data:

$\mathcal{D}_{\text{train}} \quad \left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right\}$

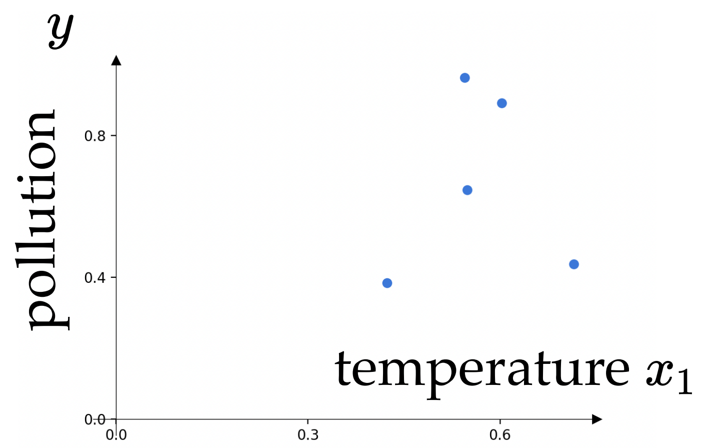
feature vector

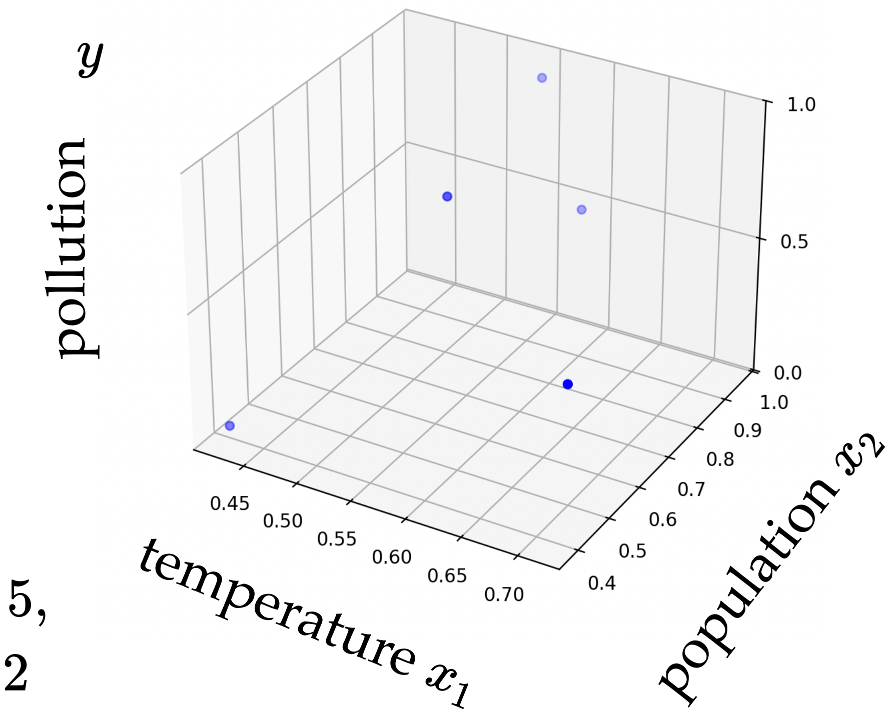
label

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_d^{(1)} \end{bmatrix} \in \mathbb{R}^d$$

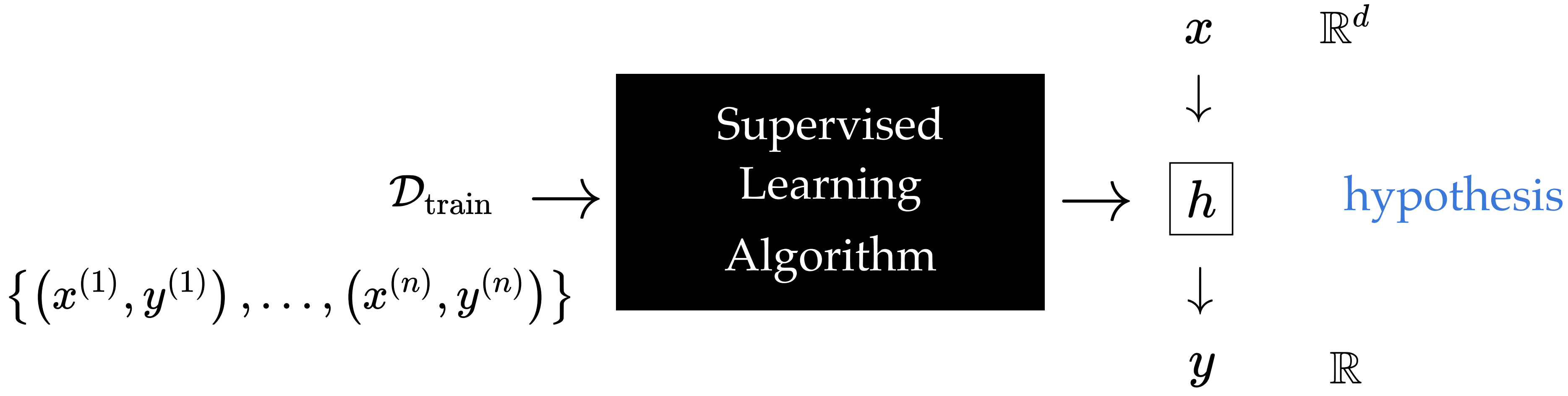
$y^{(1)} \in \mathbb{R}$

$n = 5,$
$d = 1$





$n = 5,$
$d = 2$



What do we want? A good way to label new features

For example, h : For any $x, h(x) = 1,000,000$, valid but is it any good?

Hypothesis class $\mathcal{H}$: set of h (or specifically for today, the set of hyperplanes)

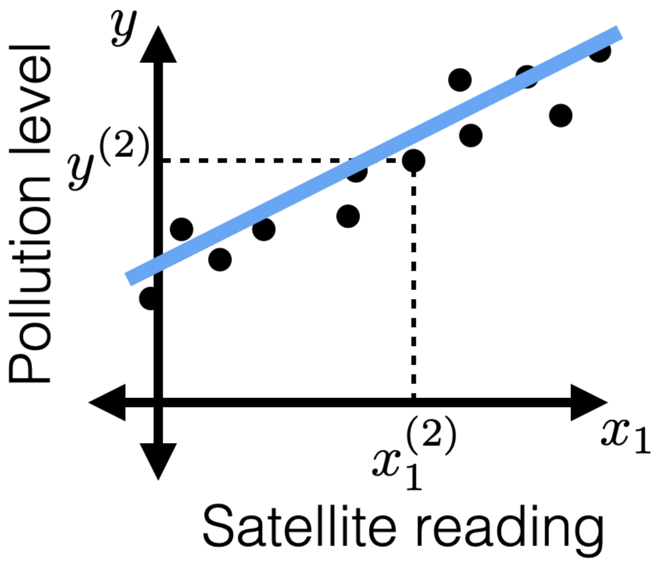
A linear regression hypothesis
:

data

$$h\left(x; \theta, \theta_0\right) = \theta^T x + \theta_0$$

$$= \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} + \theta_0$$

parameters

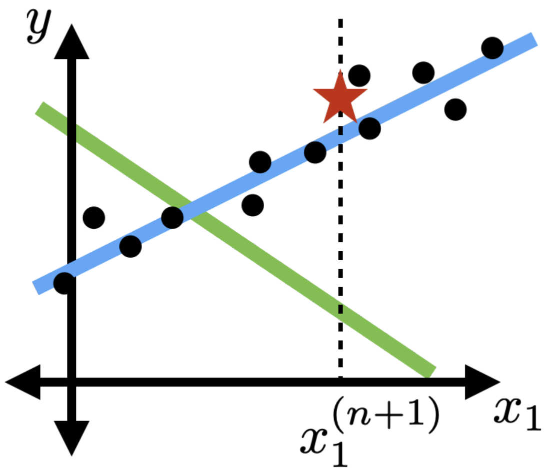


- Training error

$$\mathcal{E}_{\text{train}}\left(h\right) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right)$$

- Test error

n' new points



$$\mathcal{E}_{\text{test}}\left(h\right) = \frac{1}{n'}\sum_{i=n+1}^{n+n'}\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right)$$

- Squared loss

$$\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right) = \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Recall lab1

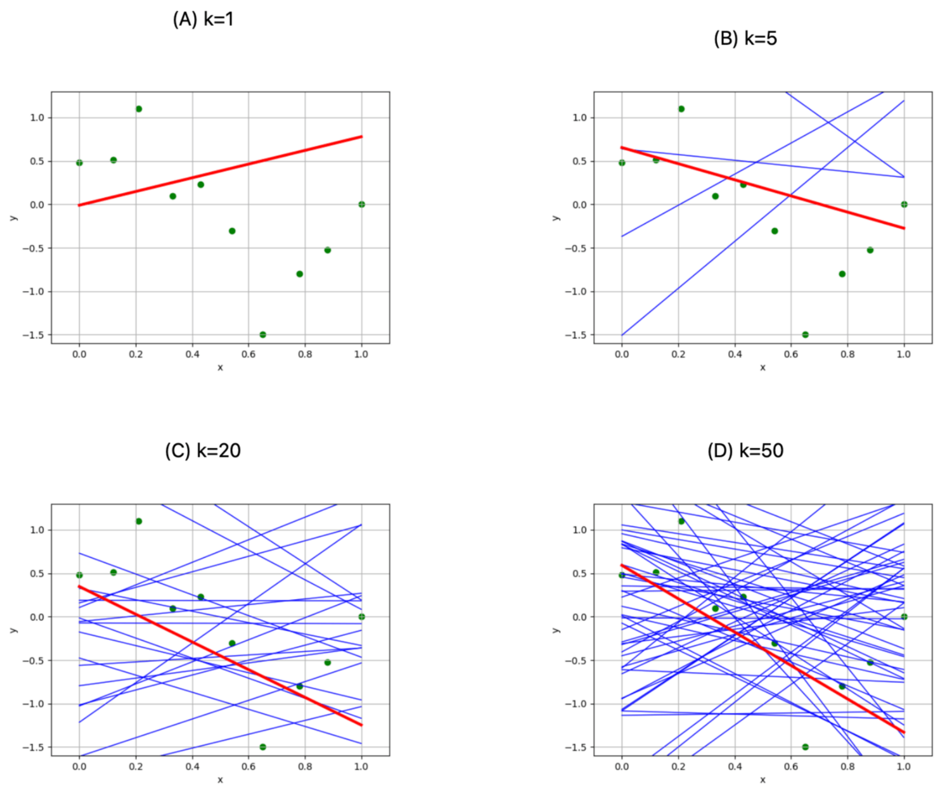
```
def random_regress(X, Y, k):
    n, d = X.shape

    # generate k random hypotheses
    ths = np.random.randn(d, k)
    th0s = np.random.randn(1, k)

    # compute the mean squared error of each hypothesis
on the data set
    errors = lin_reg_err(X, Y, ths, th0s)

    # Find the index of the hypotheses with the lowest
error
    i = np.argmin(errors)

    # return the theta and theta0 parameters that
define that hypothesis
    theta, theta0 = ths[:,i:i+1], th0s[:,i:i+1]
    return (theta, theta0), errors[i]
```



- Will this method eventually get arbitrarily close to the best solution? What do you think about the efficiency of this method?

Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validationa

Linear regression: the analytical way

- How about we just consider all hypotheses in our class and choose the one with lowest training error?
- We'll see: not typically straightforward
- But for linear regression with square loss: can do it!
- In fact, sometimes, just by plugging in an equation!

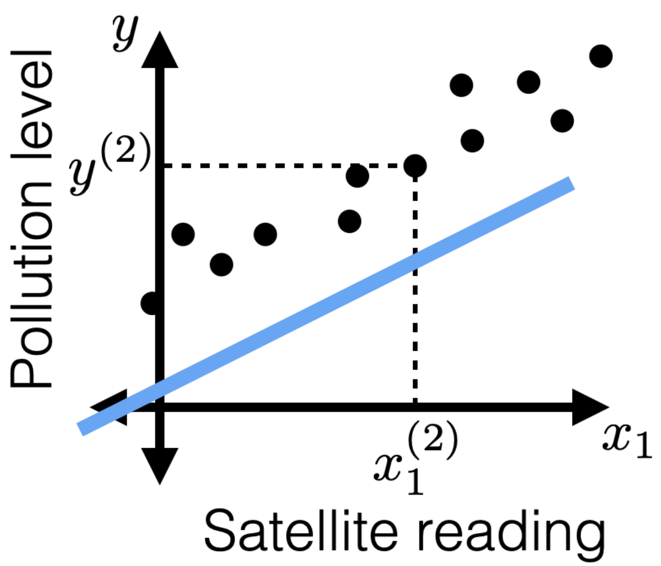
Don't want to deal with θ_0

$$h(x;\theta,\theta_0) = \theta^T x + \theta_0$$

$$= \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} + \theta_0$$

$$= \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d & \theta_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$

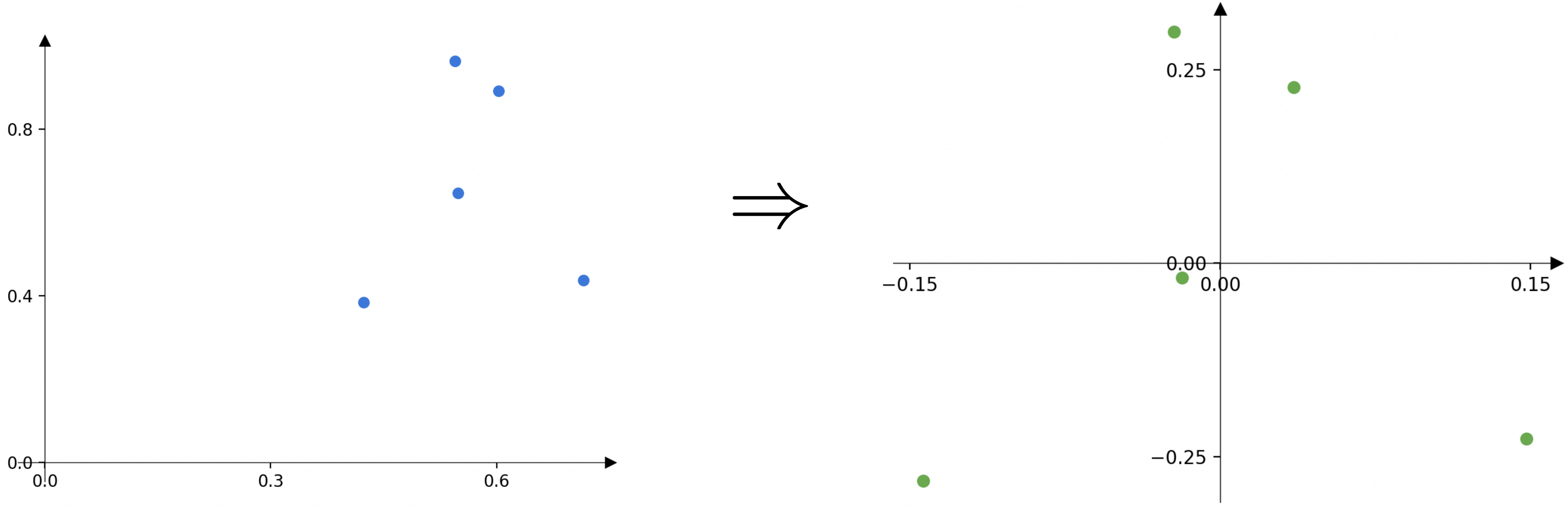
$$= \theta_{\text{aug}}^T x_{\text{aug}}$$



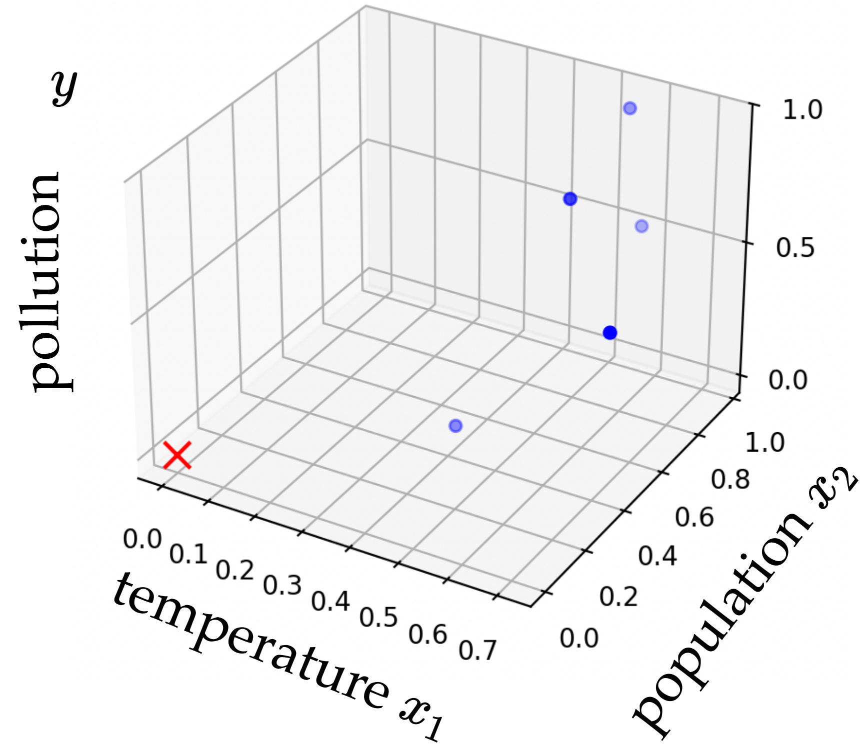
Append a "fake" feature of 1

Don't want to deal with θ_0

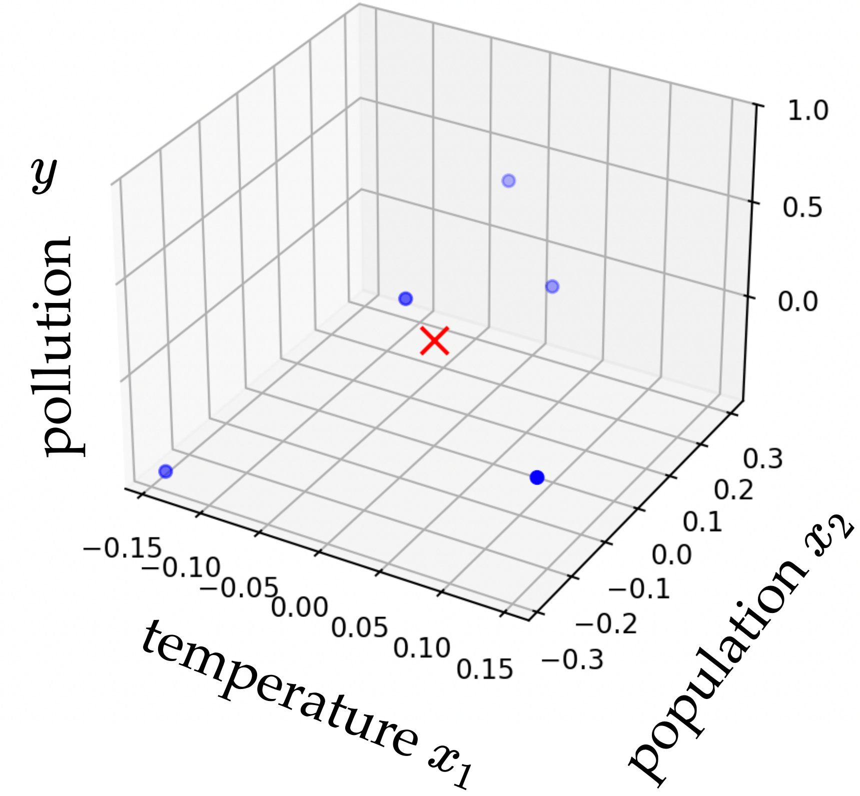
"center" the data



"center" the data



$\Rightarrow$



	Temperature	**Population**	**Pollution**
Chicago	90	45	7.2
New York	20	32	9.5
Boston	35	100	8.4

center the data $\Downarrow$

	Temperature	**Population**	**Pollution**
Chicago	41.66	-14	-1.66
New York	-28.33	-27	1.133
Boston	-13.33	41	0.033

	Temperature	**Population**	**Pollution**
Chicago	41.66	-14	-1.66
New York	-28.33	-27	1.133
Boston	-13.33	41	0.033

Assemble

$$X = \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Assemble

$$X = \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Now the training error:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left({x^{(i)}}^\top \theta - y^{(i)} \right)^2 = \frac{1}{n} (X\theta - Y)^\top (X\theta - Y)$$

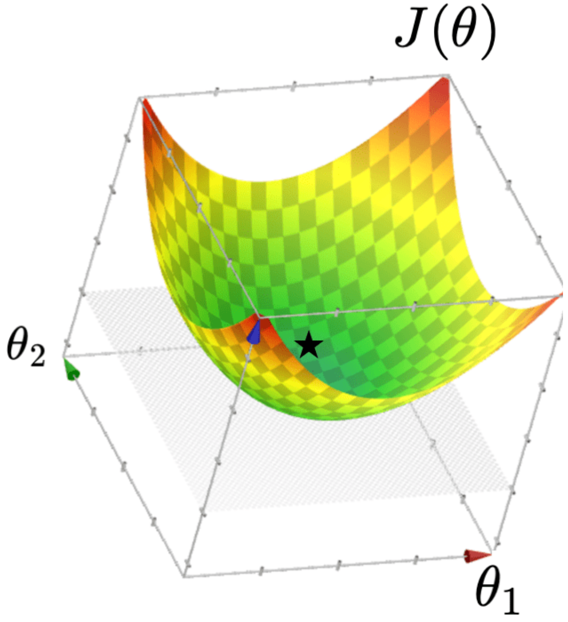
Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validationa

Objective function (training error)

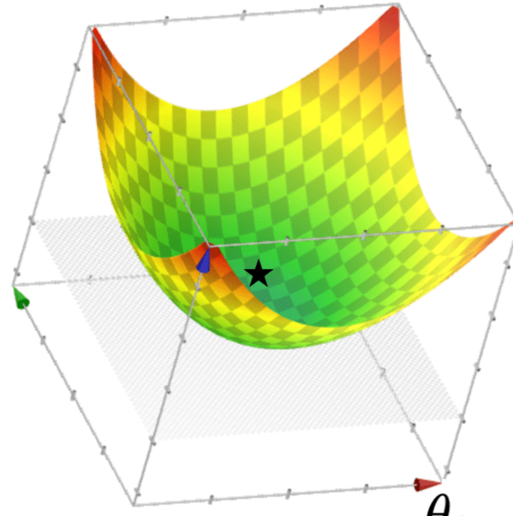
$$J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y)$$

- Goal: find θ to minimize $J(\theta)$
- Q: What kind of function is $J(\theta)$?
- A: Quadratic function
- Q: What does $J(\theta)$ look like?
- A: *Typically*, looks like a "bowl"



- Typically, $J(\theta) = \frac{1}{n}(X\theta - Y)^\top(X\theta - Y)$ "curves up" and is unique minimized at a point if gradient at that point is zero

$$\nabla_\theta J = \begin{bmatrix} \partial J / \partial \theta_1 \\ \vdots \\ \partial J / \partial \theta_d \end{bmatrix} = \frac{2}{n}\left(X^T X\theta - X^T Y\right)$$



Set the gradient $\nabla_\theta J \overset{\text{set}}{=} 0$

$$\Longrightarrow \quad \theta^* = \left(X^\top X\right)^{-1} X^\top Y$$

- When θ^* is well defined, it's indeed guaranteed to be the unique minimizer of $J(\theta)$

Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validationa

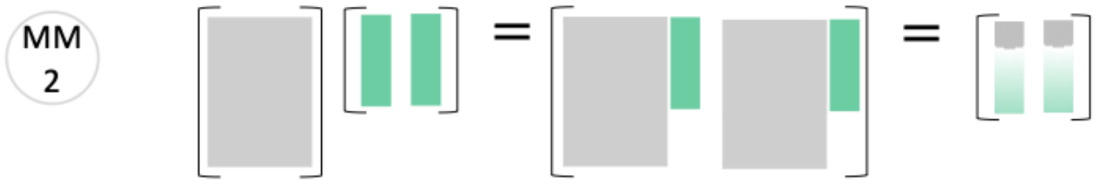
- $\theta^* = \left(X^\top X\right)^{-1} X^\top Y$ is only well-defined if $\left(X^\top X\right)$ is invertible

- and $\left(X^\top X\right)$ is invertible *if and only* if X is full column rank

So, we will be in trouble if X is not full column rank, which happens:

a. either when $n<d$, or

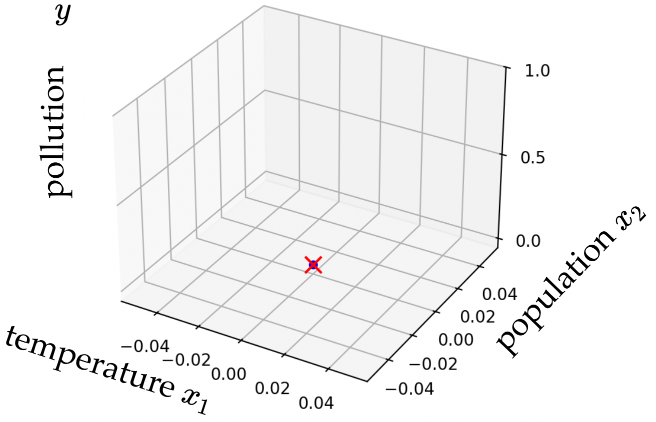
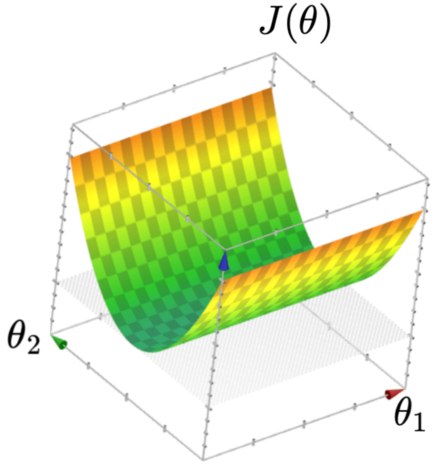
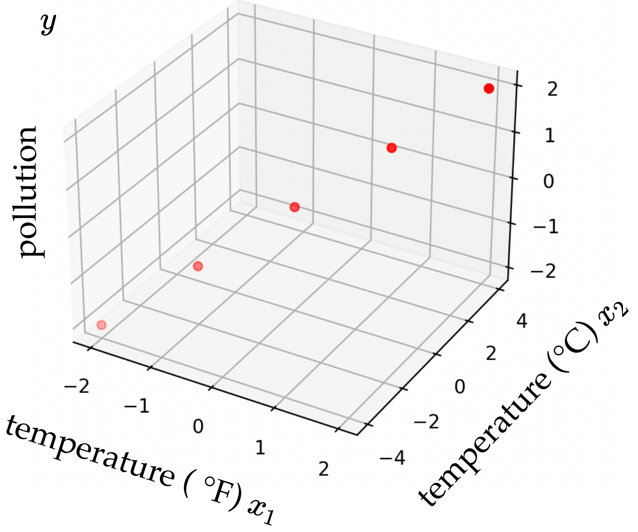
b. columns (features) in X have linear dependency



$\boldsymbol{Ax}$ and $\boldsymbol{Ay}$ are linear combinations of columns of A.

$$\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}\begin{bmatrix}x_1 & y_1\\x_2 & y_2\end{bmatrix} = A[\boldsymbol{x} \quad \boldsymbol{y}] = [A\boldsymbol{x} \quad A\boldsymbol{y}]$$

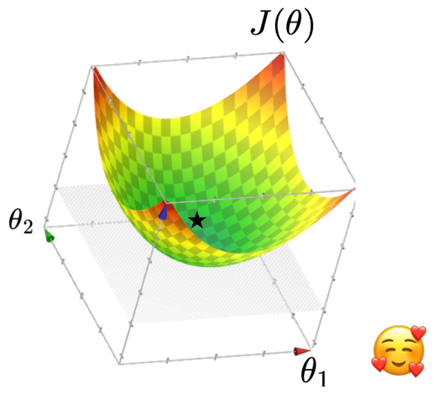
https://github.com/kenjihiranabe/The-Art-of-Linear-Algebra
https://www.3blue1brown.com/topics/linear-algebra

<table>
<tr><th>Case</th><th>Example</th><th>Objective Function Looks Like</th><th>Optimal Parameters</th></tr>
<tr><td>2a. less data than features</td><td>
</td><td rowspan="2">
</td><td rowspan="2">infinitely many optimal parameters (that define optimal hyperplanes)</td></tr>
<tr><td>2b. linearly dependent features</td><td>
</td></tr>
</table>

Quick Summary:

1. Typically, X is full column rank

- $J(\theta)$ looks like a bowl
- $\theta^* = \left(X^\top X\right)^{-1} X^\top Y$
- θ^* gives the unique optimal hyperplane

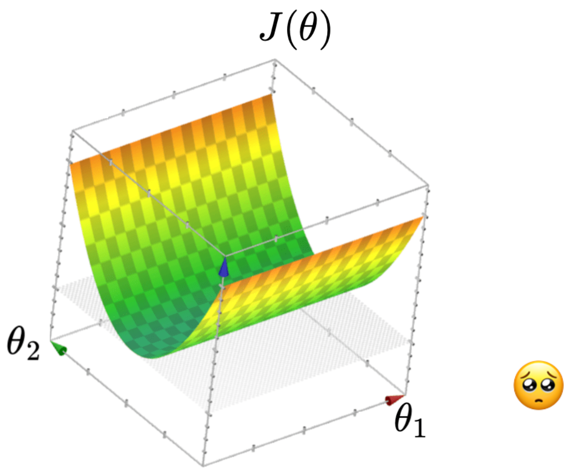


2. When X is not full column rank

a. either when $n<d$, or

b. columns (features) in X have linear dependency

- $J(\theta)$ looks like a half-pipe
- This 👈 formula is not well-defined
- Infinitely many optimal hyperplanes

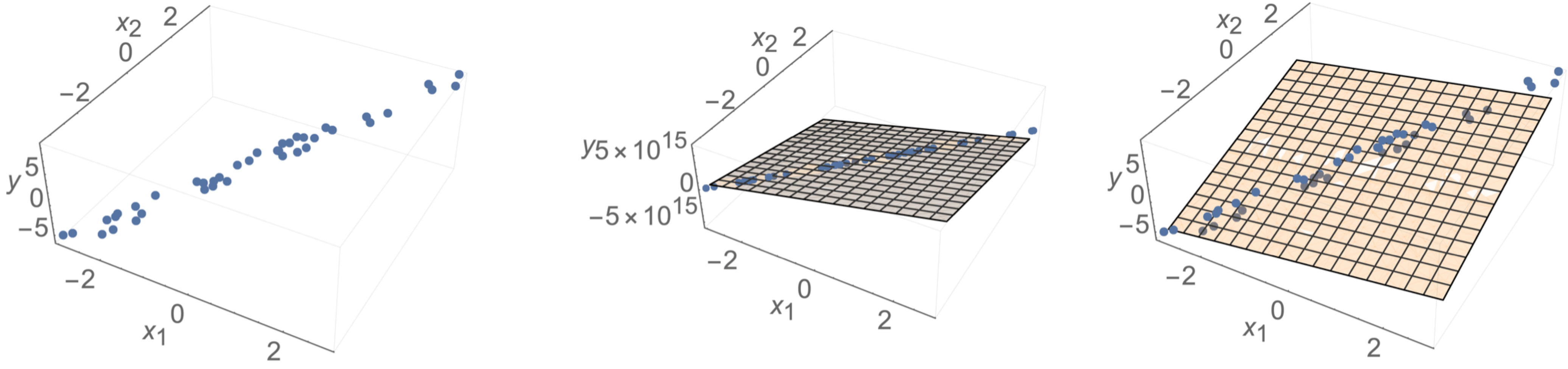


Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validation

- Sometimes, noise can resolve the invertibility issue

- but still lead to undesirable results



- How to choose among hyperplanes?

- Prefer θ with small magnitude (less sensitive prediction when x changes slightly)

Ridge Regression

- Add a square penalty on the magnitude
- $J_{\text{ridge}}(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y) + \lambda\|\theta\|^2$

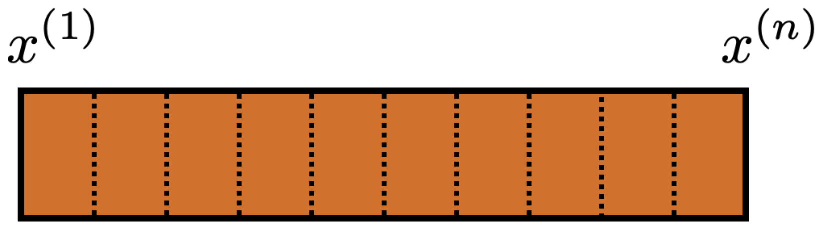
$(\lambda > 0)$

- λ is a so-called "hyperparameter"
- Setting $\nabla_\theta J_{\text{ridge}}(\theta) = 0$ we get $\theta^* = \left(X^\top X + n\lambda I\right)^{-1} X^\top Y$
- (θ^* (here) always exists, and is always the unique optimal parameters.)
- (see recitation/hw for discussion about the offset.)

Outline

- Recap: Supervised Learning Setup, Terminology
- Ordinary Least Square Regression
  - Problem Formulation
  - Closed-form Solution (when well-defined)
  - When closed-form solution is not well-defined
    - Mathematically, Practically, Visually
- Regularization and Ridge Regression
- Hyperparameter and Cross-validation

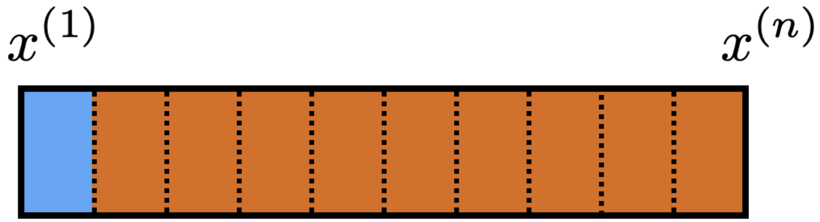
Cross-validation



Cross-validate($\mathcal{D}_n$, k)

  Divide $\mathcal{D}_n$ into k chunks $\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,k}$ (of roughly equal size)

Cross-validation

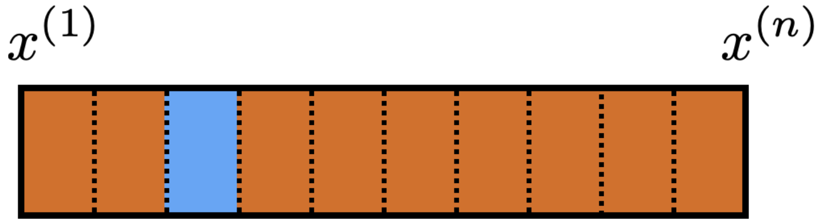


Cross-validate($\mathcal{D}_n$, k)

  Divide $\mathcal{D}_n$ into k chunks $\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,k}$ (of roughly equal size)

  for i = 1 to k

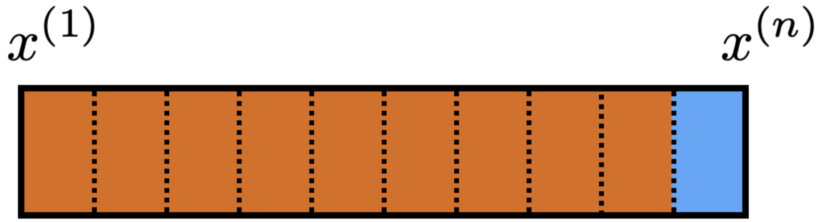
Cross-validation




```
Cross-validate( 𝒟ₙ , k )
  Divide 𝒟ₙ into k chunks 𝒟ₙ,₁,...,𝒟ₙ,ₖ (of
  roughly equal size)
  for i = 1 to k
```

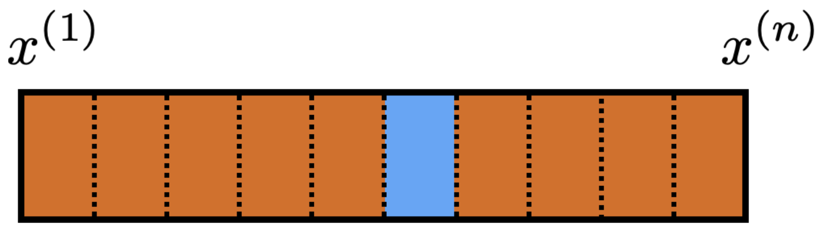
. . .

Cross-validation



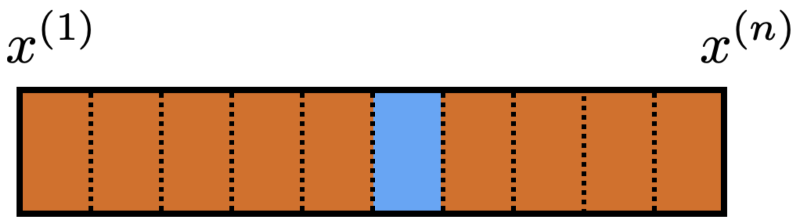

```
Cross-validate( 𝒟ₙ , k )
  Divide 𝒟ₙ into k chunks 𝒟ₙ,₁,...,𝒟ₙ,ₖ (of
  roughly equal size)
  for i = 1 to k
```

Cross-validation



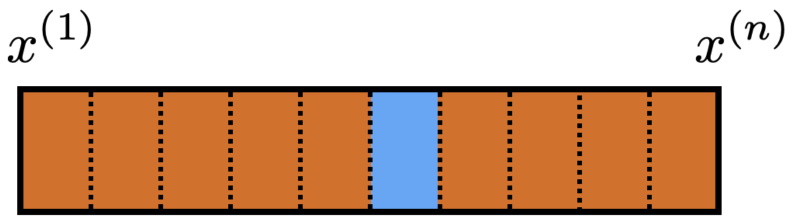

```
Cross-validate( 𝒟_n , k )
  Divide 𝒟_n into k chunks 𝒟_{n,1},...,𝒟_{n,k} (of
  roughly equal size)
  for i = 1 to k
    train h_i on 𝒟_n\𝒟_{n,i} (i.e. except chunk i)
```

Cross-validation




```
Cross-validate( 𝒟_n , k )
  Divide 𝒟_n into k chunks 𝒟_{n,1}, ..., 𝒟_{n,k} (of
  roughly equal size)
  for i = 1 to k
    train h_i on 𝒟_n \ 𝒟_{n,i} (i.e. except chunk i)
    compute "test" error ℰ(h_i, 𝒟_{n,i}) of h_i on 𝒟_{n,i}
```

Cross-validation



```
Cross-validate(𝒟_n , k )
  Divide 𝒟_n into k chunks 𝒟_{n,1},...,𝒟_{n,k} (of
  roughly equal size)
  for i = 1 to k
    train h_i on 𝒟_n\𝒟_{n,i} (i.e. except chunk i)
    compute "test" error ℰ(h_i, 𝒟_{n,i}) of h_i on 𝒟_{n,i}
  Return (1/k) Σ_{i=1}^{k} ℰ(h_i, 𝒟_{n,i})
```

Cross-validate($\mathcal{D}_n$, k)

Divide $\mathcal{D}_n$ into k chunks $\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,k}$ (of roughly equal size)

for i = 1 to k

train h_i on $\mathcal{D}_n \backslash \mathcal{D}_{n,i}$ (i.e. except chunk i)

compute "test" error $\mathcal{E}(h_i, \mathcal{D}_{n,i})$ of h_i on $\mathcal{D}_{n,i}$

Return $\frac{1}{k}\sum_{i=1}^{k}\mathcal{E}(h_i, \mathcal{D}_{n,i})$

Comments on (cross)-validation

- good idea to shuffle data first
- a way to "reuse" data
- it's not to evaluate a hypothesis
- rather, it's to evaluate learning algorithm (e.g. hypothesis class choice, hyperparameters)
- Could e.g. have an outer loop for picking good hyperparameter or hypothesis class

https://docs.google.com/forms/d/e/1FAIpQLSftMB5hSccgAbIAFmP_LuZt95w6KFx0x_R3uuzBP8WwjSzZeQ/viewform?embedded=true

We'd love to hear your thoughts.

Thanks!