Introduction

CMPUT 367: Intermediate Machine Learning

A Second Course in ML

We get to build on an ML foundation and move to more advanced modeling

Reminder about the Basics

- Focused on understanding
  - optimization concepts
  - uncertainty quantification, using the language of probability
  - how to formalize learning problems and estimate parameters
  - reasoning about generalization capabilities (bias-variance, overfitting)
- Focused on linear models and prediction
- Touched briefly on nonlinear models using polynomial regression

Two Types of Uncertainty

- **Uncertainty in our prediction**: $Y \mid x$ is a distribution, returning $\hat{y} = f(x)$ will not perfectly match the observed y
  - due to ***partial observability*** (e.g., predicting house price using only the age of the house, missing many important input variables like size etc).
  - sometimes called aleatoric uncertainty

- **Uncertainty in our estimate**: we estimate f (or parameters $\mathbf{w}$) from data; we have a distribution $p(\mathbf{w} \mid \mathscr{D})$ that shrinks with more data
  - we reasoned about the confidence in our estimator and about how many samples we need (this is the particular focus of ***Bayesian methods***)
  - sometimes called epistemic uncertainty

Intermediate ML

- **Summary**: you know about the key concepts of generalization and uncertainty, as well as optimization approaches
  - These concepts are central and do not change when moving to more advanced models
- We can now focus on **understanding more advanced models**
  - move from simple, linear models to nonlinear, high-dimensional models
  - focus more on both prediction models and generative models
  - understand the key concepts in data (re)representation, that enables us to extract powerful models that generalize well
  - more on generalization theory, which is more interesting for this larger class of models (in higher-dimensions)

Course structure

- First few chapters revisit concepts from Basics of ML, but now for slightly more complex settings
  - Understanding sensitivity of linear regression, using matrix analysis
  - Learning $p(y \mid x)$ for more general distributions (exponential family)
  - Hessians for second-order gradient descent and constrained optimization
  - Improved approaches to evaluate generalization error (cross-validation)
- Then we move to the primary new topic: Data Representations
  - Explain the goals of data representations
  - Discuss prototype-based representations, latent variable models (PCA) and neural networks
  - Show how our methods (predictions, generative models, Bayesian approaches) extend to use these nonlinear transformations
- Introduction to handling missing data (bonus topic)

Course topics

Revisiting concepts

1. Intermediate Probability (ch. 2) (i.e., a few more prob concepts)
2. Revisiting Linear Regression (with matrices) (ch. 3)
3. Intermediate Optimization (ch. 4) (mainly Hessians and momentum)
4. Generalized Linear Models (ch. 5) (to learn $p(y \mid x)$)
5. Constrained Optimization (ch. 6)
6. Evaluating Generalization Performance (ch. 7)

Course topics #2

Data representations

7. Fixed Representations (ch. 8)
8. Learned Representations (ch. 9) (neural networks and latent variables)

Course topics #2

Generative Models

9. Mixture models and Expectation-maximization (ch. 10)
10. Generative Models and Data Representations (ch. 11)

Course topics #2

Theory Basics

11. Generalization Theory Basics (ch. 12)
12. Convergence Rates (for SGD) (ch. 13)

Course topics #2

Advanced Topics

13. Dealing with Missing Data (ch. 14)
14. More Advanced Bayesian Approaches (ch. 15)

Course Admin

- Very similar to how Basics of ML (CMPUT 267) is structured
  - Consistency is good
- We need to go over this briefly anyway, and you'll simply have to hear a similar description you may have heard before

Course essentials

- **Course information:** https://marthawhite.github.io/ml-intermediate/
  - Schedule and readings
- **Access-controlled course information:** eClass
  - Getting Started and FAQ (please visit this today!)
  - Video recordings, links to lecture meetings and assignment submission

Hybrid Teaching

- **Lectures will be in-person in a classroom AND on Zoom**
- In class, I will project my screen. My screen will be shared in Zoom too.
- I will monitor Zoom questions.
- The lectures will be recorded and posted right after class.

Course essentials

- **Course information:** https://marthawhite.github.io/ml-intermediate/
  - Schedule and readings
- **Access-controlled course information:** eClass
  - Getting Started and FAQ (please visit this today!)
  - Video recordings, links to lecture meetings and assignment submission
- **Lectures:** Tues. Thurs 3:30-4:50pm in CAB 243 and on Zoom
  - Lectures will be recorded and posted on eClass
- **Office hours:** Tuesday noon - 1 pm (on Zoom and in-person in ATH 3-05)

Teaching Assistants

Samuel Neumann

Haseeb Shah

Hugo Luis Silva

- **Office hours:** see eClass for times and locations+Zoom links
  - Typically question/answer sessions
  - In a classroom, to allow space to bring laptops, etc.
- **No TA office hours this week**
- There is no lab, you can ask coding questions during office hours

Readings

- Readings are from the Intermediate ML textbook
  - Available on course site and written by myself
  - **Disclaimer: These notes are still quite new**
  - I changed them a lot based on reactions from last year
- See the schedule for sections and for reading deadlines
- Readings have an associated marked component called Thought Questions

General Disclaimer

- **This course is still relatively new and not yet fully polished**
  - It is a great opportunity to teach ML to a group of students that have taken the Basics of ML
  - It is a bit like teaching a graduate class
- The notes and assignments are relatively new
- There will be some adjusting as we go and mistakes
- If this is going to make you really frustrated, then you should talk to me

Lectures

- Lectures will mostly involve me writing on my iPad (like a whiteboard)
- I highly encourage you to ask any question
    - You can raise your hand and then ask outloud
    - You can type questions in Zoom chat
    - We will use Discord for any questions you think of outside of class, that I will address in class
- We will have small (exercise) breaks in class
    - sometimes I'll give you a small derivation or exercise
- I will post my written notes afterwards (and videos will be published)

Course Discussion

- We have create a **Discord group; please sign up!**
- I want to generate as much class discussion as possible
- Please go there first to ask questions
- Please answer your classmates questions!
  - We'll step in if there is misinformation, but in many cases you can all help each other faster than we can get to the question
  - Peer discussions can very beneficial
- **Details in FAQ and Getting Started linked on eClass**

Prerequisites

- CMPUT 267 or CMPUT 296 (Basics of ML)
  - or my permission based on your other background
- Linear algebra and Algorithms course
- Motivation to learn and think **beyond the material**
  - This is what thought questions are meant to practice
- A desire to understand the mathematical underpinnings of ML

Using Julia instead of Python

- Many of you have used Julia at this point
  - Some of you might have even liked it!
  - Others were likely ambivalent
  - And there is probably a subgroup of those that actively dislike Julia
- For those that have not used Julia, we have tutorials linked in Getting Started
- Regardless of whether you like or don’t like it, it’s just a language
  - We provide you lots of guidance in the notebooks

But, but, don't I need Python to be useful in ML in the real world?

- No.
- Languages come and go. The foundational knowledge and ability to translate that into implementation is the key skill
- In fact, one constant in ML is how much you switch between languages, so you may as well practice now

Grading

- 30%: Assignments
  - Mixture of mathematical problems and programming exercises
- 5%: Quiz on **October 6**
- 20%: Midterm exam on **November 17**
- 35%: Final exam **December 12**
- 10%: Thought questions

Assignments

- Four assignments
- Coarse binned grading:
  - 80 - 100 → 100
  - 60 - 80 → 80
  - 40 - 60 → 60
  - **0 - 40** → **0**

Three exams

- Giving **clear** answers to short answer questions is a **skill**
  - It takes practice!
  - An important skill in CS is clear written communication.
- Practice questions will be available
- Exams will be in-person
- For all exams you are allowed a **two page cheat-sheet**
  - One page, front-and-back
  - No collaborating on cheat-sheats

Collaboration policy

Detailed version on the syllabus section of the website

You are **encouraged to discuss assignments** with other students:

1. You must **list** everyone you talked with about the assignment.
2. You **may not** share or look at each other's **written work or code**.
3. You must **write up** your solutions individually

Individual work only on **exams**: No collaboration allowed
If cheat-sheets have chunks of content that the same, then that is cheating

Academic conduct

- Submitting someone else's work as your own is **plagiarism**.
- So is helping someone else to submit your work as their own.
- We report **all cases** of academic misconduct to the university.
- The university takes academic misconduct **very seriously**.
  Possible consequences:
  - Zero on the assignment or exam (virtually guaranteed)
  - Zero for the course
  - Permanent notation on transcript
  - Suspension or expulsion from the university
- **If you are thinking of cheating, since you are stuck or doing poorly, please just talk to me instead. We'll figure it out.**

Additional Questions

- Any questions you have are likely answered in the FAQ and Getting Started document that we have linked on eClass
  - Policies like “No late assignments accepted”, “How to contact TAs”, “What to do if you are going to miss a deadline or exam”
  - “How can I get extra resources?” and “How can I brush up on my math background?”

Readings and Thought Questions

- **It is critical that you do the readings**
- I wrote the notes, and in class lectures essentially follow them quite closely
- If you read and understand the notes, you have learned a lot about ML
- Marked Thought Questions encourage you to actually do the readings

Thought questions

- Thought questions correspond to readings in the notes
- They should demonstrate that you have read **and thought about** the topics
- Needn't have an answer

General format:

1. First, show/explain how you understand a concept
2. Given this context, propose a follow-up question
3. Optional: Proposal an answer to the question, or the way you might find it

Example: "Good" Thought Question

"After reading about independence, I wonder how one could check in practice if two variables are independent, given a database of samples? Is this even possible? One possible strategy could be to approximate their conditional distributions, and examine the effects of changing a variable. But it seems like there could be other more direct or efficient strategies."

Example: "Bad" Thought Questions

- "I don't understand linear regression. Could you explain it again?"
  - i.e., a request for an explanation. If you want to request a clarification, please use slido. avoid any clarification requests from thought questions
- "Derive the maximum likelihood approach for a Gaussian."
  - i.e., an exercise question from a textbook. This is not showing your understanding
- "What is the difference between a probability mass function and a probability density function?"
  - i.e., a question that could be directly answered by reading definitions
  - *BUT* the following modification would be fine: "I understand that PMFs are for discrete random variables and PDFs are for continuous random variables. Is there a way we could define probabilities over both discrete and continuous random variables in a unified way, without having to define two different kinds of function?"

Thought Question marks (10%)

- Four Thoughts Question deadlines (TQ1, TQ2, TQ3, TQ4)
- For each, you need to submit two questions about different subsections in the readings
  - e.g., for TQ1, you might submit one for Section 2.1 and say one for Section 3.2 (please label the corresponding question in your submission)
  - Sometimes the question is more high-level and spans sections. That is fine too; you can write (Spans sections) as the section
- 9% of this mark is for the average of the best three of four
- 1% of this mark is for posting your question on Discord for feedback

One Final Comment

- There is **a lot** to learn in Machine Learning
- You might ask yourself, why are we learning topic X and not topic Y?
  - For example, you might have heard of GANs and are wondering why we learn about VAEs instead of GANs
  - Or you might wonder why we learn about PCA, when everyone just uses neural networks anyway
- **The answer**: my primary goals it to teach you **skills** not topics
  - certain algorithms/topics are useful case studies to teach those skills
  - if you know the underlying concept/approach, you can learn new (more advanced) things yourself

On to the course!

- The introductory chapter discusses
  - Generative Models and Predictors
  - The Blessing and Curse of Dimensionality
  - Matrix Methods
  - A Brief Refresher of Basics of ML
- Let us briefly discuss those here before moving to Intermediate Probability

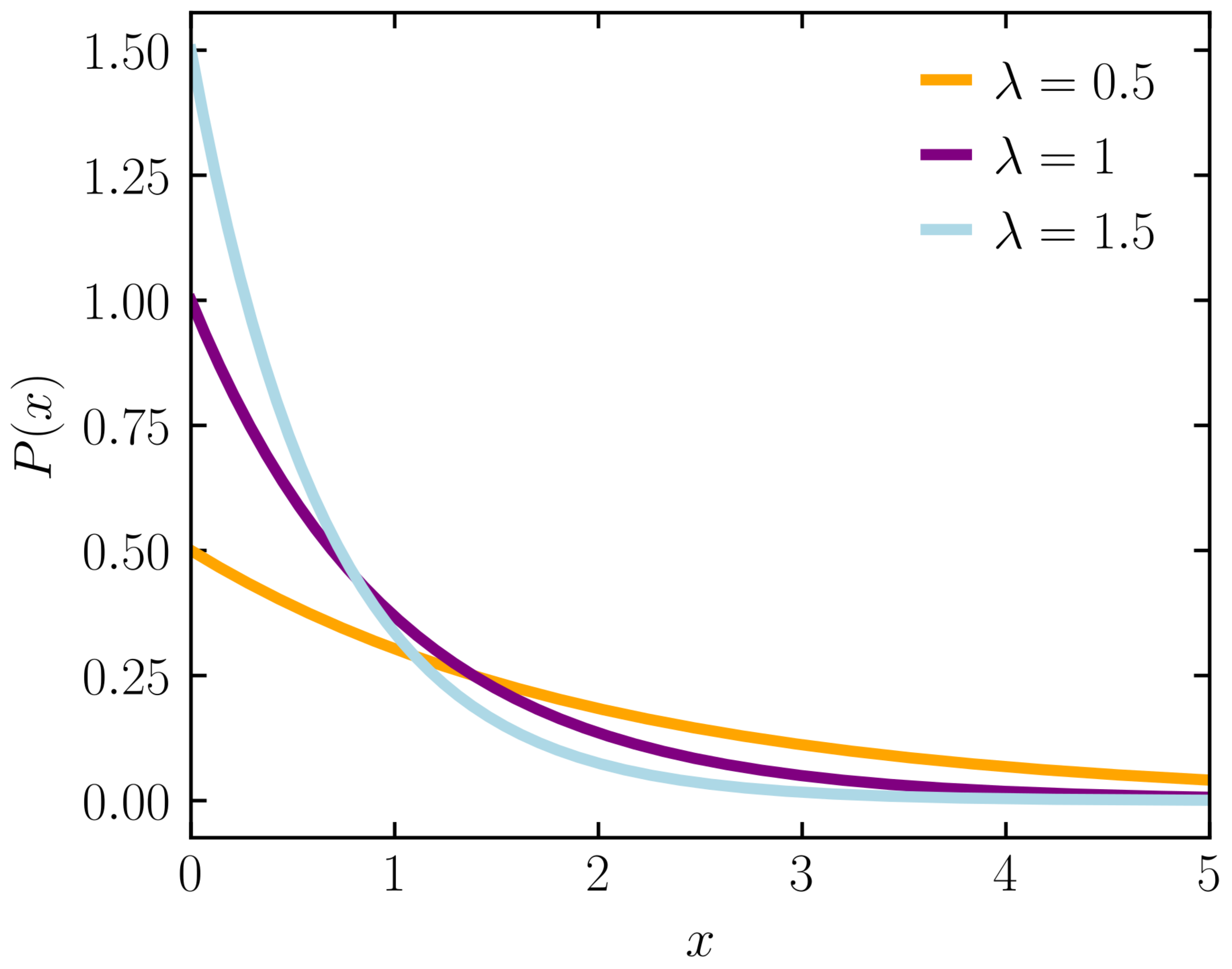
Prediction Models and Generative Models

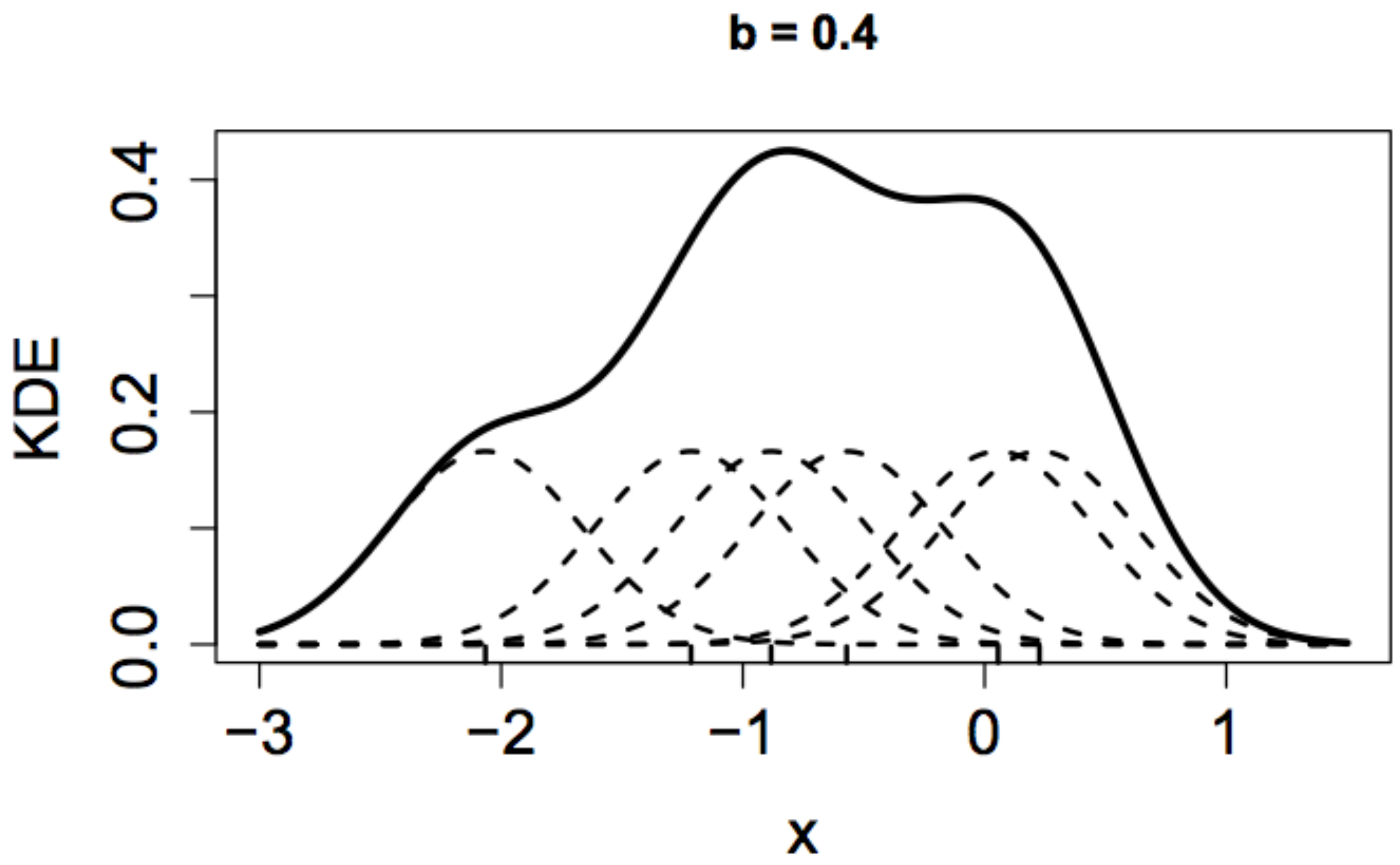
- We looked at both learning $p(x)$ and $p(y \mid x)$
- We usually think of $p(x)$ as a generative model and learn $p(y \mid x)$ for prediction (using $\mathbb{E}[Y \mid x]$ for regression and $p(y \mid x)$ for classification)
- This distinction is not quite right. Rather, key is how we use these models
- **Generative models**: learn (complex) distributions to generate potential outcomes; focus is obtaining accurate models of the target variable
- **Prediction models**: learn (simple) distributions to facilitate prediction; focus is obtaining useful predictions, even if distribution not quite right

Examples

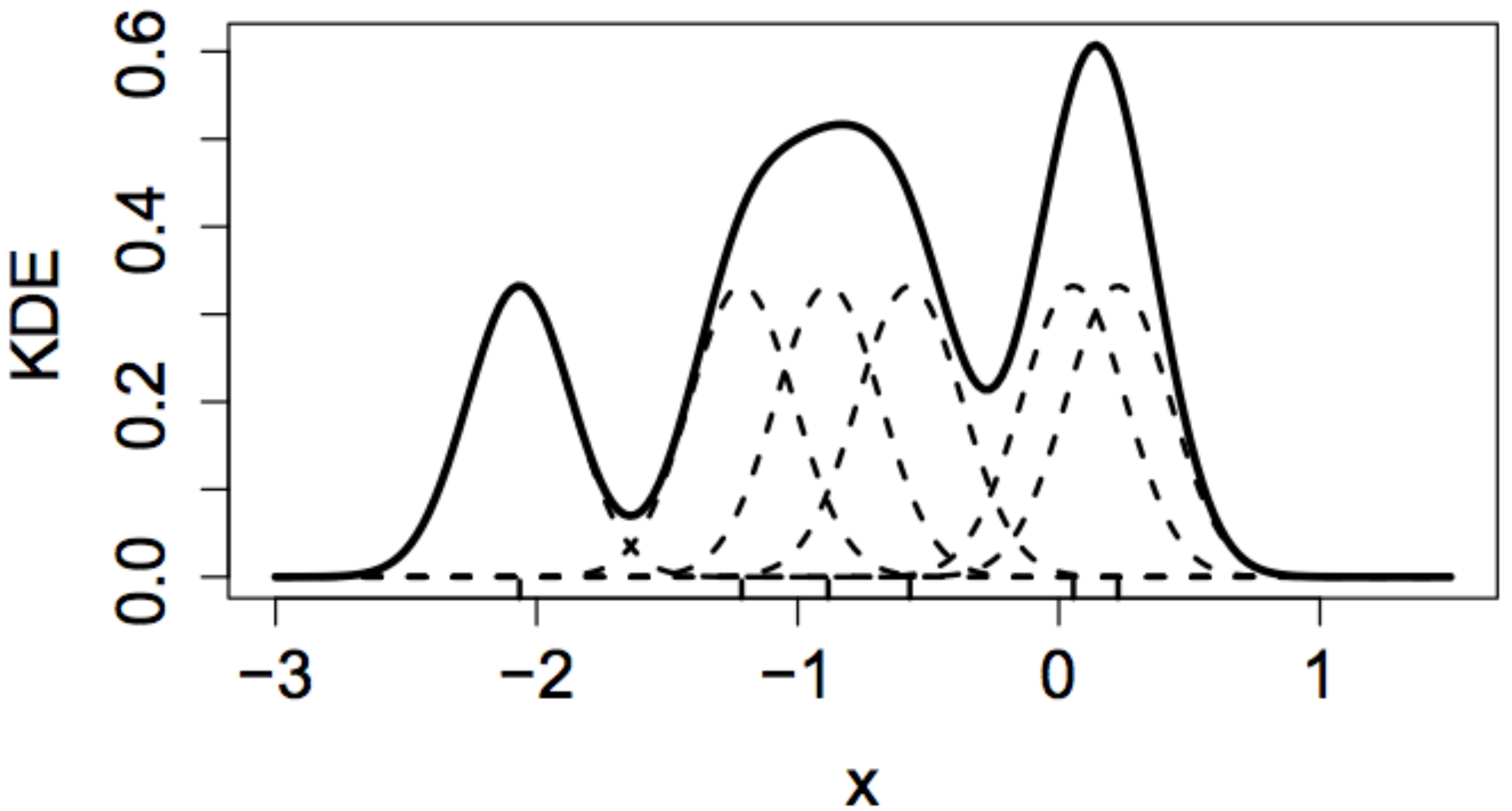
- Let X be images of faces (a multi-dimensional RV) and Y be a binary RV that is 0 if the face is not narrow and 1 if it is narrow
- $p(x)$ is a generative model, because we will simulate hypothetical faces by sampling $x \sim p$
- $p(y \mid x)$ is a prediction model, since we will use this to classify if the face is narrow or not narrow
- $p(x \mid y)$ is a conditional generative model, because we will simulate hypothetical faces, conditioned on whether they are narrow or not
- The distinction is primarily on the complexity of the RV that we are modelling; otherwise, for each case we still learn a (conditional) distribution

What is a complex distribution?





More simple → More complex



This distinction matters a lot

- Once we are modelling more complex variables, then we have to consider how to do so efficiently and still enable sampling from that distribution
- Sampling from a Bernoulli is easy. Sampling from the set of all faces is harder
- So, though they are clearly highly related and the distinction is not quite crisp, the field of generative modelling is quite distinct from prediction
- For prediction, we often care primarily about classification (simple discrete targets) or means of targets (modelled as univariate Gaussians)
- For generative models, we often care about learning complex distributions

Blessing and Curse of Dimensionality

- Interesting concentration phenomena occur in high-dimensions
  - The volume of a high-dimensional ball concentrates near its surface, rather than the interior
- This phenomena has ramifications for us when learning
  - Blessing: data becomes separable in high-dimensions
  - Curse: distances become less meaningful
- High-dimensional representations can significantly improve performance, we simply have to be careful about how we use them

Let us now no longer use these loaded terms

Matrix Methods

- Basics of ML (mostly) avoided matrices
- Intermediate ML will embrace this tool (linear algebra is useful)
- Primarily, we use:
  - Matrix-vector and matrix-matrix product
  - Matrix inverses
  - Matrix decompositions (eigenvalue decomposition, svd)

A matrix is an mxn array

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \dots \\ \mathbf{a}_m \end{bmatrix}$$

Matrix-vector product

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} \langle \mathbf{a}_1, \mathbf{x} \rangle \\ \langle \mathbf{a}_2, \mathbf{x} \rangle \\ \cdots \\ \langle \mathbf{a}_m, \mathbf{x} \rangle \end{bmatrix} = \begin{bmatrix} \langle \mathbf{A}_{1:}, \mathbf{x} \rangle \\ \langle \mathbf{A}_{2:}, \mathbf{x} \rangle \\ \cdots \\ \langle \mathbf{A}_{m:}, \mathbf{x} \rangle \end{bmatrix} \in \mathbb{R}^m$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \dots \\ \mathbf{a}_m \end{bmatrix}$$

Matrix-matrix product

$$A \in \mathbb{R}^{m \times n} \qquad \mathbf{B} \in \mathbb{R}^{n \times k}$$

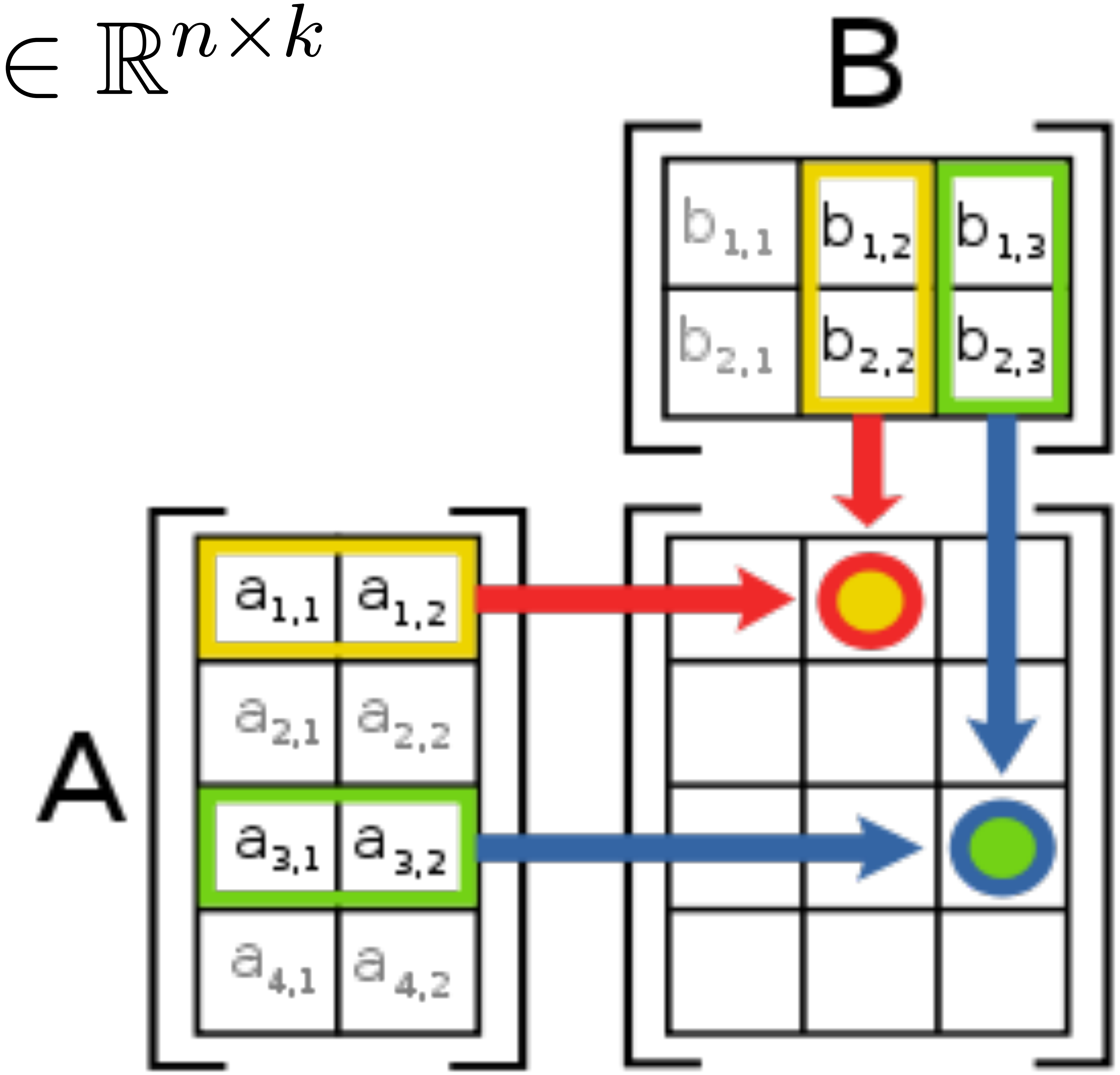
$$\mathbf{AB} = [\mathbf{AB}_{:,1}, \mathbf{AB}_{:,2}, \dots, \mathbf{AB}_{:,k}] = \begin{bmatrix} \mathbf{A}_{1,:}\mathbf{B}_{:,1} & \mathbf{A}_{1,:}\mathbf{B}_{:,2} & \dots & \mathbf{A}_{1,:}\mathbf{B}_{:,k} \\ \mathbf{A}_{2,:}\mathbf{B}_{:,1} & \mathbf{A}_{2,:}\mathbf{B}_{:,2} & \dots & \mathbf{A}_{2,:}\mathbf{B}_{:,k} \\ & \dots & & \\ \mathbf{A}_{m,:}\mathbf{B}_{:,1} & \mathbf{A}_{m,:}\mathbf{B}_{:,2} & \dots & \mathbf{A}_{m,:}\mathbf{B}_{:,k} \end{bmatrix} \in \mathbb{R}^{m \times k}$$

A nicer picture of matrix multiplication

$A \in \mathbb{R}^{m \times n}$ $\mathbf{B} \in \mathbb{R}^{n \times k}$

What is m, n and k in this example?

m = 4, n = 2, k = 3



Matrix-matrix product

$$A \in \mathbb{R}^{m \times n} \qquad \mathbf{B} \in \mathbb{R}^{n \times k}$$

$$\mathbf{AB} = [\mathbf{AB}_{:,1}, \mathbf{AB}_{:,2}, \ldots, \mathbf{AB}_{:,k}] = \begin{bmatrix} \mathbf{A}_{1,:}\mathbf{B}_{:,1} & \mathbf{A}_{1,:}\mathbf{B}_{:,2} & \ldots & \mathbf{A}_{1,:}\mathbf{B}_{:,k} \\ \mathbf{A}_{2,:}\mathbf{B}_{:,1} & \mathbf{A}_{2,:}\mathbf{B}_{:,2} & \ldots & \mathbf{A}_{2,:}\mathbf{B}_{:,k} \\ & \ldots & & \\ \mathbf{A}_{m,:}\mathbf{B}_{:,1} & \mathbf{A}_{m,:}\mathbf{B}_{:,2} & \ldots & \mathbf{A}_{m,:}\mathbf{B}_{:,k} \end{bmatrix} \in \mathbb{R}^{m \times k}$$

Notice that the inner dimension matches:

$m \times n$ times $n \times k$ produces a $m \times k$ matrix

It is an easy rule of thumb to check if you have made a mistake somewhere, by checking that these dimension match and you have a valid operation

Matrix Inverse for Diagonal Matrix

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 & \dots & 0 & 0 \\ 0 & a_2 & 0 & \dots & 0 \\ & \dots & & & \\ 0 & 0 & \dots & 0 & a_d \end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix} 1/a_1 & 0 & \dots & 0 & 0 \\ 0 & 1/a_2 & 0 & \dots & 0 \\ & \dots & & & \\ 0 & 0 & \dots & 0 & 1/a_d \end{bmatrix}$$

where you can verify that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ for identity matrix $\mathbf{I}$ that has 1s on the diagonal

$$\mathbf{I} \stackrel{\mathsf{def}}{=} \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \dots & & & \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

Matrix Decompositions

- Singular Value Decomposition (SVD)
  - every matrix has an SVD
- Eigenvalue Decomposition
  - every square, symmetric matrix has an eigenvalue decomposition
  - other matrices do too, but we don’t need to reason about the eigenvalue decomposition for anything by square, symmetric matrices
- These decompositions are useful for reasoning about the properties of the matrix and computing the inverse of the matrix

A matrix as an operator

- $\mathbf{M}$ is an operator on vectors: $\tilde{\mathbf{x}} = \mathbf{M}\mathbf{x}$
  - it transforms the input vector $\mathbf{x}$ to a new $\tilde{\mathbf{x}}$
- How can we reason about the properties of this operator?

Singular Value Decomposition

- $\mathbf{M}$ is an operator on vectors: $\tilde{\mathbf{x}} = \mathbf{M}\mathbf{x}$
  - it transforms the input vector $\mathbf{x}$ to a new $\tilde{\mathbf{x}}$
- Any matrix can be decomposed using an SVD: $\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$
- $\boldsymbol{\Sigma}$ is a **diagonal matrix** with nonnegative elements on the diagonal
- $\mathbf{U}, \mathbf{V}$ are **orthonormal matrices,** meaning that
  - $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$
  - $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$

Singular Value Decomposition

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \qquad \mathbf{M}\mathbf{x} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{x} = \mathbf{U}\boldsymbol{\Sigma}(\mathbf{V}^\top\mathbf{x})$$

Every matrix is a linear operator that can be decomposed into a rotation (V), scaling (Sigma), and rotation (U) operation

Exercise: What happens if a singular value is zero?

$$\mathbf{M}\mathbf{x} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{x} = \mathbf{U}\boldsymbol{\Sigma}(\mathbf{V}^\top\mathbf{x})$$

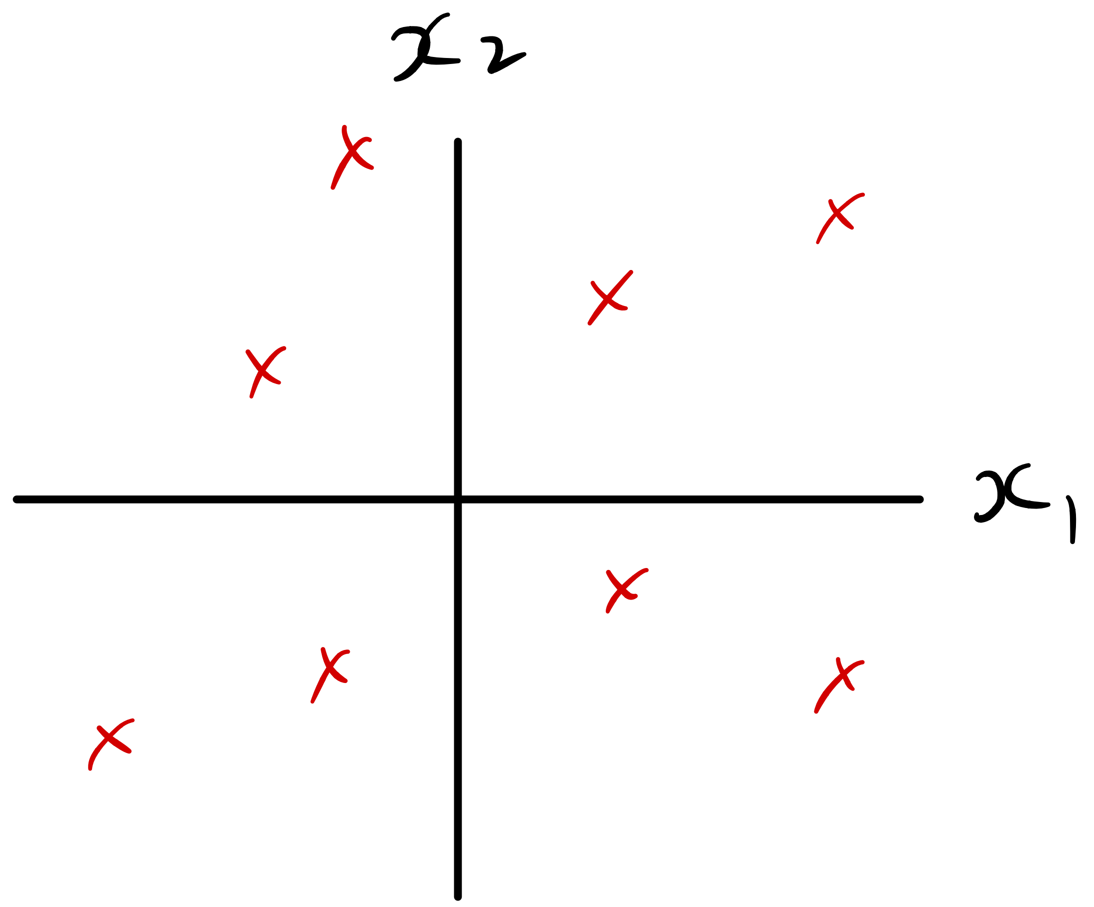
- Every matrix is a linear operator that can be decomposed into
a rotation (V), scaling (Sigma), and rotation (U) operation
- What does the scaling operation do?

- Answer: it zeros out a component of $\tilde{\mathbf{x}} = \mathbf{V}^\top\mathbf{x}$
- $\mathbf{U}\tilde{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{u}_i\tilde{x}_i = \sum_{i=1}^{n-1} \mathbf{u}_i\tilde{x}_i$ is a weighted sum of n-1 basis vector
- It reduces the dimension by 1: it projects the vector into a lower-dimensional space

Example using SVD on data matrix

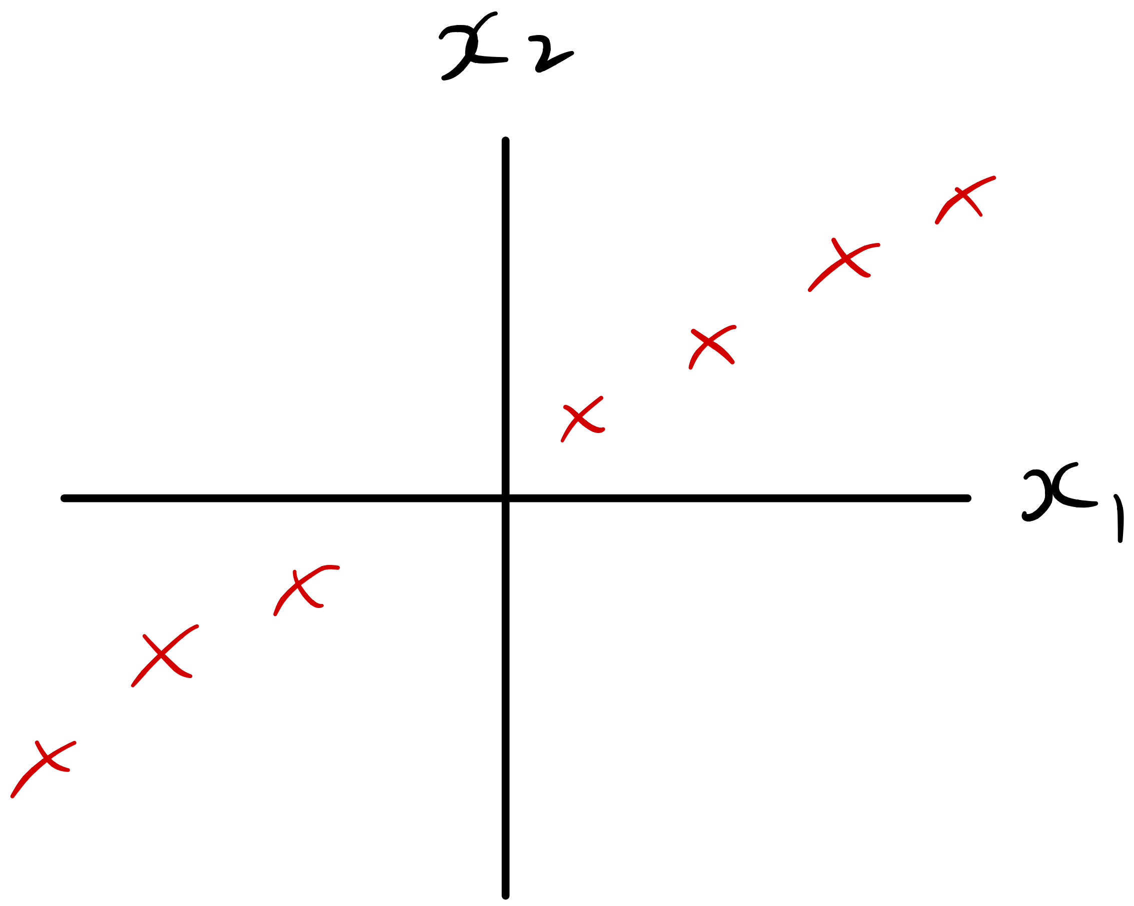
- $\mathbf{X} \in \mathbb{R}^{n \times d}$ for n samples and d features
- Let's imagine d = 2
- $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top = [\mathbf{u}_1, \mathbf{u}_2]\text{diag}(\sigma_1, \sigma_2)[\mathbf{v}_1^\top; \mathbf{v}_2^\top] = \sum_{j=1}^{2} \sigma_j \mathbf{u}_j \mathbf{v}_j^\top$
- where $\mathbf{u}_j \in \mathbb{R}^n, \sigma_j \geq 0, \mathbf{v}_j \in \mathbb{R}^2$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2]$
- A row (sample) equals $\mathbf{x}_i = U_{i1}\sigma_1\mathbf{v}_1^\top + U_{i2}\sigma_2\mathbf{v}_2^\top = \beta_1\mathbf{v}_1^\top + \beta_2\mathbf{v}_2^\top$
    - a linear combination of (right singular) vectors $\mathbf{v}_j$

Visualizing $\sigma_2 = 0$



Full rank X

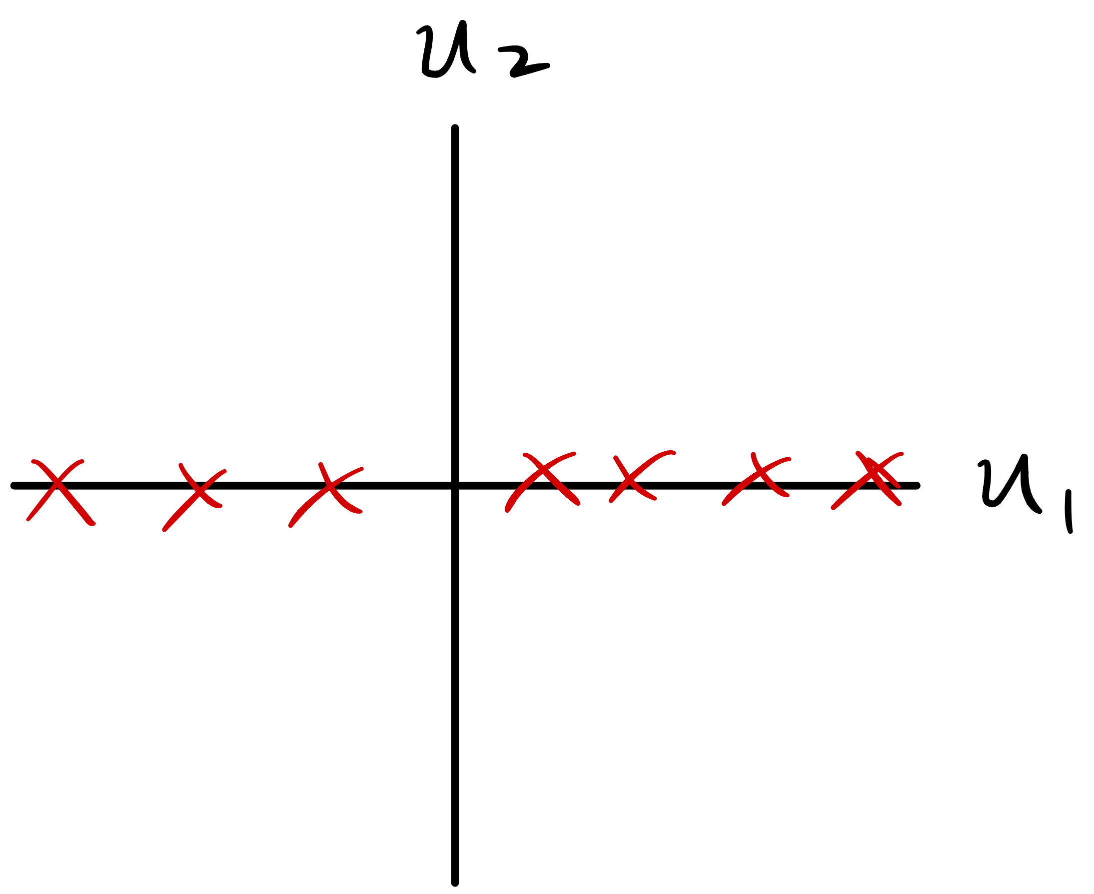
$$X = [u_1 \; u_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} [v_1^T ; v_2^T]$$



$\sigma_2 = 0$ for X

$\xrightarrow{\text{Rotate}}$

$$\hat{X} = XV$$
$$= \sigma_1 u_1$$



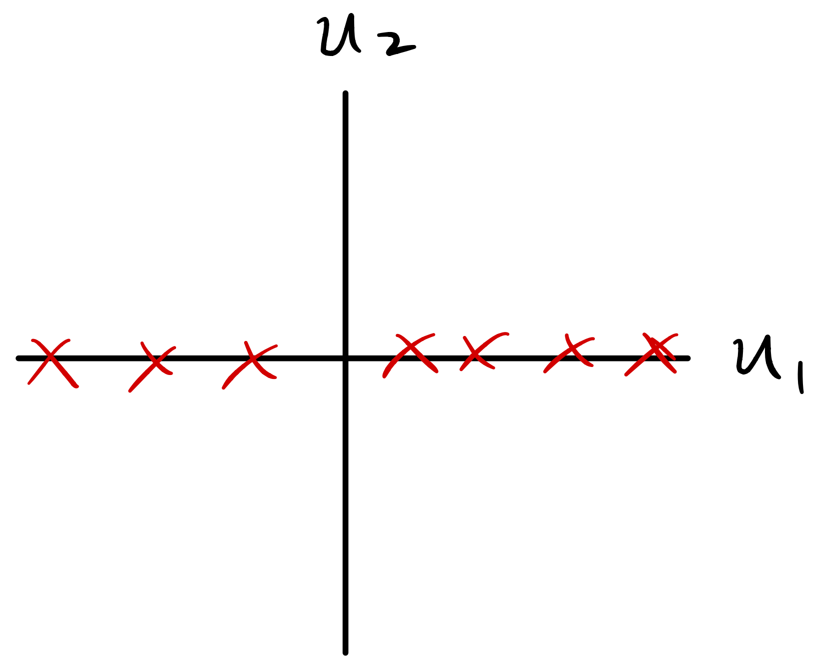
All points only vary in one dimension

The effect on predictions

Rotate $\xrightarrow{}$

$\tilde{X} = XV$

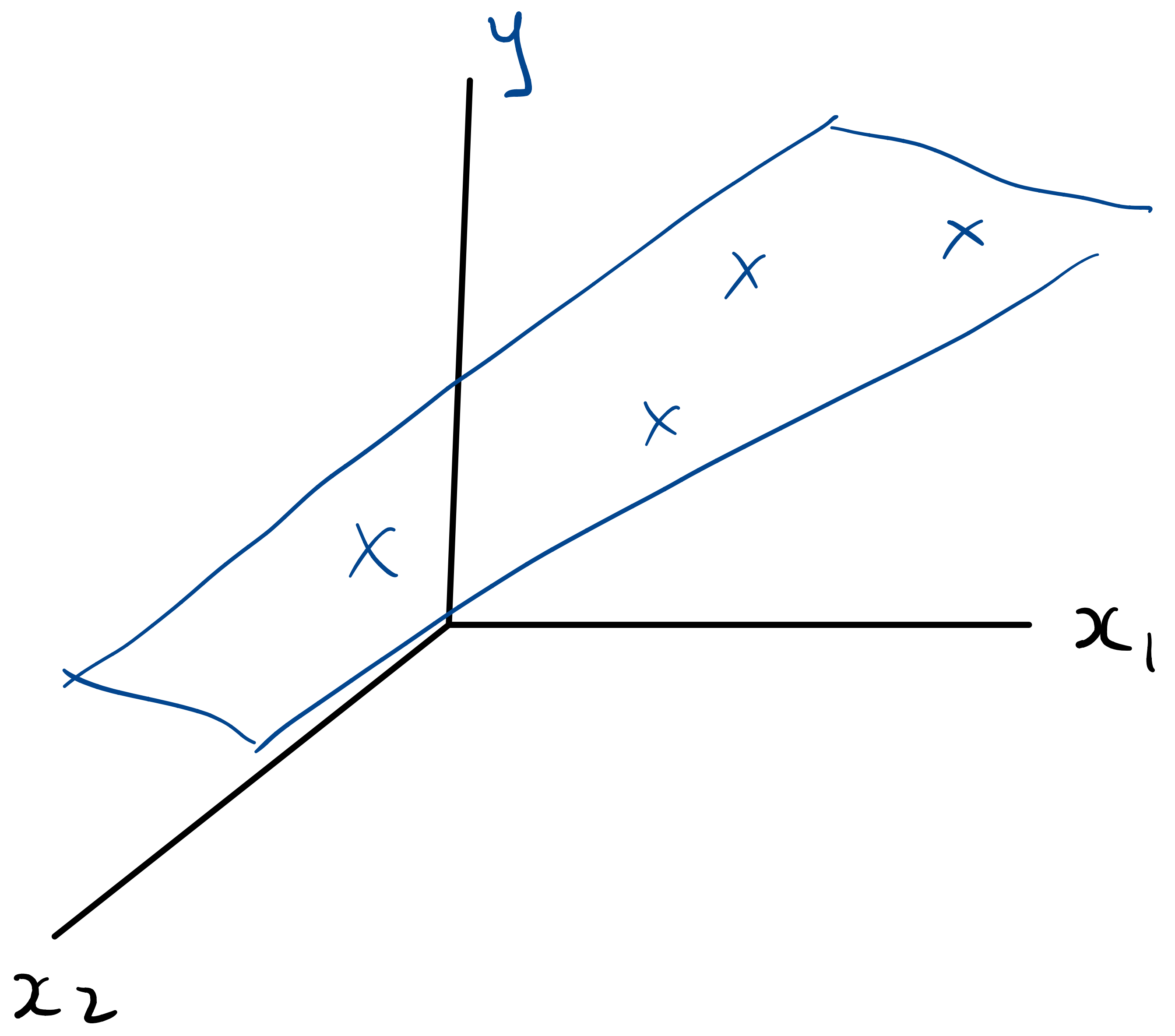
$= \sigma_1 u_1$



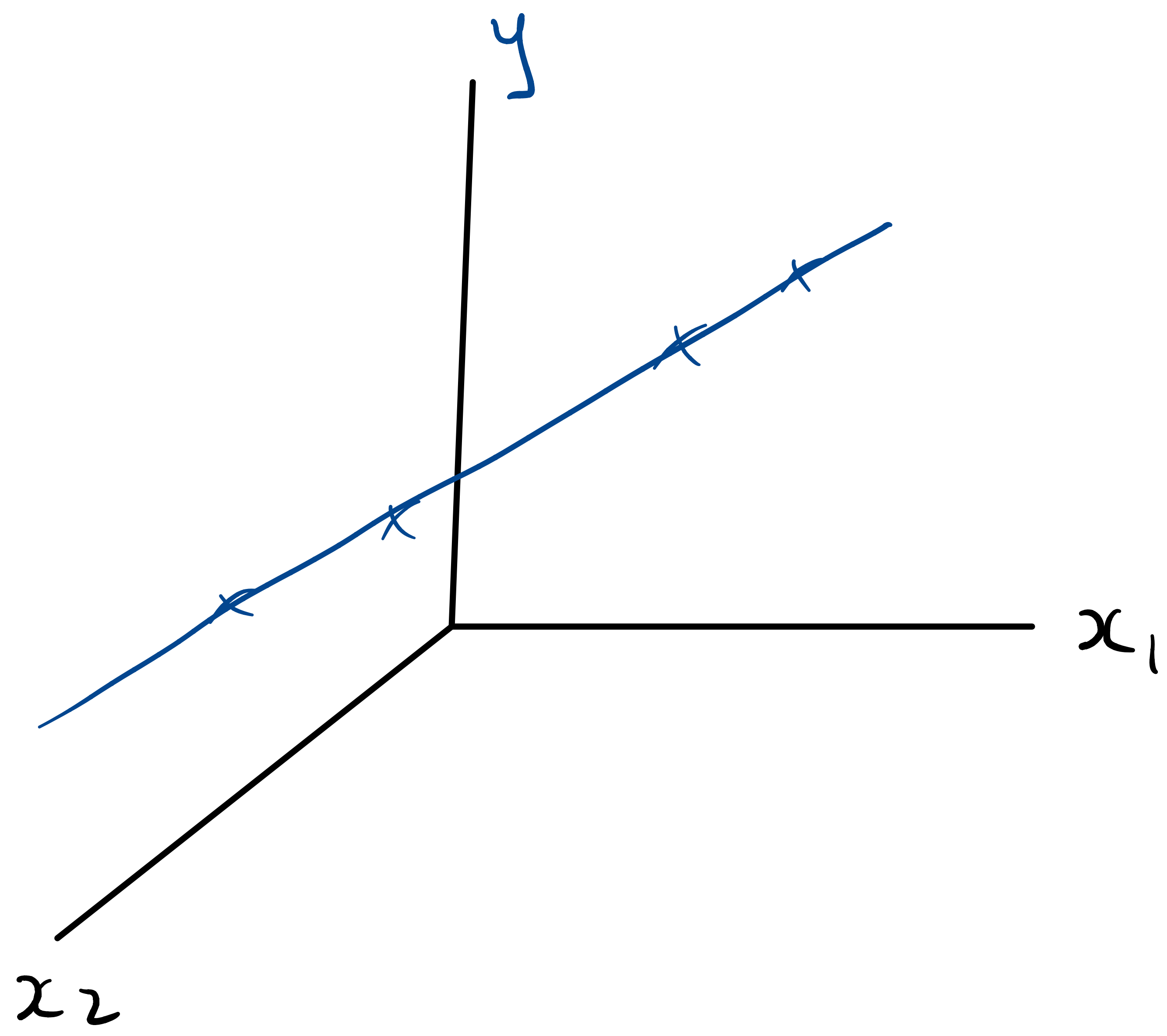
All points only vary in one dimension

- $\hat{y} = \mathbf{X}\mathbf{w} = \tilde{\mathbf{X}}\tilde{\mathbf{w}}$ where $\tilde{\mathbf{w}} = \mathbf{V}^{\top}\mathbf{w}$
- $\tilde{\mathbf{X}} = [\sigma_1\mathbf{u}_1, \mathbf{0}]$, meaning $\tilde{\mathbf{X}}\tilde{\mathbf{w}} = \sigma_1\mathbf{u}_1\tilde{w}_1$
- Effectively only have one degree of freedom
- Usually for a 2d input space, for a linear function, y lies on a 2d plane. Here, it lies on a 1d plane (a line)

Effectively learning in a lower-dimensional space



Full rank X



$\sigma_2 = 0$ for X

Rank of a matrix

- The number of non-zero singular values is the rank
- The rank of a matrix is the dimension of the space it spans
- In this example, it is the dimension of the space that it projects vectors to
- For a matrix with one singular value that is zero, it projects all vector to one dimension lower (a plane in dimension n-1 inside the large n-dimensional space)

Eigenvalue Decomposition

- A square symmetric matrix $\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
    - $\mathbf{\Lambda}$ is a diagonal matrix
    - Notice this is really an SVD, where the second rotation is $\mathbf{U}$ again
- Computing the inverse is now easy: $\mathbf{M}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^\top$
- How do we know? We can check.

Checking the Inverse Condition

$\mathbf{M}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top$

$$\begin{aligned}
\mathbf{M}\mathbf{M}^{-1} &= (\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top)\mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \\
&= \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top\mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \\
&= \mathbf{U}\boldsymbol{\Lambda}\mathbf{I}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \\
&= \mathbf{U}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top \\
&= \mathbf{U}\mathbf{I}\mathbf{U}^\top \\
&= \mathbf{U}\mathbf{U}^\top \\
&= \mathbf{I}
\end{aligned}$$

Exercise: Check that $\mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$

Refresher of Basics of ML

- Goal was to learn a prediction function $f_{\mathbf{w}} : \mathscr{X} \rightarrow \mathscr{Y}$ for weights $\mathbf{w}$
- Input vector of observations $\mathbf{x} \in \mathbb{R}^d$
- Outputs a prediction $\hat{y} \in \mathscr{Y}$
- If $\mathscr{Y}$ is a discrete, unordered set, then we have a **classification** problem
- If $\mathscr{Y}$ is a continuous set, then we have a **regression** problem
- (Some cases we have a discrete, order set, and get ordinal regression)

Main goals

- How do we learn this function?
- How do we evaluate whether it is good?

Formalizing the learning problem

- We need a clear criterion (objective function) to optimize
- Ultimate goal: function with low **expected cost** $\mathbb{E}[\text{cost}(f(\mathbf{x}), y)]$
  - later we called this **generalization error**
- For regression, cost was squared error
  - Optimal predictor is $f^*(\mathbf{x}) = \mathbb{E}[Y \mid \mathbf{x}]$
- For classification, we used the 0-1 cost
  - Optimal predictor is $f^*(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y \mid \mathbf{x})$

Beyond formalization, to implementation

- We cannot directly find these optimal predictors, rather we are stuck using data sampled from $p(\mathbf{x}, y)$
- Formalized the MAP and MLE objectives on this data, as a reasonable proxy to approximate these optimal predictors
  - MAP $\max_\theta p(\theta \mid \mathcal{D})$ versus MLE $\max_\theta p(\mathcal{D} \mid \theta)$

MAP Objective

- For regression we assume $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$, and Gaussian prior on weights
- The MAP objective corresponded to l2 regularized linear regression (ridge regression)

$$
\begin{aligned}
\operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^k} p(\mathbf{w}|\mathcal{D}) &= \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^k} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^k} \sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i, \mathbf{w}) + \ln p(\mathbf{w}) \\
&= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^k} -\sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i, \mathbf{w}) - \ln p(\mathbf{w})
\end{aligned}
$$

- The objective became $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$

MLE Objective

- MLE for was the linear regression objective, without regularization (no prior)
- MLE objective is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$
- For logistic regression (binary classification), it was the cross-entropy objective

MLE Objective

- MLE for was the linear regression objective, without regularization (no prior)
- MLE objective is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ MAP: $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$
- For logistic regression (binary classification), it was the cross-entropy objective
- **Question**: what is the MAP objective for binary classification with a Gaussian prior?
- **Question**: what is the MAP objective for polynomial regression with a Gaussian prior?

MLE Objective

- MLE for was the linear regression objective, without regularization (no prior)
- MLE objective is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ MAP: $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$
- For logistic regression (binary classification), it was the cross-entropy objective
- **Question**: what is the MAP objective for binary classification with a Gaussian prior?
  - Answer: cross-entropy$(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$
- **Question**: what is the MAP objective for polynomial regression with a Gaussian prior?
  - Answer: $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$ for $f_{\mathbf{w}}(\mathbf{x}_i)$ a polynomial function

MAP vs MLE

- MAP adds prior information, to help prevent **overfitting**
- The l2 regularizer preferred simpler solutions, those where weights did not deviate too far from zero
  - We will revisit why very soon, now with matrices

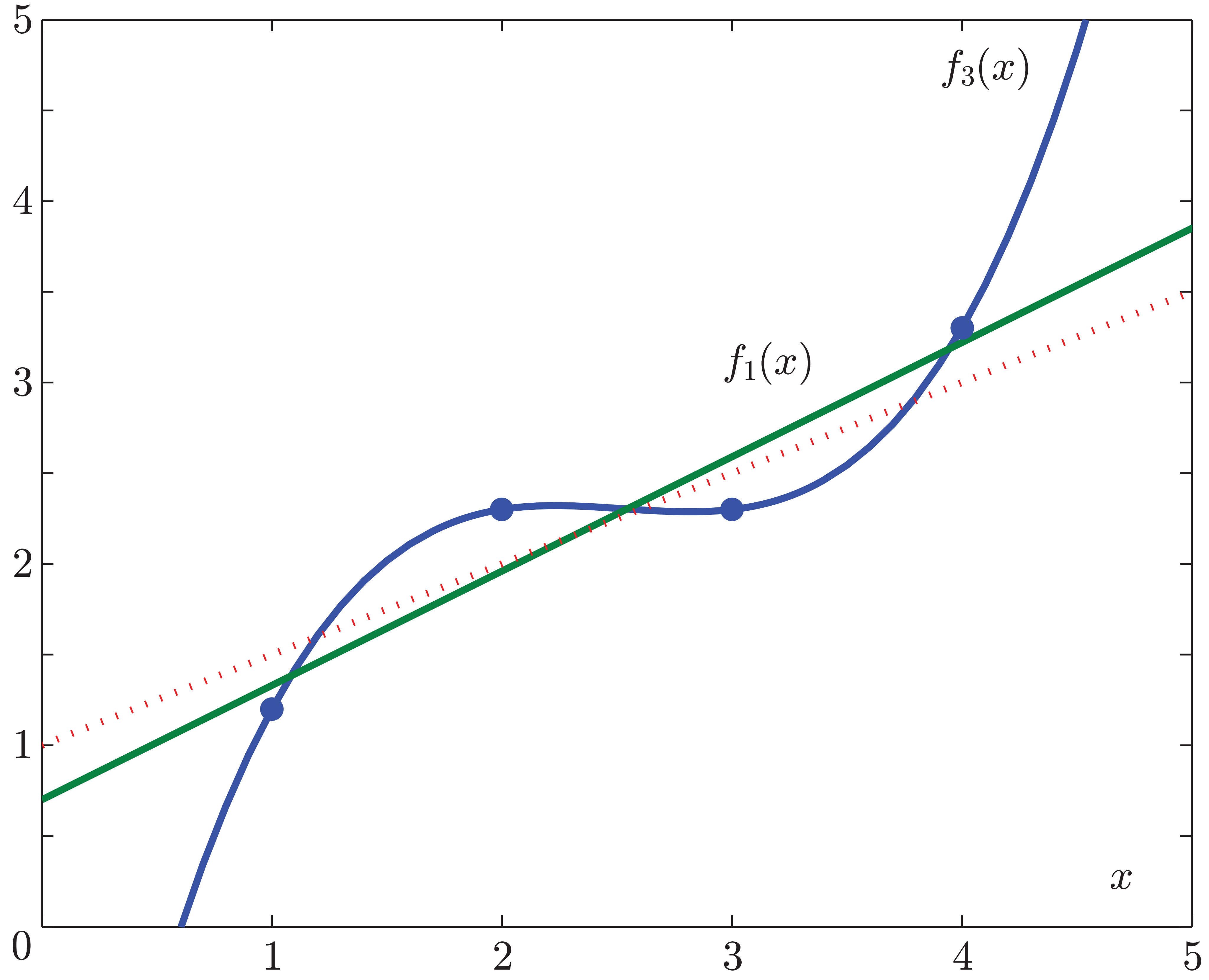
Announcements

- Assignment 1 released, please get started!
  - Make sure you upgrade Julia
- Thought Questions 1 due very soon (September 20)
  - Biggest reading since it covers much of the background
- Some typos in the notes, updated on the website
  - I will be adding more examples, but I will try to limit big modifications to the notes only to later parts, before Reading Assignments
- Assignment 0 also released, these are just practice exercises for your fun

Exercise

- We derived linear regression using MLE
- How would you derive polynomial regression using MLE?

Polynomial function is a strict generalization of linear functions



Polynomial regression derivation

- $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$ for polynomial function $f_{\mathbf{w}}(\mathbf{x})$
- MLE objective is $\sum_{i=1}^{n} - \ln p(y_i \mid \mathbf{x}_i)$

$$\ln p(y_i \mid \mathbf{x}_i) = -\frac{1}{2}\ln(2\pi\sigma^2) + \ln\exp\left(-\frac{1}{2\sigma^2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2\right)$$

- $= \text{constants} - \frac{1}{2\sigma^2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$

Polynomial regression derivation

- $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$ for polynomial function $f_{\mathbf{w}}(\mathbf{x})$
- $\ln p(y_i \mid \mathbf{x}_i) =$ constants $- \dfrac{1}{2\sigma^2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$
- MLE objective is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ because for constants c_1, c_2
- $\arg\min_{\mathbf{w}} \sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \arg\min_{\mathbf{w}} c_1 + c_2 \sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$

l2-regularized polynomial regression

- Do you think l2 regularization is more useful for polynomial regression
  - (Q1) for high-degree polynomials rather than low-degree ones?
  - (Q2) than for linear regression?

Evaluating a function

- Now we know how to find a function $f_{\mathbf{w}}$, but how do we evaluate if it is good?
- One simple way is to split the data into training and test data
  - e.g., take a dataset of size 10k, use 8k for training, 2k for testing
- Then we learn $f_{\mathbf{w}}$ on the training data
- And get an estimate of generalization error on the test set

Exercise

- Imagine we learned $f_{\mathbf{w}}$ using polynomial regression with p=3
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$
- What is the size of $\mathbf{w}$?
- How do we evaluate $f_{\mathbf{w}}$ on the test set? (What is the formula)

Exercise

- Imagine we learned $f_{\mathbf{w}}$ using polynomial regression with p=3
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$
- What is the dimension (size) of $\mathbf{w}$?
  - Same number of elements as $\boldsymbol{\phi}(\mathbf{x})$
- How do we evaluate $f_{\mathbf{w}}$ on the test set? (What is the formula)
- $\frac{1}{m} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{test}}} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ for m the number of test samples

Exercise

- Imagine we learned $f_{\mathbf{w}}$ using polynomial regression with p=3
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$
- If we had learned $f_{\boldsymbol{\beta}}$ with p = 2, then would
  - $f_{\boldsymbol{\beta}}$ have lower or higher **training** error than $f_{\mathbf{w}}$?
  - $f_{\boldsymbol{\beta}}$ have lower or higher **testing** error than $f_{\mathbf{w}}$?

Exercise

- Imagine we learned $f_{\mathbf{w}}$ using polynomial logistic regression with p=3
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = 1$ if $\boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w} > 0,$ else = 0
- How do we evaluate $f_{\mathbf{w}}$ on the test set? (What is the formula?)

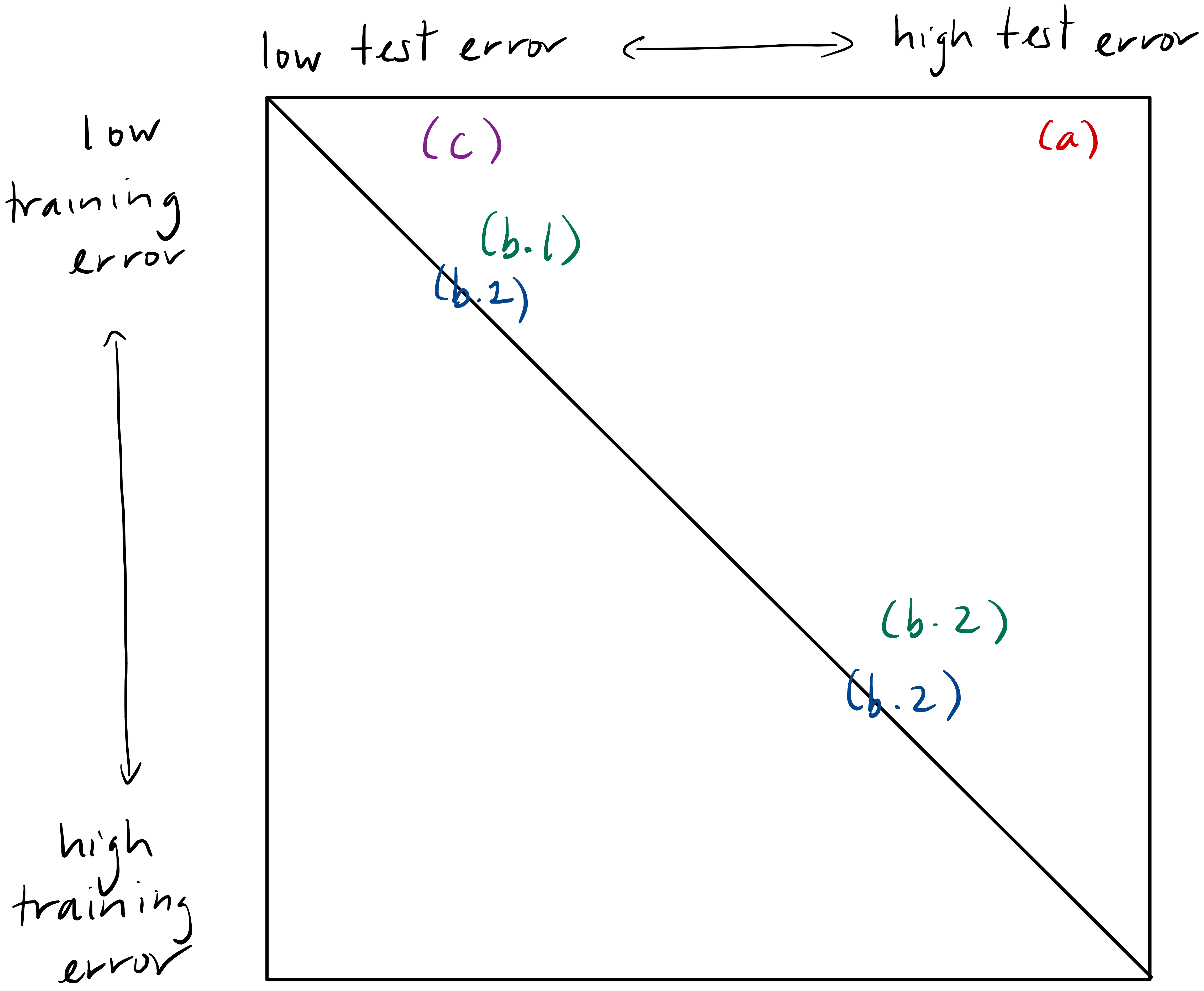
Exercise

- Imagine we learned $f_{\mathbf{w}}$ using polynomial logistic regression with p=3
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = 1$ if $\boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w} > 0$, else = 0
- How do we evaluate $f_{\mathbf{w}}$ on the test set? (What is the formula?)
- $\displaystyle \frac{1}{m} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{test}}} 1(f_{\mathbf{w}}(\mathbf{x}_i) \neq y_i)$ for m the number of test samples

Conceptual Evaluation

- We can empirically evaluate and we often reason about when estimators should or should not perform well
- We discussed bias and variance, and the connection to generalization error
- If it sometimes worth introducing bias to reduce variance, and so reduce the MSE to the true function in expectation

Different Cases



(a) η small, $\mathcal{F}$ complex
(e.g. $\mathcal{F}$ = 8^{th} degree polynomials)

(b) η small, $\mathcal{F}$ simple (e.g linear)

Case 1: f_{true} simple

Case 2: f_{true} complex

(c) η big, $\mathcal{F}$ complex

(d) η big, $\mathcal{F}$ simple

Case 1: f_{true} simple

Case 2: f_{true} complex

Uncertainty in Our Estimator

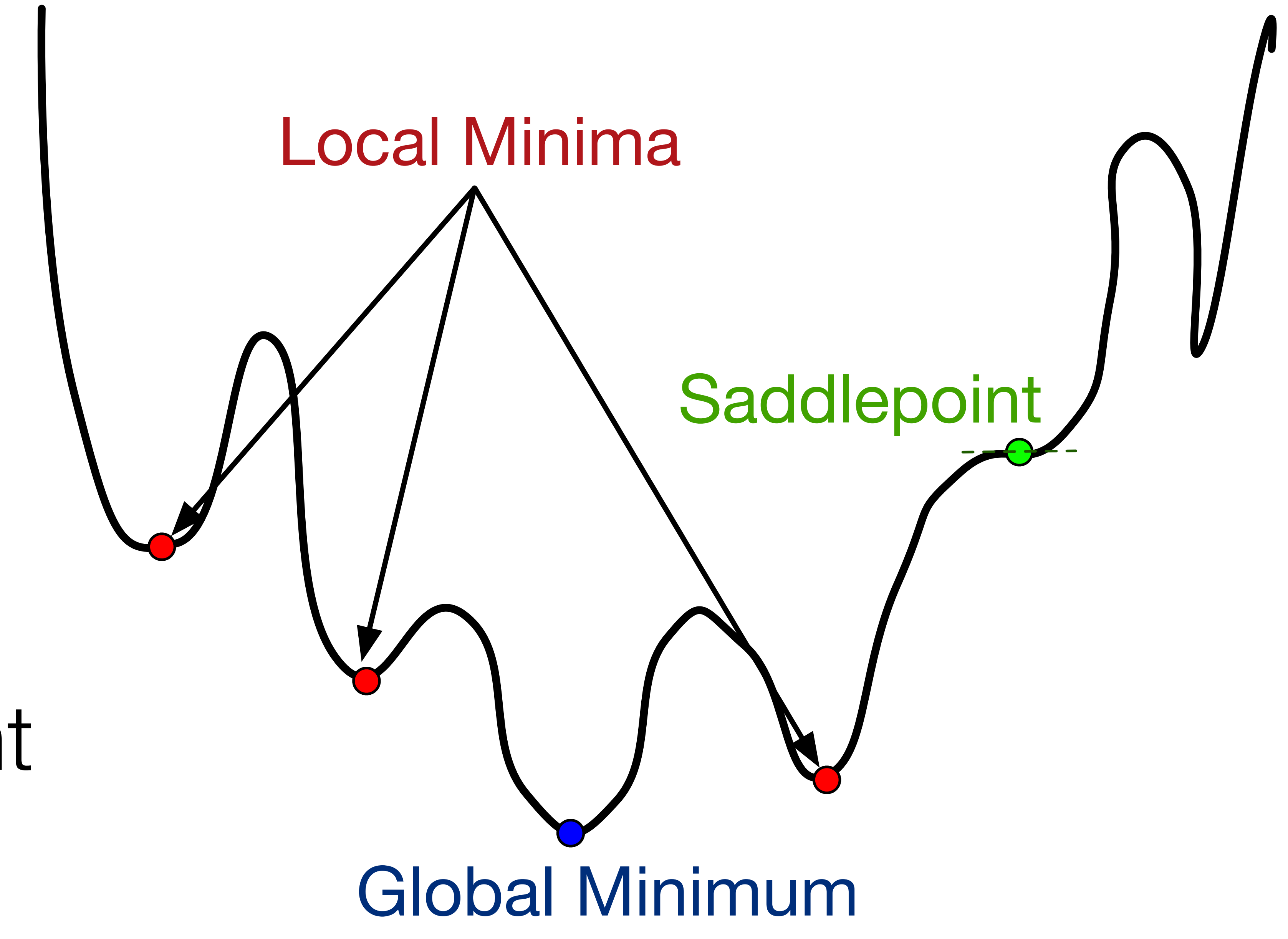
- Confidence intervals to assess uncertainty in a mean estimator
  - obtained using distributional assumptions like the student-t and with less assumptions using concentration inequalities
  - also discussed using Bayesian approach to get credible interval
- Bayesian methods obtain $p(\mathbf{w} \mid \mathscr{D})$the posterior
  - can use this to get a credible interval over the weights and over predictions

Probability and Optimization at the Core

- Through used optimization tools, to have practical ways to obtain weights
- The objective function told us what to optimize, but not how to do so
- We discussed
  - brute-force search for low-dimensional, discrete problems
  - gradient descent for smooth, continuous optimization problems
  - more efficient approximation to GD using mini-batch stochastic GD (SGD)
  - when GD or SGD will reach global solutions or get stuck in local minima (or saddlepoints)

Fun Fact about Saddlepoints

- We seemed to worry about local minima than saddlepoints
- It is believed that SGD often skips past saddlepoints
- It is actually hard work to descend perfectly to a saddlepoint more likely you overshoot and keep descending
- It is harder to jump out of a local minima, using only the stochasticity from SGD



Final Exercise

- Let’s go back to the first example we looked at in Basics of ML and see if you can answer the questions for yourself now
  - And see if you can explain it to your past self

Example: Predicting house prices

- **Goal**: we want to predict house prices, given only the age of the house

  $f(age) =$ price of the house

- **Dataset**: house sales this year, with attributes **age** and target value **price**

  $\{(age_1, price_1), (age_2, price_2), \ldots, (age_9, price_9)\}$

- **Idea**: A function that accurately outputs price from age for these specific pairs might also provide good predictions for new houses

Formalizing the problem

Definitions:

Let x be **age** and y be **price**

Let $D = \{(x_1, y_1), \ldots, (x_9, y_9)\}$ be our dataset

Objective:

We want to make the **difference** between $f(x_i)$ and y_i **small**

$$\min_{f \text{ in function space}} \sum_{i=1}^{9} (f(x_i) - y_i)^2$$

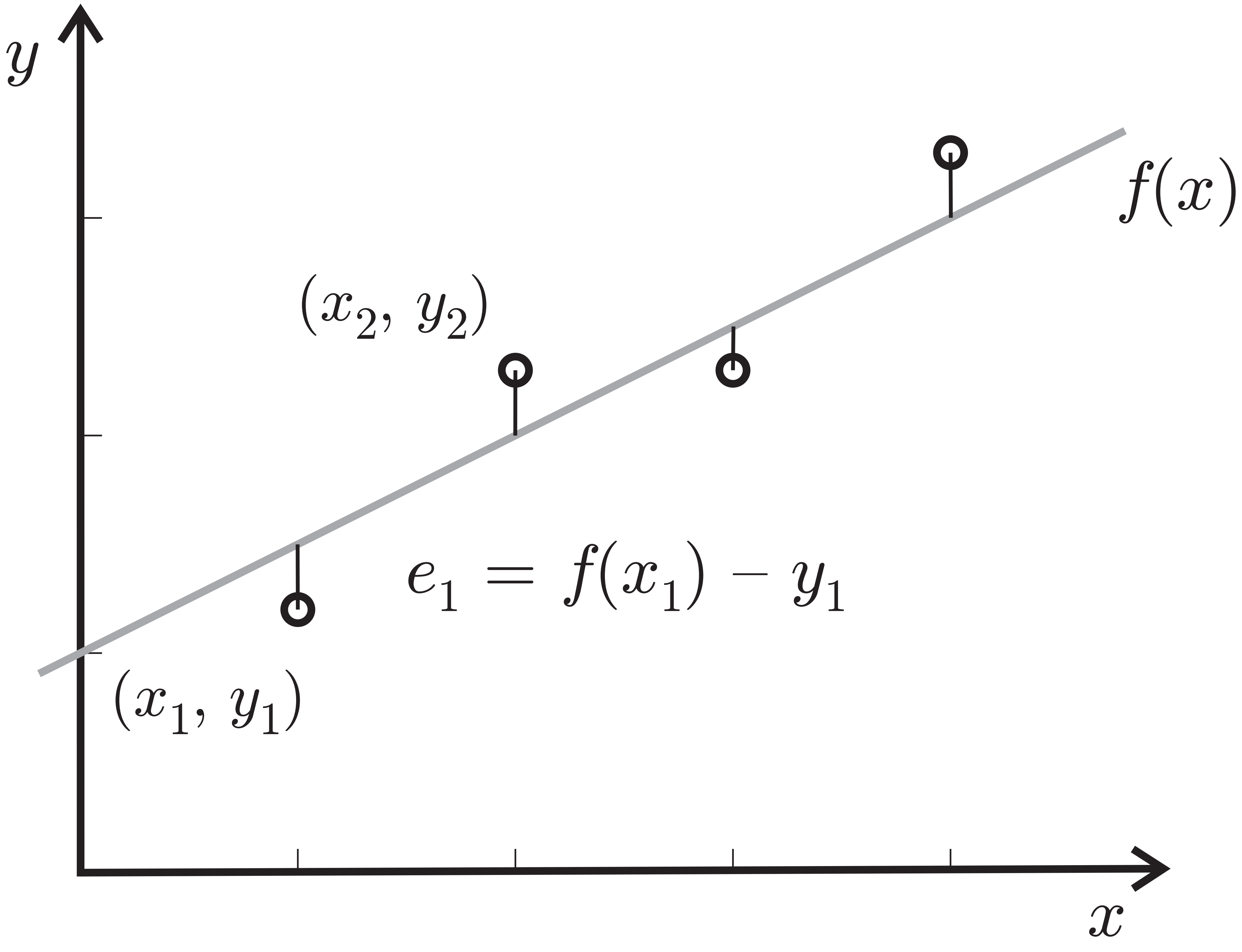
Questions:

1. Why are we **squaring** the difference?
2. Why are we **summing** up the errors?
3. What could we consider for the **function space**?

Linear function space

Definition:
A function f is a **linear function** of x if it can be written as $f(x) = w_0 + w_1 x$

$$\min_{f \text{ in linear functions}} \sum_{i=1}^{9} (f(x_i) - y_i)^2$$



Solving for the optimal function

Objective becomes:

$$\min_{f \text{ in function space}} \sum_{i=1}^{9} (f(x_i) - y_i)^2 = \min_{w_0, w_1} \sum_{i=1}^{9} (\underbrace{w_0 + w_1 x_i}_{f(x_i)} - y_i)^2$$

Questions:

1. Would you use this to predict the value of a house? Why/why not?
2. Will this predict well? How do we know?
3. What is missing to make these assessments?

Solving for the optimal function

Objective becomes:

$$\min_{f \text{ in function space}} \sum_{i=1}^{9} (f(x_i) - y_i)^2 = \min_{w_0, w_1} \sum_{i=1}^{9} (\underbrace{w_0 + w_1 x_i}_{f(x_i)} - y_i)^2$$

Questions:

1. How might you evaluate your predictor?
2. How might you improve this predictor?