https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 2: Linear regression and regularization

Shen Shen

Feb 9, 2024

(many slides adapted from Tamara Broderick)

Instructors

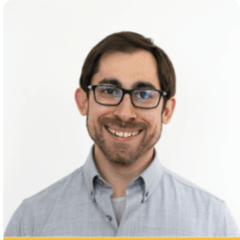

Priya Donti


Kyle Keane


Manolis Kellis


Alexandre Megretski


Vince Monardo


Chris Tanner


Shen Shen

Course Assistant


Taylor Braun

Logistical issues? Personal concerns? We'd love to help out at **6.390-personal@mit.edu**

TAs


George Bian


Linh Nguyen


Andrew Hutchison


Emily Liu


Emily Jiang


Haley Nakamura


Elisa Xia


Yogi Sragow

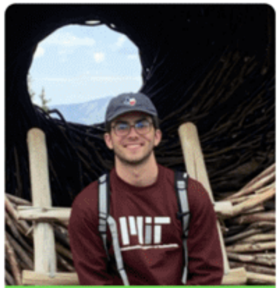

Abhay Basireddy


Shaunticlair Ruiz


Kevin Bunn


Shaden Alshammari


Claire Lu


Andi Spiride


Lucian Covarrubias

and ~40 awesome LAs

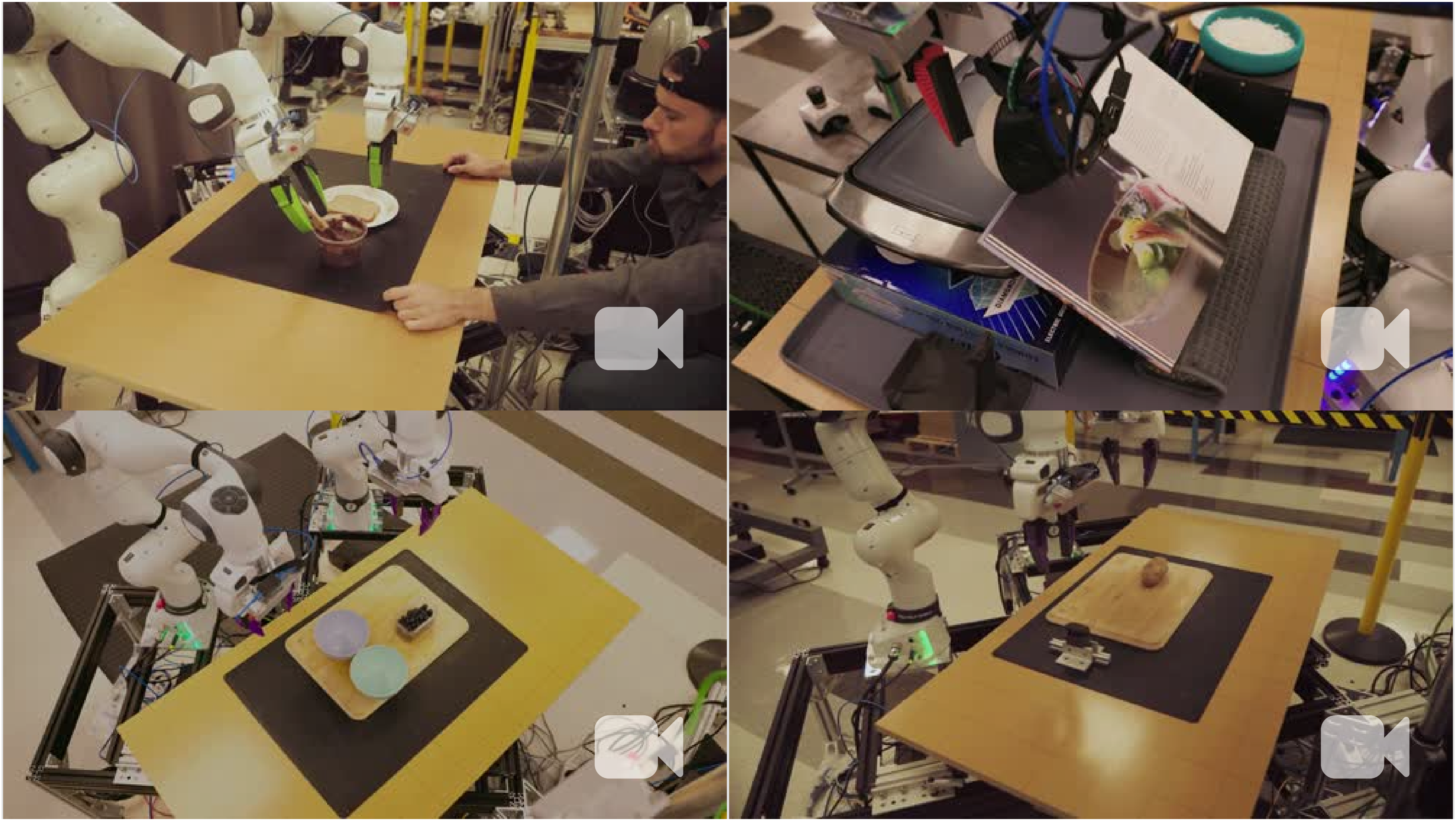
Logistics

- 11am Section 3 and 4 are completely full and we have many requests to switch. Physical space packed.
- If at all possible, please help us by signup/switch to other slots.

- OHs start this Sunday, please also join our Piazza
- Thanks for all the assignments feedback. We are adapting on-the-go but these certainly benefit future semesters.
- Start to get assignments due now. (first up, exercises 2, keep an eye on the "due")

https://shenshen.mit.edu/demos/gifs/atlas_darpa_overall.gif

Optimization + first-principle physics



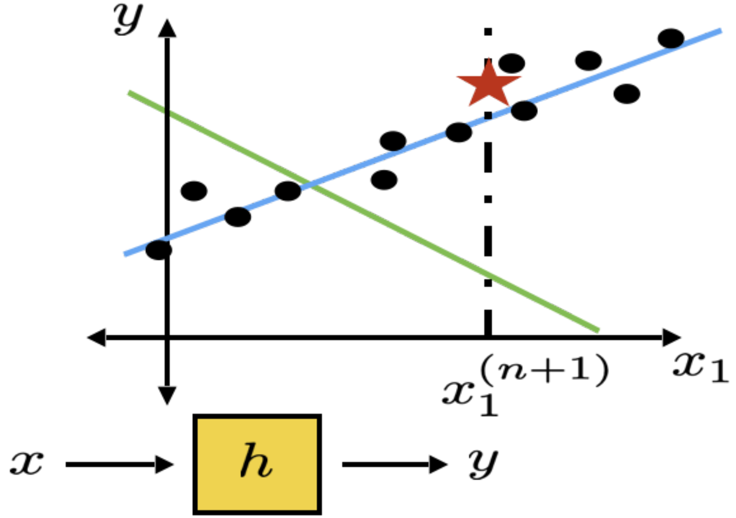
https://www.youtube.com/embed/fn3KWM1kuAw?enablejsapi=1

Outline

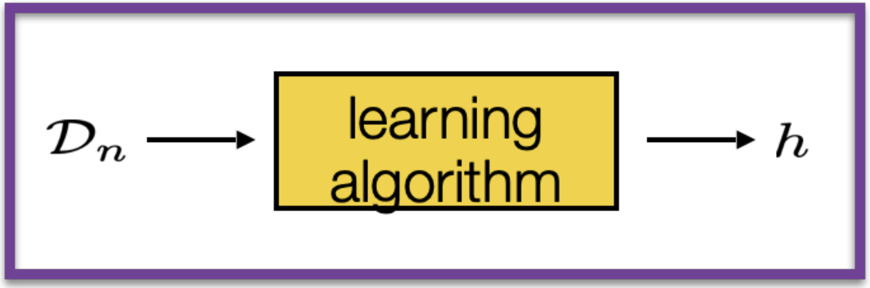
- Recap of last (content) week.
- Ordinary least-square regression
  - Analytical solution (when exists)
  - Cases when analytical solutions don't exist
    - Practically, visually, mathamtically
- Regularization
- Hyperparameter, cross-validation

How do we learn?

- Have data; have hypothesis class
- Want to choose (learn) a good hypothesis h (or more concretely, a set of parameters)



How to get it:
(Next time!)

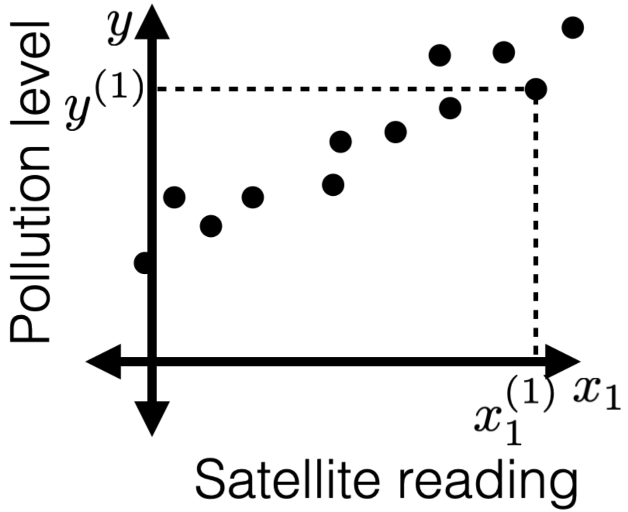


Example: predict pollution level

(Training) data

- n training data points
- For data point $i \in \{1, \ldots, n\}$
  - Feature vector $x^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})^\top \in \mathbb{R}^d$
  - Label $y^{(i)} \in \mathbb{R}$

- Training data $\mathcal{D}_n = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$



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

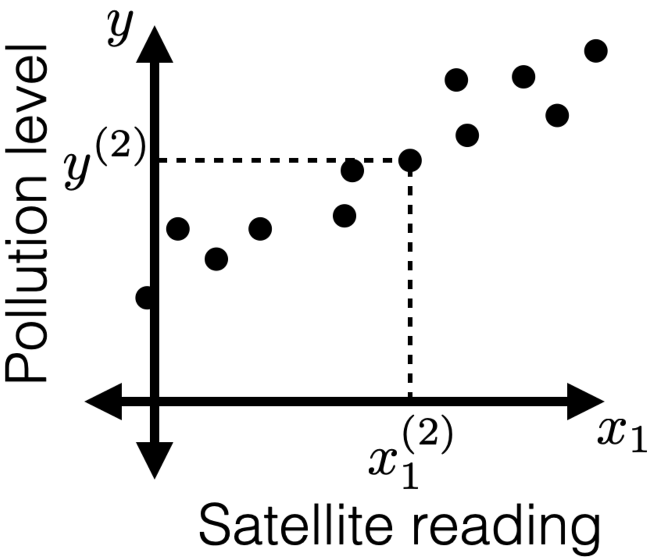
How to label? Hypothesis $h : \mathbb{R}^d \to \mathbb{R}$

$x \to \boxed{h} \to y$

Is this a ***good*** hypothesis?

- Example *h*: For any *x*, *h*(*x*) = 1,000,000

Linear regressors

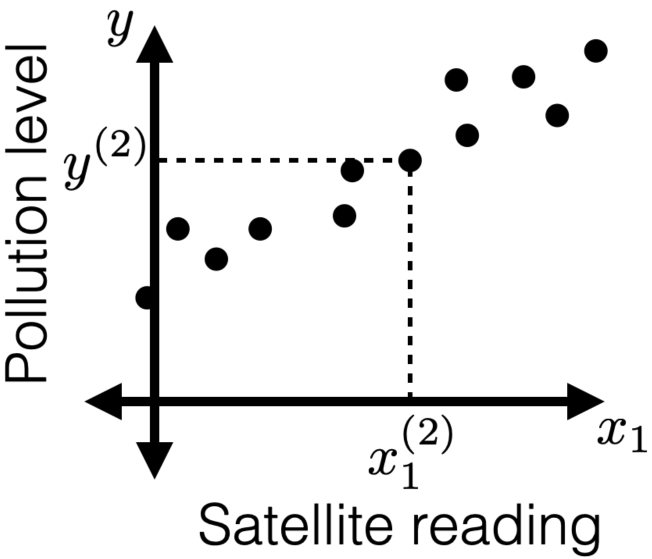


Linear regressors

- Hypothesis class $\mathcal{H}$: set of h

- A linear regression hypothesis when *d*=1:

$$h(x) = \theta x + \theta_0$$

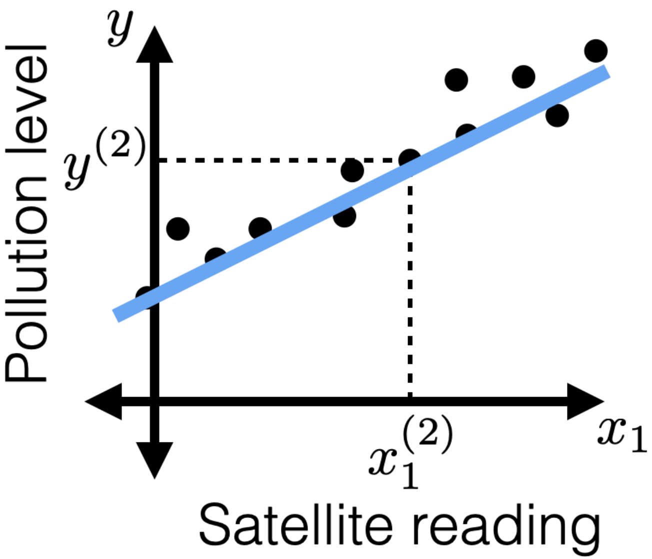


Linear regressors

- Hypothesis class $\mathcal{H}$: set of h

- A linear regression hypothesis when *d*=1:

$$h(x) = \theta x + \theta_0$$



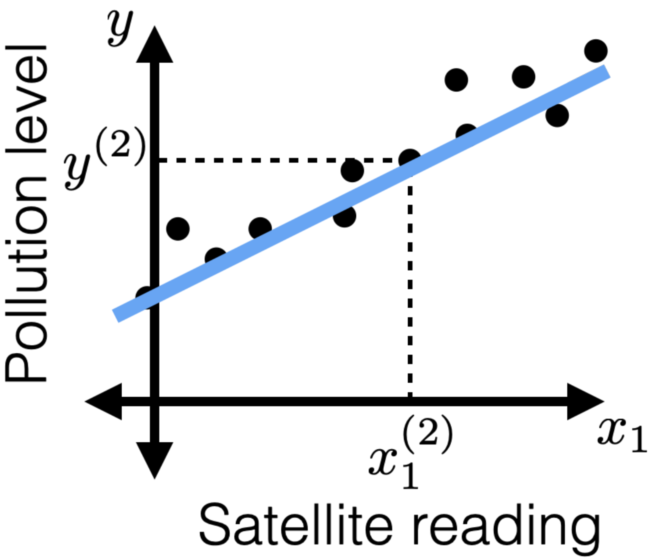
Linear regressors

- Hypothesis class $\mathcal{H}$: set of h

- A linear regression hypothesis when *d*=1:

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

parameters



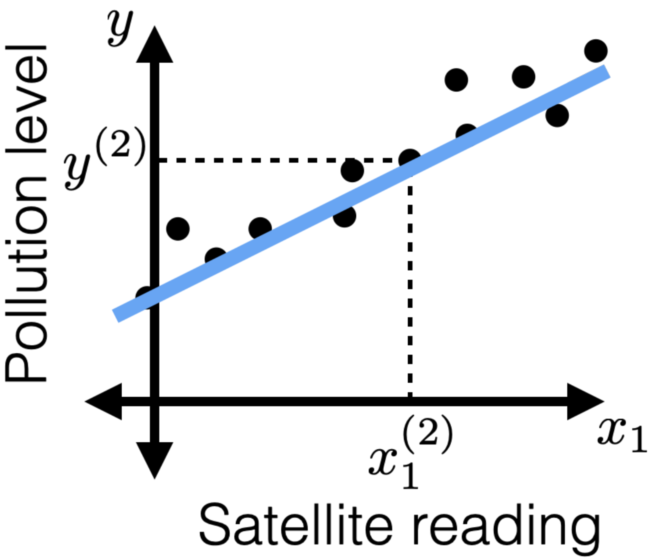
Linear regressors

- Hypothesis class $\mathcal{H}$: set of h

- A linear regression hypothesis when d=1:

  parameters

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



- A linear reg. hypothesis when d≥1:

$$\begin{aligned} h(x; \theta, \theta_0) &= \theta_1 x_1 + \cdots + \theta_d x_d + \theta_0 \\ &= \theta^\top x + \theta_0 \end{aligned}$$

OR

$$\begin{aligned} h(x) &= \theta_1 x_1 + \cdots + \theta_d x_d + (\theta_0)(1) \\ &= \theta^\top x \end{aligned}$$

Linear regressors

- A linear reg. hypothesis when $d \geq 1$:

$$h(x;\theta,\theta_0) = \theta_1 x_1 + \cdots + \theta_d x_d + \theta_0$$
$$= \theta^\top x + \theta_0$$

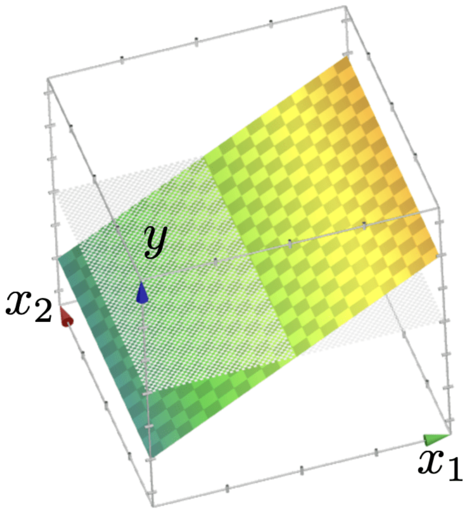
1x2,2x1

OR

$$h(x;\theta) = \theta_1 x_1 + \cdots + \theta_d x_d + (\theta_0)(1)$$
$$= \theta^\top x$$

1x3,3x1

Notational trick: *not* the same θ & x!

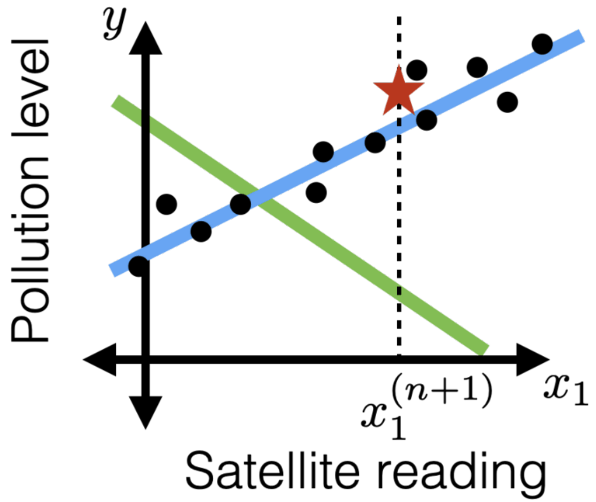


- Our hypothesis class in linear regression will be the set of all such *h*

Hypothesis is a "hyperplane"

How good is a regression hypothesis?

- Should predict well on future data
- How good is a regressor at one point? Loss $L(g, a)$ g: guess, a: actual
  - Ex: squared loss
    $$L(g, a) = (g - a)^2$$
  - Example: asymmetric loss
    $$L(g, a) = \begin{cases} (g-a)^2 \text{ if } g > a \\ 2(g-a)^2 \text{ if } g \leq a \end{cases}$$

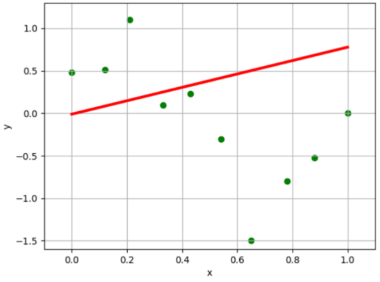


- Test error (n' new points): $\mathcal{E}(h) = \frac{1}{n'} \sum_{i=n+1}^{n+n'} L(h(x^{(i)}), y^{(i)})$
- Training error: $\mathcal{E}_n(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$
- One idea: prefer h to $\tilde{h}$ if $\mathcal{E}_n(h) < \mathcal{E}_n(\tilde{h})$

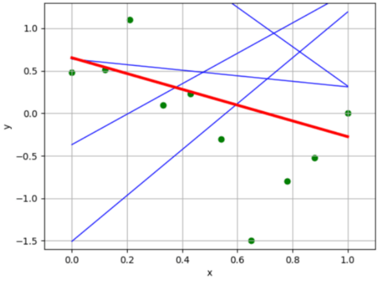
1.3)

Now, here are some executions for different values of k (shown in red is the hypothesis with the lowest MSE, among the k tested).

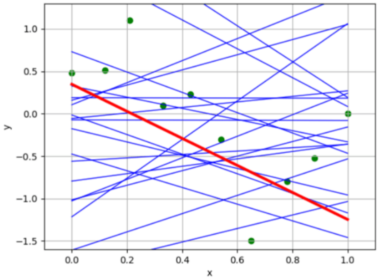
(A) k=1



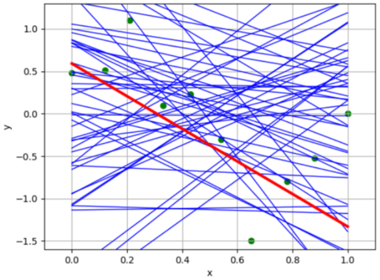
(B) k=5



(C) k=20



(D) k=50



- What happens as we increase `k`? Compare the four “best” linear regressors found by the random regression algorithm with different values of `k` chosen, which one does your group think is "best of the best"?
- How does it match your initial guess about the best hypothesis?
- Will this method eventually get arbitrarily close to the best solution? What do you think about the efficiency of this method?

Outline

- Recap of last (content) week.
- Ordinary least-square regression
    - Analytical solution (when exists)
    - Cases when analytical solutions don't exist
        - Practically, visually, mathemtically
- Regularization
- Hyper-parameter, cross-validation

Linear regression: Another 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!

Linear regression: Another 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!

- Recall: training error: $\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$
- Training error: square loss, linear regr., extra “1” feature

$$\frac{1}{n}\sum_{i=1}^{n}(h(x^{(i)}) - y^{(i)})^2$$

Linear regression: Another 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!

- Recall: training error: $\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$

- Training error: square loss, linear regr., extra "1" feature

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

Linear regression: Another way

- Training error: square loss, linear regr., extra “1” feature

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

Define

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

nxd

nx1

Linear regression: Another way

- Training error: square loss, linear regr., extra “1” feature

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

Define

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

nxd

nx1

Linear regression: A Direct Solution

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

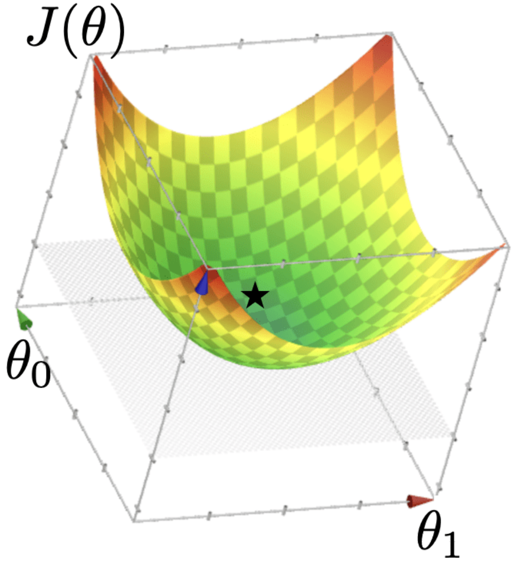
- Q: what kind of function is $J(\theta)$

- Q: how does $J(\theta)$ look like?

- A: $J(\theta)$ quadratic function;

  typically look like a "bowl" (but

  there're exceptions)



Linear regression: A Direct Solution

- Goal: minimize $J(\theta) = \frac{1}{n}(\tilde{X}\theta - \tilde{Y})^\top(\tilde{X}\theta - \tilde{Y})$
- Uniquely minimized at a point if gradient at that point is zero and function "curves up" [see linear algebra]
- Gradient $\nabla_\theta J(\theta) \overset{\text{set}}{=} 0$

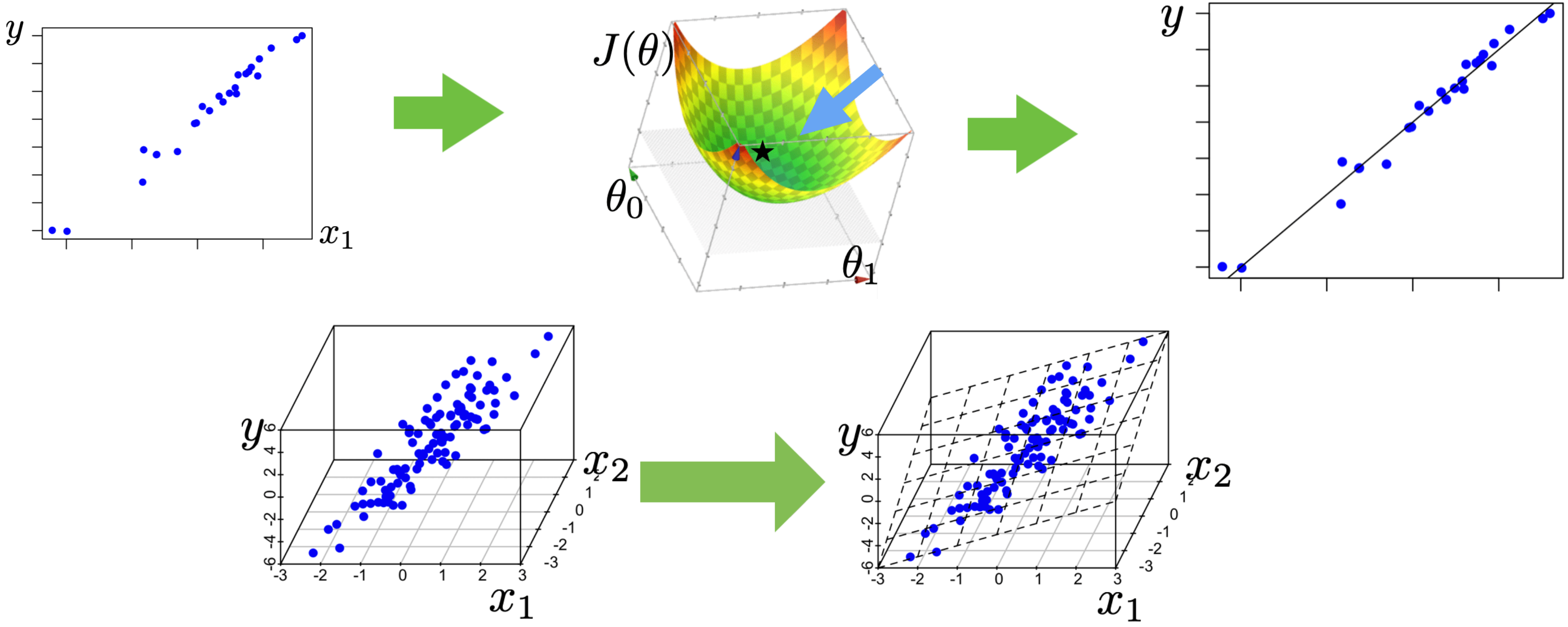
dx1

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

Comments about $\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$

- When θ^* exists, guaranteed to be unique minimizer of

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

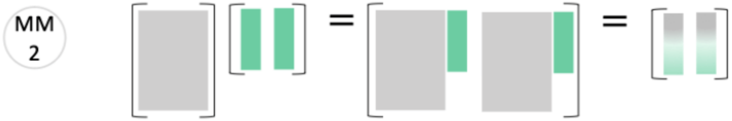


Now, the catch: $\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$ may not be well-defined

- $\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$ is not well-defined if $\left(\tilde{X}^\top \tilde{X}\right)$ is not invertible

- Indeed, it's possible that $\left(\tilde{X}^\top \tilde{X}\right)$ is not invertible.

- In particular, $\left(\tilde{X}^\top \tilde{X}\right)$ is not invertible if and only if $\tilde{X}$ is not full column rank



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

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

Now, the catch: $\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$ is not well-defined

if $\tilde{X}$ is not full column rank

Recall

indeed $\tilde{X}$ is not full column rank

$$\underset{\text{nxd}}{\tilde{X}} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$

1. if $n<d$
2. if columns (features) in $\tilde{X}$ have linear dependency

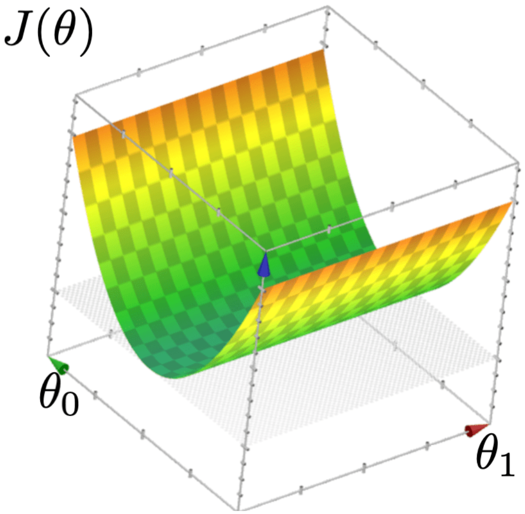
Recap:

$$\underset{\text{nxd}}{\tilde{X}} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}$$

1. if $n<d$ (i.e. not enough data)
2. if columns (features) in $\tilde{X}$ have linear dependency (i.e., so-called co-linearity)

$$\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$$ is not defined

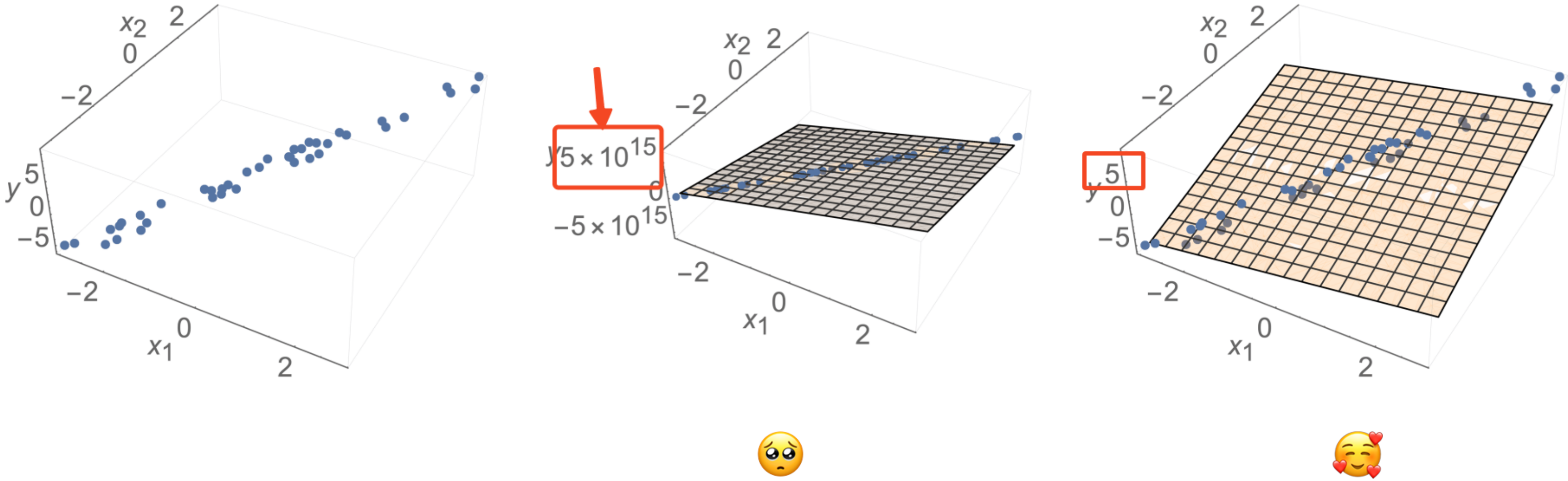
- Both cases do happen in practice
- In both cases, loss function is a "half-pipe"
- In both cases, infinitily-many optimal hypotheses
- Side-note: sometimes noise can resolve invertabiliy issue, but undesirable



Outline

- Recap of last (content) week.
- Ordinary least-square regression
    - Analytical solution (when exists)
    - Cases when analytical solutions don't exist
        - Practically, visually, mathemtically
- Regularization
- Hyper-parameter, cross-validation

Regularization



- How to choose among hyperplanes? Preference for θ components being near zero

Ridge Regression Regularization

- Linear regression with square penalty: ridge regression

$$J_{\text{ridge}}(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} (\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2 + \lambda \|\theta\|^2$$

Ridge Regression Regularization

- Linear regression with square penalty: ridge regression

$$J_{\text{ridge}}(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2 + \lambda\|\theta\|^2$$

Ridge Regression Regularization

- Linear regression with square penalty: ridge regression

$$J_{\text{ridge}}(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2 + \boxed{\lambda\|\theta\|^2} \qquad \boxed{(\lambda > 0)}$$

- Special case: ridge regression with no offset

$$J_{\text{ridge}}(\theta) = \frac{1}{n}(\tilde{X}\theta - \tilde{Y})^\top(\tilde{X}\theta - \tilde{Y}) + \lambda\|\theta\|^2$$

- Min at: $\nabla_\theta J_{\text{ridge}}(\theta) = 0$

$$\Rightarrow \theta = (\underbrace{\tilde{X}^\top \tilde{X}}_{\text{dxn,nxd}} + n\lambda \underbrace{\boxed{I}}_{\text{dxd}})^{-1}\tilde{X}^\top \tilde{Y}$$

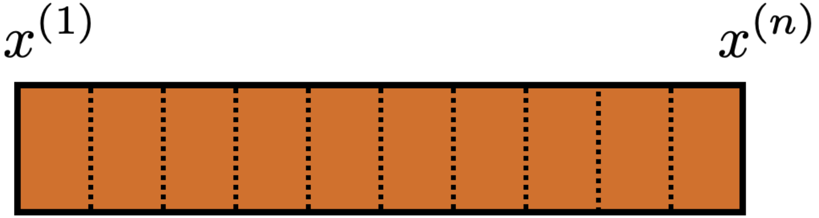
- When $\lambda > 0$, always “curves up” & can invert
- Can also solve with an offset

λ is a hyper-parameter

Outline

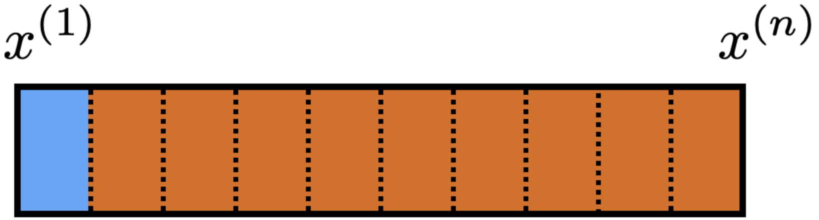
- Recap of last (content) week.
- Ordinary least-square regression
  - Analytical solution (when exists)
  - Cases when analytical solutions don't exist
    - Practically, visually, mathemtically
- Regularization
- Hyper-parameter, cross-validation

Cross-validation



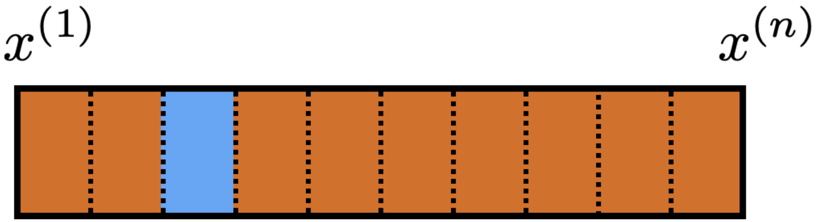

```
Cross-validate( 𝒟ₙ , k )
  Divide 𝒟ₙ into k chunks 𝒟ₙ,₁,...,𝒟ₙ,ₖ (of
  roughly equal size)
```

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
```

Cross-validation



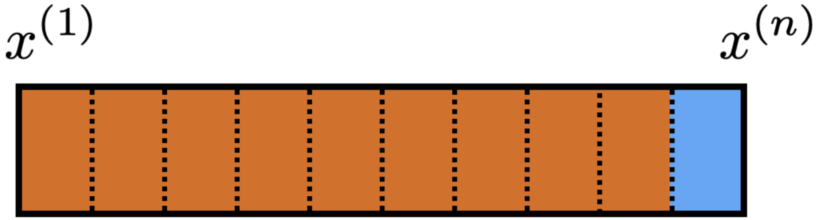
Cross-validate($\mathcal{D}_n$, k)

  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

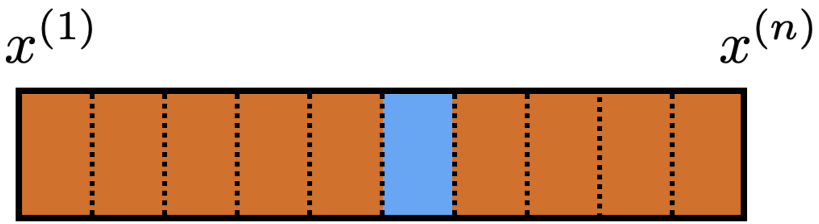


Cross-validate($\mathcal{D}_n$, k)

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

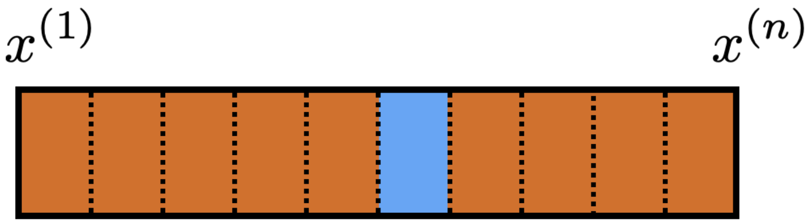
  for i = 1 to k

Cross-validation




```
Cross-validate( 𝒟ₙ , k )
  Divide 𝒟ₙ into k chunks 𝒟ₙ,₁,...,𝒟ₙ,ₖ (of
  roughly equal size)
  for i = 1 to k
    train hᵢ on 𝒟ₙ\𝒟ₙ,ᵢ (i.e. except chunk i)
```

Cross-validation



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

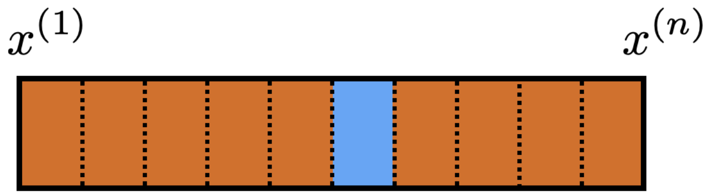
  Divide $\mathcal{D}_n$ into k chunks $\mathcal{D}_{n,1}, \ldots, \mathcal{D}_{n,k}$ (of roughly equal size)

  for i = 1 to k

    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 $\mathcal{D}_{n,i}$

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)

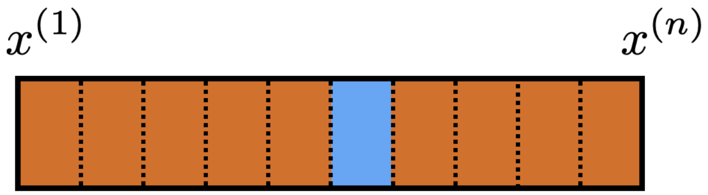
for i = 1 to k

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

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

    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 about cross-validation

- good idea to shuffle data first
- a way to "reuse" data
- not evaluating a hypothesis, but rather
- evaluating learning algorithm. (e.g. hypothesis class, hyper-parameter)
- Could e.g. have an outer loop for picking good hyper-parameter/class

Summary

- One strategy for finding ML algorithms is to reduce the ML problem to an optimization problem.
- For the ordinary least squares (OLS), we can find the optimizer analytically, using basic calculus! Take the gradient and set it to zero. (Generally need more than gradient info; suffices in OLS)
- Two ways to approach the calculus problem: write out in terms of explicit sums or keep in vector-matrix form. Vector-matrix form is easier to manage as things get complicated (and they will!) There are some good discussions in the lecture notes.

Summary

- What does it mean to well posed.
- When there are many possible solutions, we need to indicate our preference somehow.
- Regularization is a way to construct a new optimization problem
- Least-squares regularization leads to the ridge-regression formulation. Good news: we can still solve it analytically!
- Hyper-parameters and how to pick them. Cross-validation

We'd love it for you to share some lecture feedback.

Thanks!