Introduction

CMPUT 467: Machine Learning II

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 (MLE and MAP)
  - 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

ML II

- **Summary**: you know about the key concepts of generalization and uncertainty, as well as optimization approaches
  - These concepts are central, 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 also on generative models, not just prediction models
  - understand the key concepts in data (re)representation, that enables us to extract powerful models that generalize well
  - more advanced algorithms (constrained optimization, EM, etc.)

Course structure

- First few chapters revisit concepts from ML I, 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 and temporal data

What do I expect from you?

- I assume you have mostly forgotten details about ML (from CMPUT 267)
  - But I expect you to re-pick them up quickly
  - e.g., MLE was weird the first time you heard about it, now it will be quick to remember
- I also expect you to want to learn the mathematical underpinnings of ML
  - Or at least are willing to learn them so that you can be a better ML practitioner

Expectation Management on Content

- 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 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
  - learning how to understand derivations is key for learning new ML algorithms/ideas
  - I am not teaching you to derive new algorithms, just understand derivations

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.
- **It is better to come to class**, Zoom is only a back-up
  - And you will have to come to class at least for the 4 quizzes and 1 midterm

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
- **Office hours:** Tuesday 2 - 3 pm (on Zoom and in-person in ATH 3-05)

Teaching Assistants

Alex Ayoub
Sarosh Dandoti
Marcos Jose
Matthew Vandergrift

- **Office hours:** see eClass for times and locations+Zoom links
  - Typically question/answer sessions
- **No office hours this week**
- There is no lab, you can ask coding questions during office hours

Course Discussion

- We have created a **Discord group; please sign up!**
- I want to generate as much class discussion as possible
- Please go there first to ask questions. TAs will monitor and answer 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**

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)

Readings

- Readings are from the ML II 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 when I taught CMPUT 367
- See the schedule for sections and for reading deadlines
- Readings have an associated marked component, Reading Exercises (eClass Quizzes)

General Disclaimer

- **This course is still relatively new and not yet fully polished**
- The structure, 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

Grading

- 28%: Assignments (4)
    - Mixture of mathematical problems and programming exercises
- 12%: Reading Exercises (5)
- 25%: Midterm exam
- 35%: Final exam

Assignments

- Four assignments
- Trying something new this year: we will not mark your assignments
- Instead
  - Still have to submit the assignment (small percentage for submitting an assignment where you attempted each question)
  - An in-class quiz, testing your knowledge of the assignment (need to do the assignment to do well)
- 4% for assignment submissions, 24% for Assignment Quizzes
- One mulligan: get to drop the lowest Assignment Quiz mark (best 3 of 4)

Readings

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

Reading Exercises

- Five readings with Reading Exercises quizzes on eClass
- They are open for multiple weeks and you can complete them anytime until the deadline listed on eClass
  - Three attempts
- Provide Practice Exercises
- One mulligan: get to drop the lowest Reading Exercise mark (best 4 of 5)

Two exams

- Exams will be in-person
- Practice questions will be available
- For all exams you are allowed a **two page cheat-sheet**
    - One page, front-and-back
    - No collaborating on cheat-sheets

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.**

More on cheating

You can collaborate on assignments, but you cannot plagiarize
- We will still check your submitted assignments and mark that you completed them and check for plagiarism
- Your answers are not getting marked, so why are you plagiarizing? The assignments are for you to learn, try to do the questions yourself

In an exam, if you talk to anyone beside you or pass any objects (even an eraser), then we will assume you are cheating and take away your paper

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?”

Due Dates

- First Assignment and First Reading Exercise quiz will be released by Friday
- You can start on the First Readings now (Chapters 1 - 4)
  - Later Chapters are still being improved by me, I will ensure they are complete as soon as the previous reading exercise is due
  - If you find any typos or issues, then feel free to email me

On to the course!

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

On to the course!

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

Refresher of Basics of ML

- Goal was to learn a prediction function $f_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$ for weights $\mathbf{w}$
- Input vector of observations $\mathbf{x} \in \mathbb{R}^d$
- Outputs a prediction $\hat{y} \in \mathcal{Y}$
- If $\mathcal{Y}$ is a discrete, unordered set, then we have a **classification** problem
- If $\mathcal{Y}$ is a continuous set, then we have a **regression** problem
- (Some cases we have a discrete, ordered 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 $f^*(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y \mid \mathbf{x})$

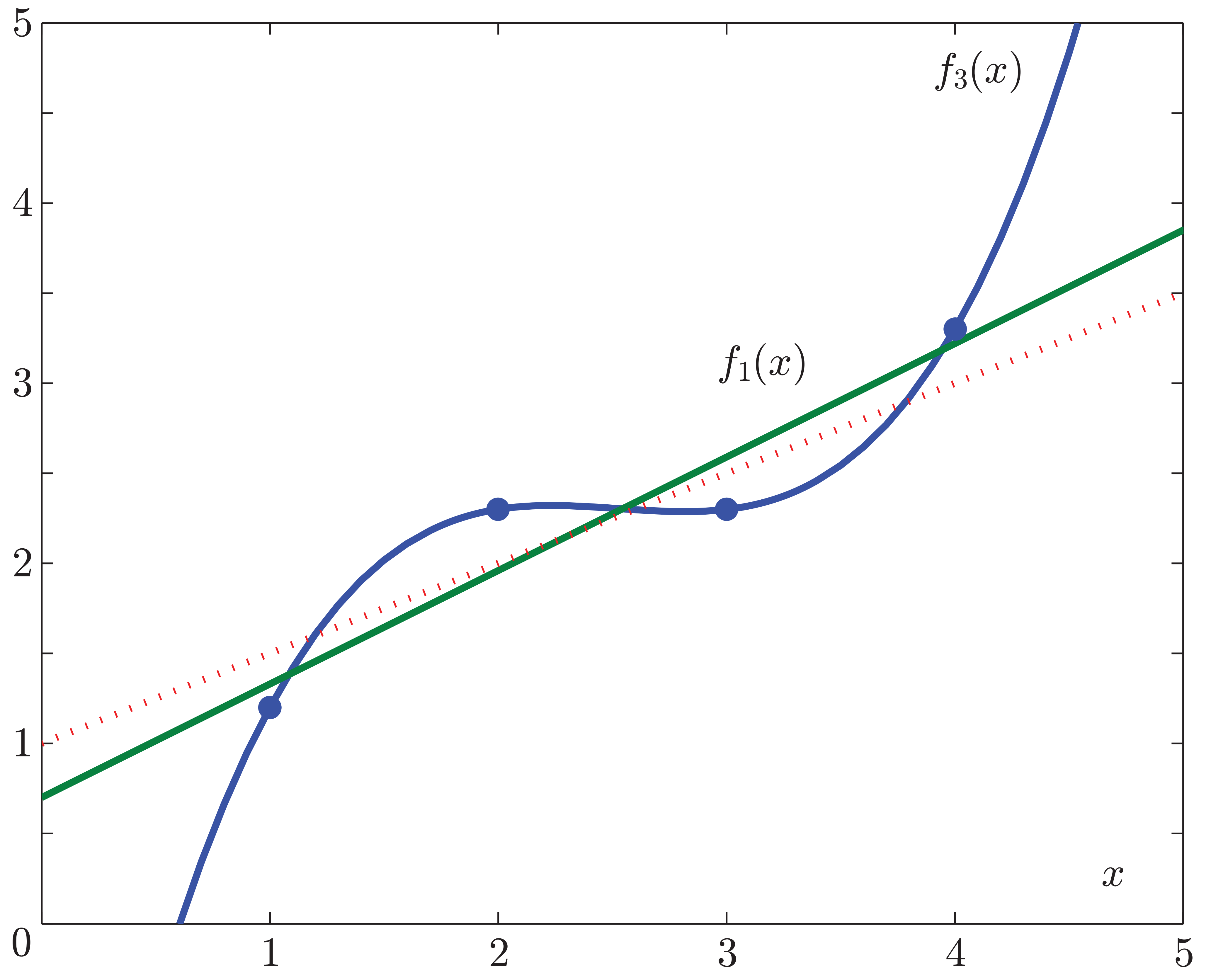
Beyond formalization, to implementation

- 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 $\max_{\theta} p(\theta \mid \mathscr{D})$ and MLE $\max_{\theta} p(\mathscr{D} \mid \theta)$

MAP and MLE for Polynomial Regression

- Let's revisit these concepts for Linear Regression and Polynomial Regression
- For regression we assume $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$, and Gaussian prior on weights
- $f_{\mathbf{w}}(\mathbf{x})$ could be a linear function (linear regression) or a polynomial function

Polynomial function is a strict generalization of linear functions



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 $\mathbf{w}$?

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 $\mathbf{w}$?
    - Same number of elements as $\boldsymbol{\phi}(\mathbf{x})$

Polynomial regression derivation (with MLE objective)

- $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$ for polynomial function $f_{\mathbf{w}}(\mathbf{x})$
- MLE objective is negative log likelihood $-\sum_{i=1}^{n} \ln p(y_i \mid \mathbf{x}_i, \mathbf{w})$
- Exercise: Show that $\arg\max_{\mathbf{w}} p(\mathscr{D} \mid \mathbf{w}) = \arg\min_{\mathbf{w}} - \sum_{i=1}^{n} \ln p(y_i \mid x_i, \mathbf{w})$

Equivalence

$$\arg\max_{\mathbf{w}} p(\mathscr{D} \mid \mathbf{w}) = \arg\max_{\mathbf{w}} p((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} p((\mathbf{x}_i, y_i) \mid \mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \ln \prod_{i=1}^{n} p((\mathbf{x}_i, y_i) \mid \mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \ln p((\mathbf{x}_i, y_i) \mid \mathbf{w})$$

ln(ab) = ln a + ln b

$$= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \ln p(y_i \mid \mathbf{x}_i, \mathbf{w}) + \ln p(\mathbf{x}_i \mid \mathbf{w})$$

p(x,y |w) = p(y|x, w) p(x|w) = p(x|y, w) p(y|w)

$$= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \ln p(y_i \mid \mathbf{x}_i, \mathbf{w})$$

p(x |w) = p(x), constant wrt w

$$= \arg\min_{\mathbf{w}} - \sum_{i=1}^{n} \ln p(y_i \mid \mathbf{x}_i, \mathbf{w})$$

maximize f or minimize -f

Polynomial regression derivation (with MLE objective)

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

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

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

Polynomial regression derivation (with MLE objective)

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

MAP Objective for Regression

- $p(y\,|\,\mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$, and Gaussian prior on weights $p(w_j) = \mathcal{N}(0, \sigma^2/\lambda)$
- 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 is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$

Contrasting MLE and MAP

- Assume $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$ with Gaussian prior $p(w_j) = \mathcal{N}(0, \sigma^2/\lambda)$
- MAP is $\max_{\mathbf{w} \in \mathbb{R}^d} p(\mathbf{w} \mid \mathcal{D})$ whereas MLE is $\max_{\mathbf{w} \in \mathbb{R}^d} p(\mathcal{D} \mid \mathbf{w})$
- MLE objective is $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ vs MAP: $\sum_{i=1}^{n} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$
- MLE sets $\lambda = 0$, like having a uniform prior on each weight
  - But not equivalent, since can't have zero variance Gaussian

l2-regularized polynomial regression

- MAP adds prior information, to help prevent **overfitting**
- The l2 regularizer prefers simpler solutions, those where weights do not deviate too far from zero
  - We will revisit why, now with matrices
- Do you think l2 regularization is more useful
  - (Q1) for polynomial regression than for linear regression?
  - (Q2) for high-degree polynomials rather than low-degree ones?

Why did we go to all this trouble?

- We could have just jumped to the squared cost and add an l2 regularizer, without talking about MAP and MLE
- Main reason: when generalizing to other settings, MAP and MLE are very useful
  - we need these tools, so may as well do a unified treatment and get used to them
- Minor reason: it does provide some probabilistic insight into l2 regularization
- **Exercise**: What if we want to learn the conditional mean and variance of the data? Do we just used the squared loss for both? Derive the MLE objective if we assumed $p(y \mid \mathbf{x}) = \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}), g_{\boldsymbol{\theta}}^2(\mathbf{x}))$

MLE Objective for Classification

- For binary classification with logistic regression, the MLE objective was the cross-entropy objective
- We will revisit this, when we talk about Generalized Linear Models and Multinomial Logistic 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
    - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
    - $f_{\mathbf{w}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$
- 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
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1 x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$
- How do we evaluate $f_{\mathbf{w}}$ on the test set? (What is the formula)
- $\frac{1}{m} \sum_{(\mathbf{x}_i, y_i) \in \mathscr{D}_{\text{test}}} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ 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
  - $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, \ldots, x_d, x_1x_2, \ldots, x_d^3]$
  - $f_{\mathbf{w}}(\mathbf{x}) = 1$ 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?)

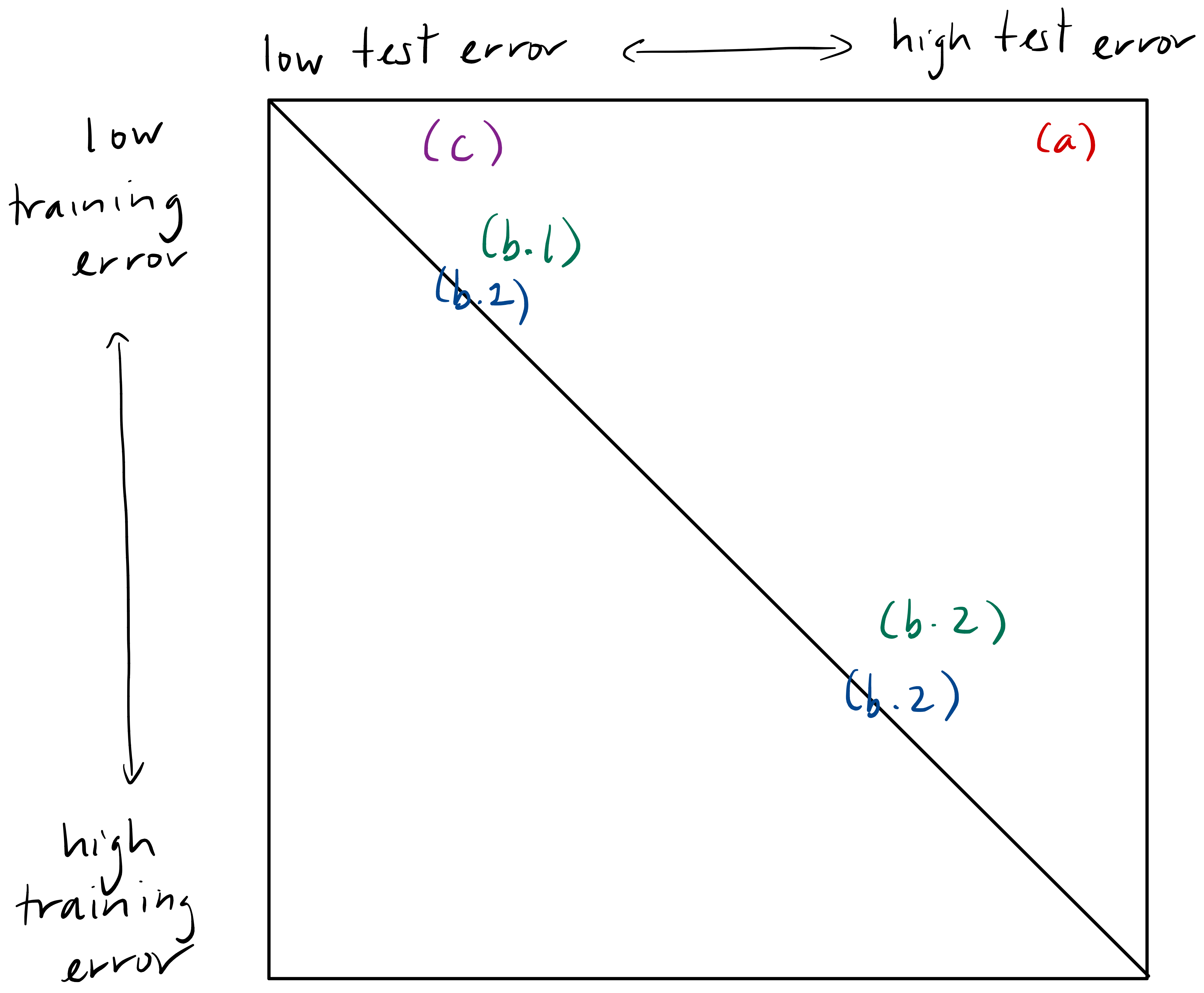
Exercise

- 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?)
- $\frac{1}{m} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{test}}} 1(f_{\mathbf{w}}(\mathbf{x}_i) \neq y_i)$ for m the number of test samples

Conceptually Reasoning aout which models to select

- We can empirically evaluate to select models but we also 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. $\mathcal{F}$ = 8th 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

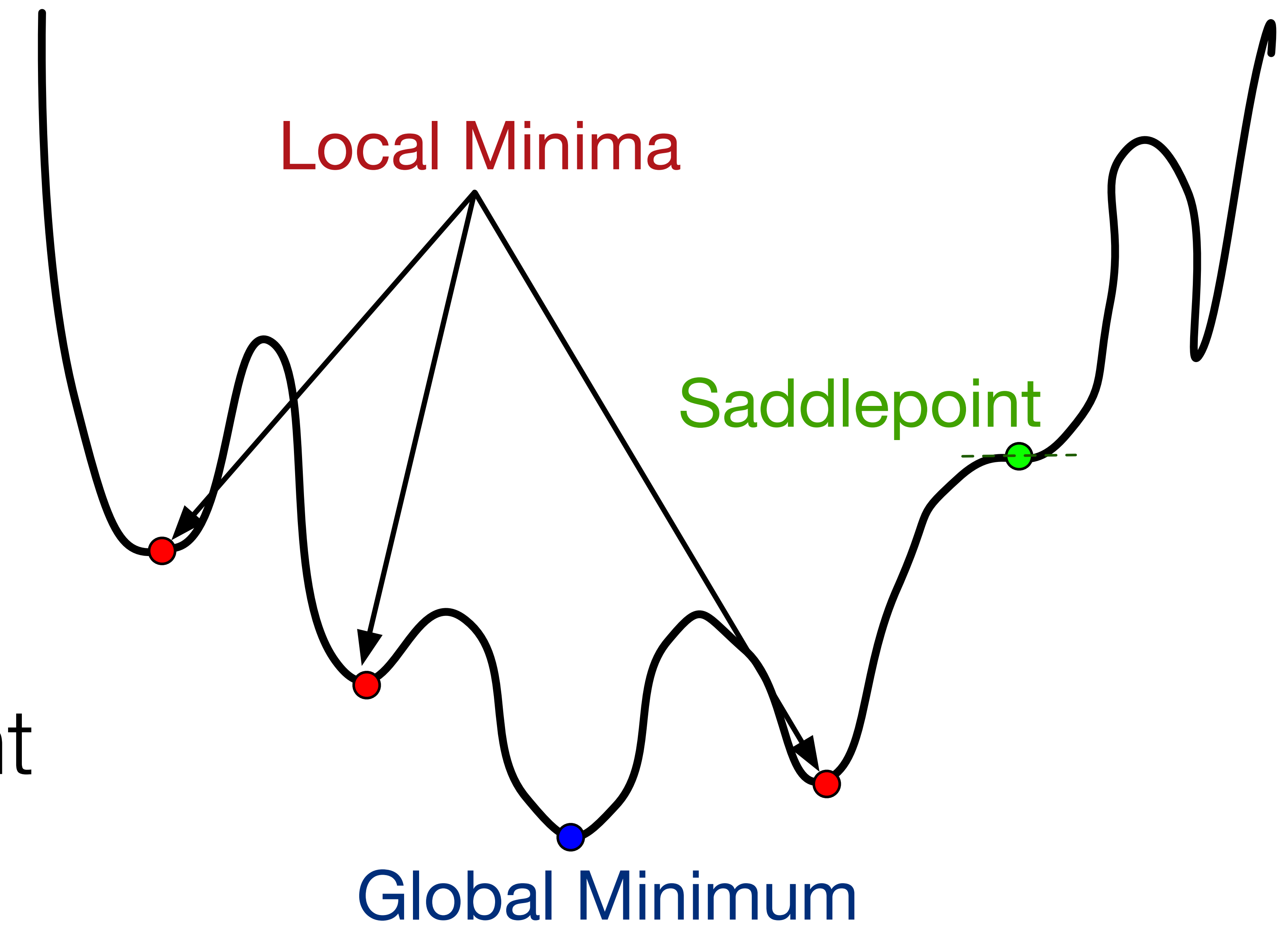
- Confidence intervals to assess uncertainty in a mean estimator
  - obtained using distributional assumptions like the Student-t and with less assumptions using concentration inequalities
  - also discussed using Bayesian approach to get credible interval
- Bayesian methods obtain $p(\mathbf{w} \mid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

Probability and Optimization at the Core

- Throughout used optimization tools, to have practical algorithms 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)
    - the role of stepsize selection

Fun Fact about Saddlepoints

- Are local minima or saddlepoints worse?
- 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



On to the course!

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

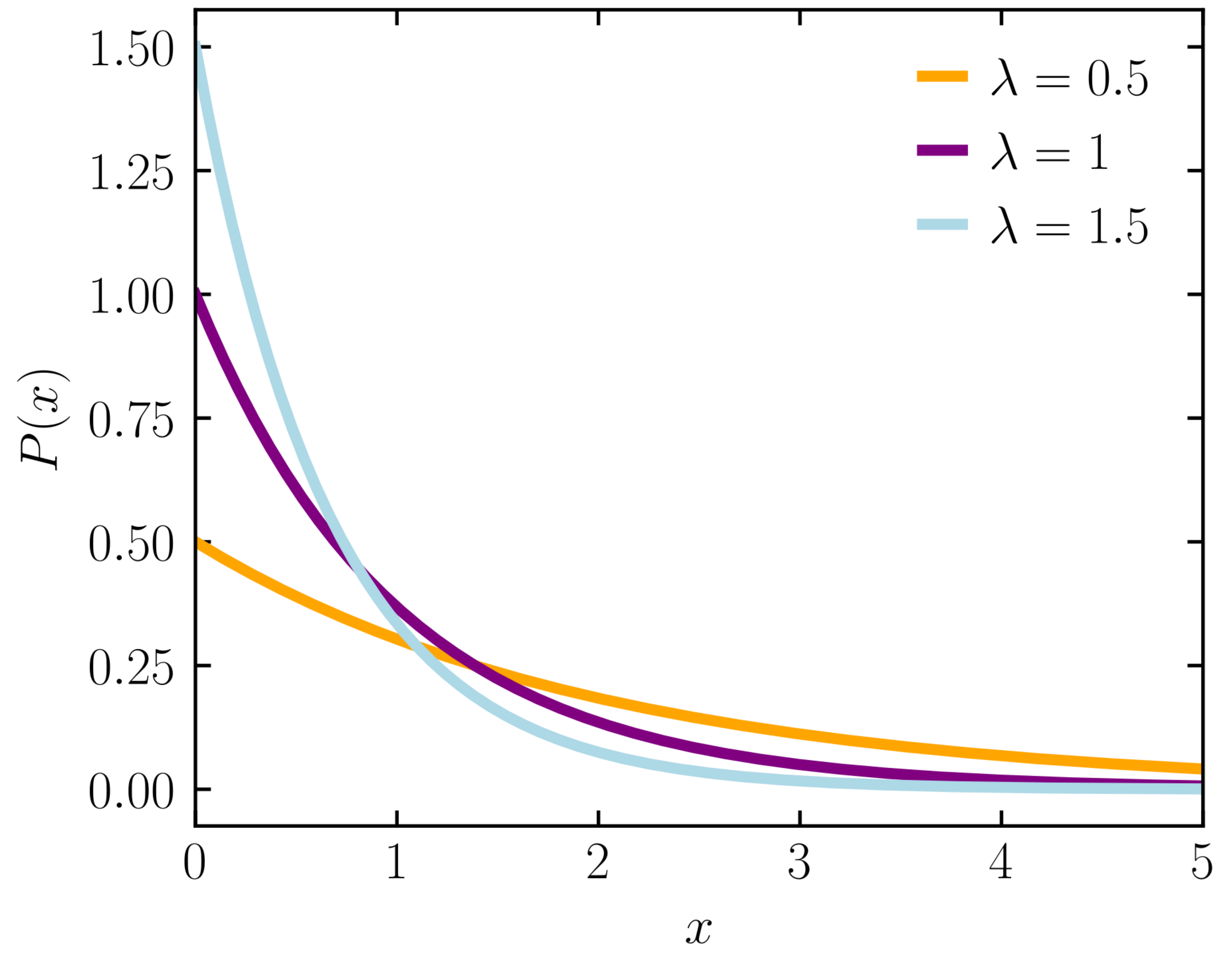
Prediction Models and Generative Models

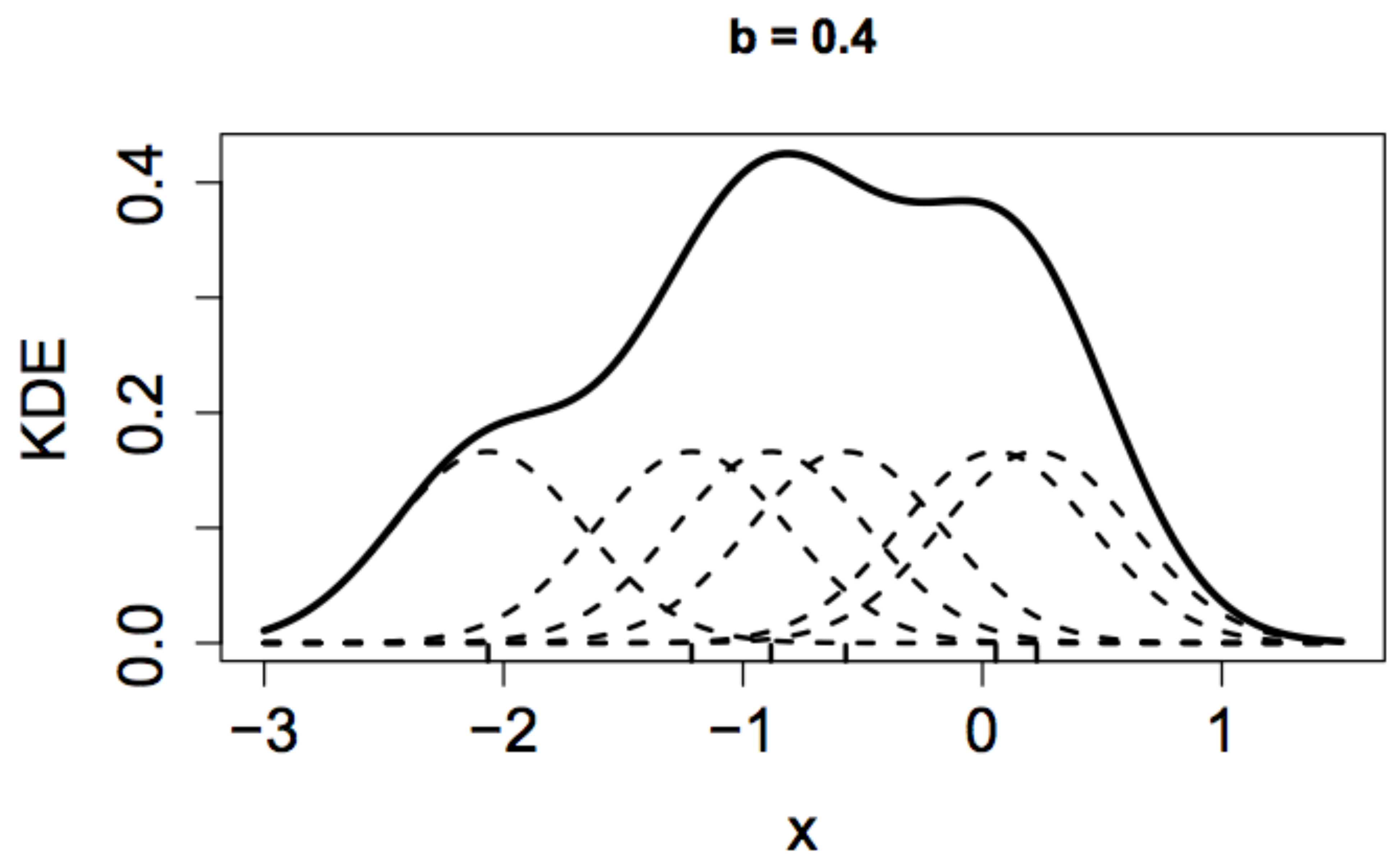
- We looked at both learning $p(x)$ and $p(y \mid x)$
- We usually think of $p(x)$ as a generative model and learn $p(y \mid x)$ for prediction (using $\mathbb{E}[Y \mid x]$ for regression and $p(y \mid x)$ for classification)
- This distinction is not quite right. Rather, key is how we use these models
- **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

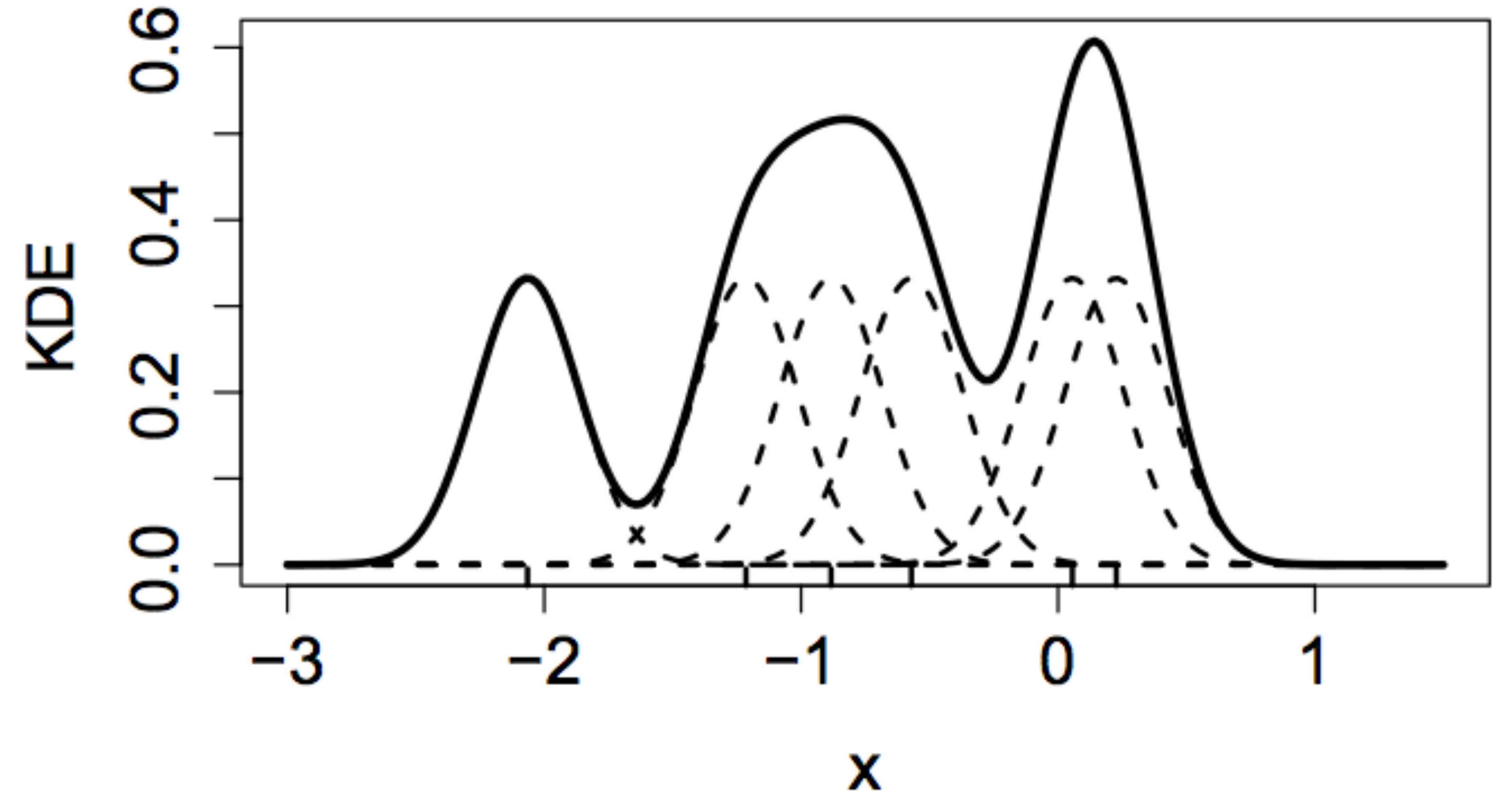
Examples

- 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
  - for each case we still learn a (conditional) distribution

What is a complex distribution?







More simple $\longrightarrow$ 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
- ML II 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

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

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

Matrix-matrix product

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

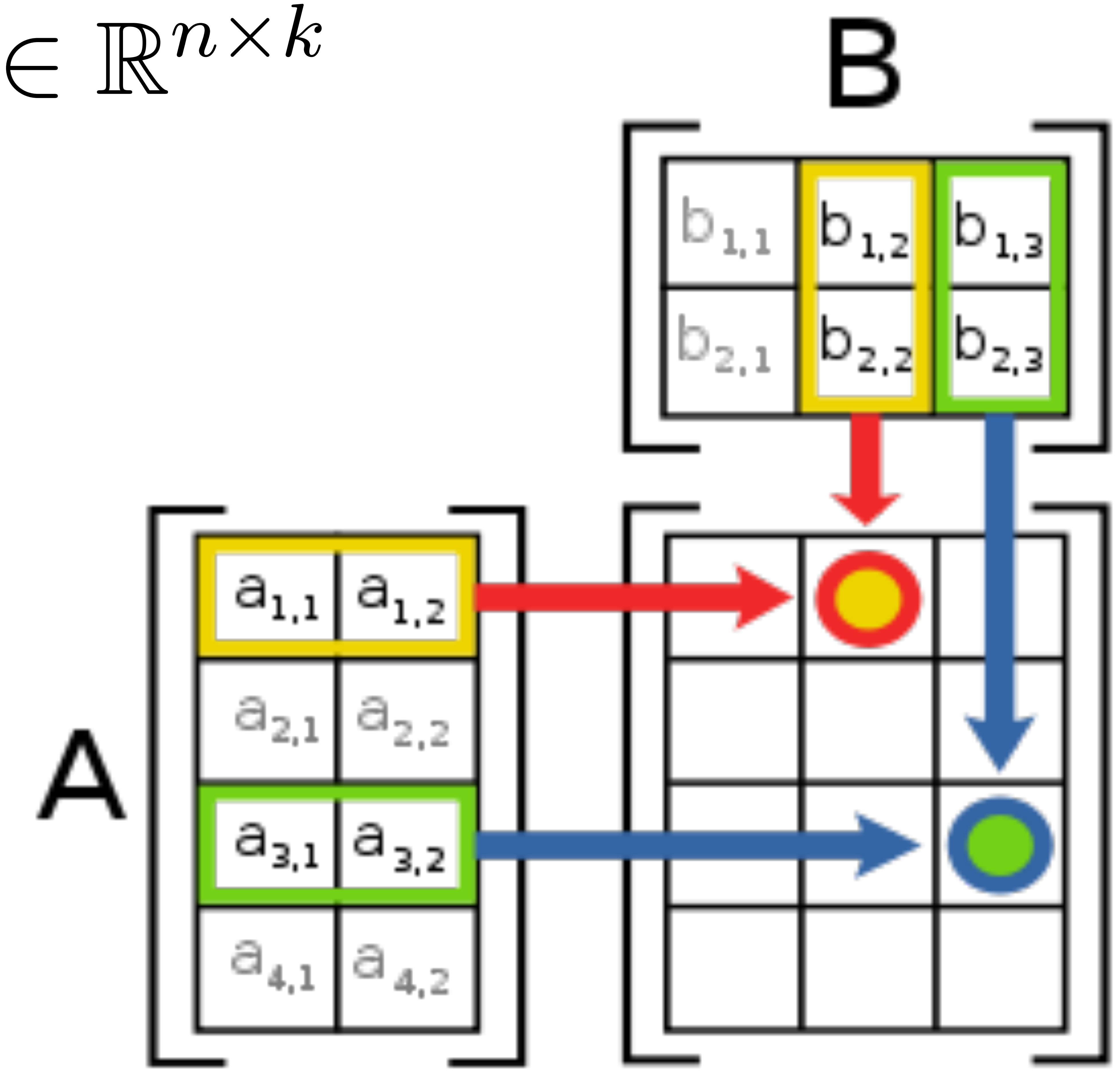
$$\mathbf{AB} = [\mathbf{AB}_{:,1}, \mathbf{AB}_{:,2}, \ldots, \mathbf{AB}_{:,k}] = \begin{bmatrix} \mathbf{A}_{1,:}\mathbf{B}_{:,1} & \mathbf{A}_{1,:}\mathbf{B}_{:,2} & \ldots & \mathbf{A}_{1,:}\mathbf{B}_{:,k} \\ \mathbf{A}_{2,:}\mathbf{B}_{:,1} & \mathbf{A}_{2,:}\mathbf{B}_{:,2} & \ldots & \mathbf{A}_{2,:}\mathbf{B}_{:,k}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

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

$$\mathbf{I} \overset{\text{def}}{=} \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \dots & & & \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

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

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

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

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?

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

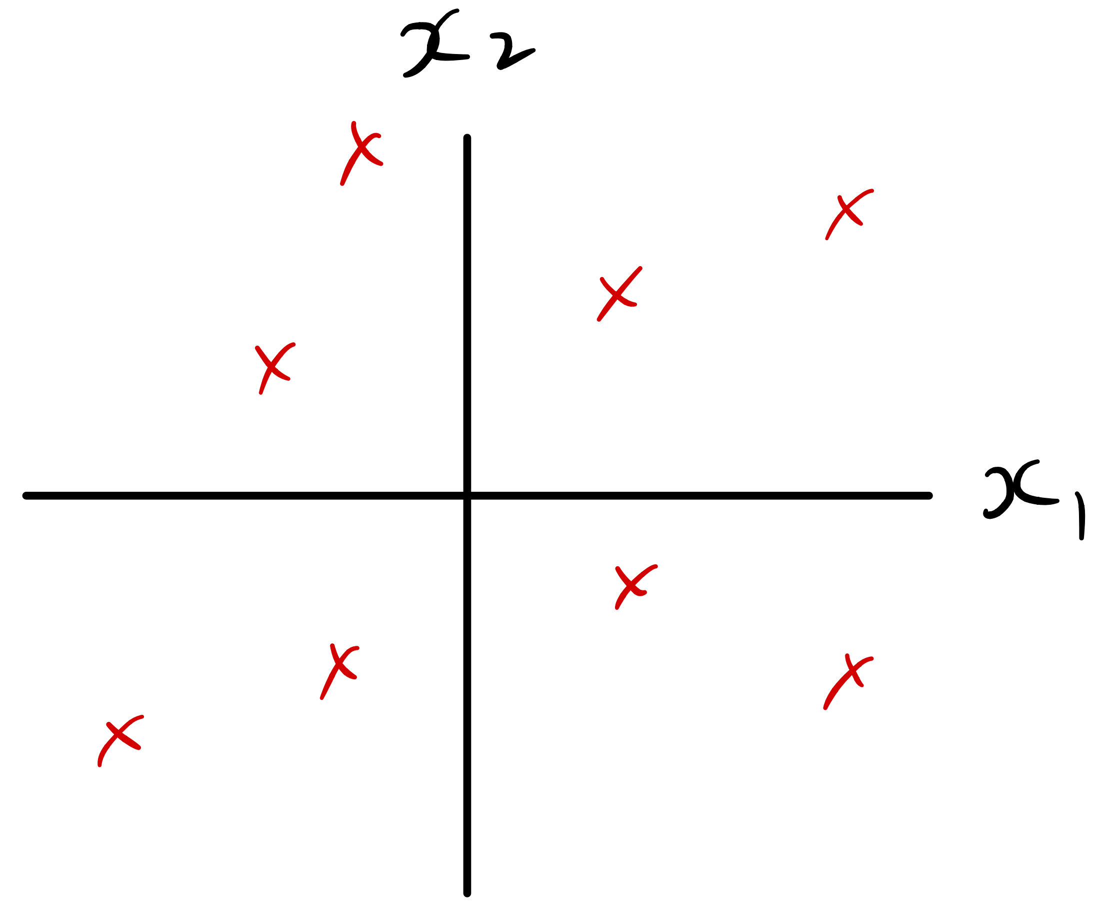
- Every matrix is a linear operator that can be decomposed into
a rotation (V), scaling (Sigma), and rotation (U) operation
- What does the scaling operation do?

- 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

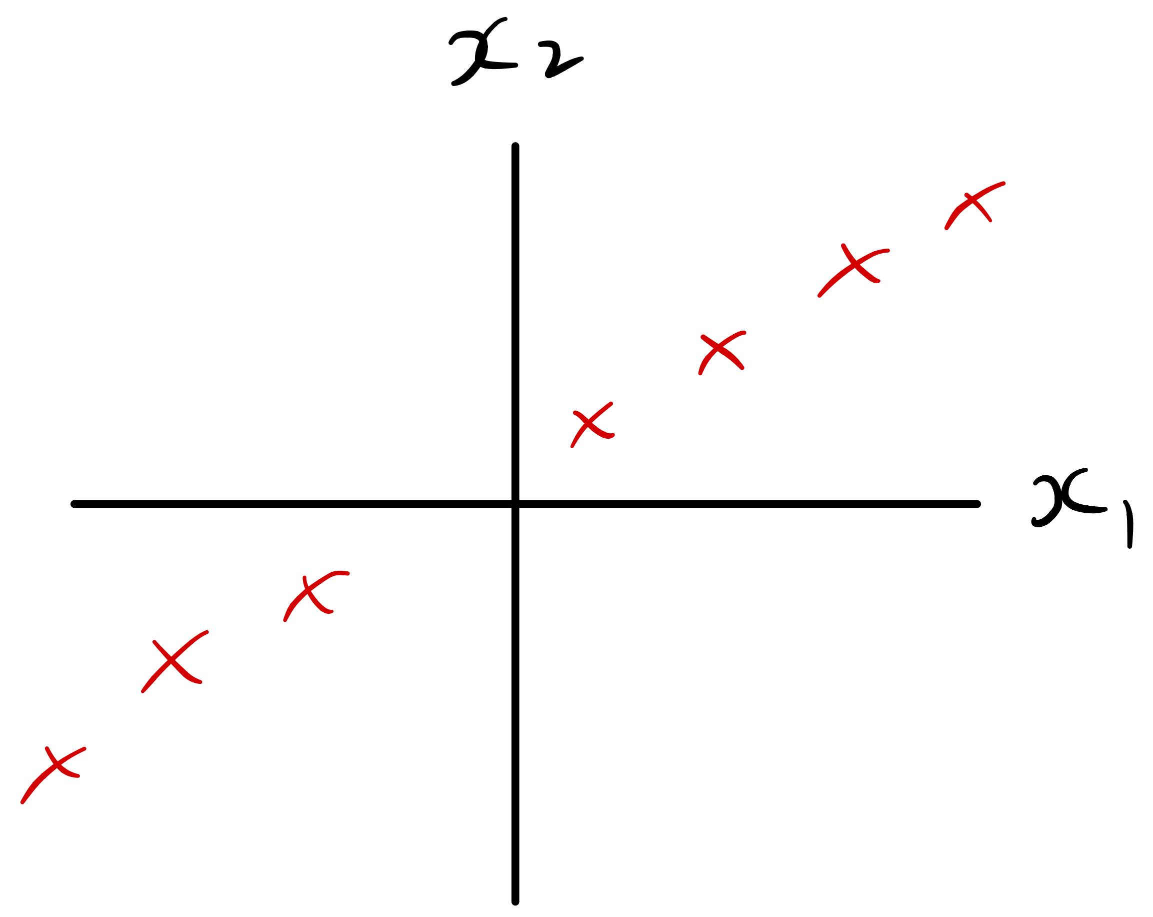
- $\mathbf{X} \in \mathbb{R}^{n \times d}$ for n samples and d features
- Let's imagine d = 2
- $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top = [\mathbf{u}_1, \mathbf{u}_2]\text{diag}(\sigma_1, \sigma_2)[\mathbf{v}_1^\top; \mathbf{v}_2^\top] = \sum_{j=1}^{2} \sigma_j \mathbf{u}_j \mathbf{v}_j^\top$
- 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

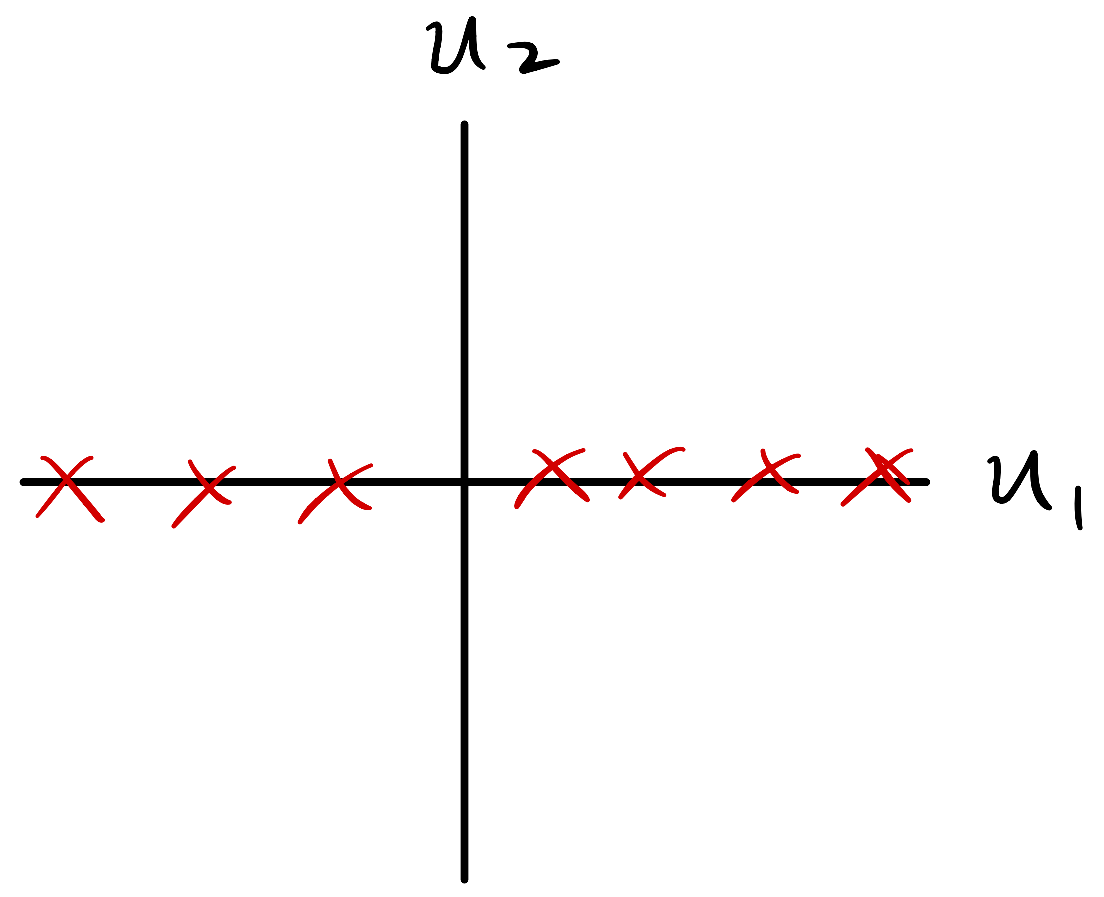
$X = [u_1\ u_2]\begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}[v_1^T; v_2^T]$



$\sigma_2 = 0$ for X

$$\xrightarrow{\text{Rotate}}$$

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



All points only vary in one dimension

Notice neither x1 nor x2 are always zero,
features look like they vary in both dimensions

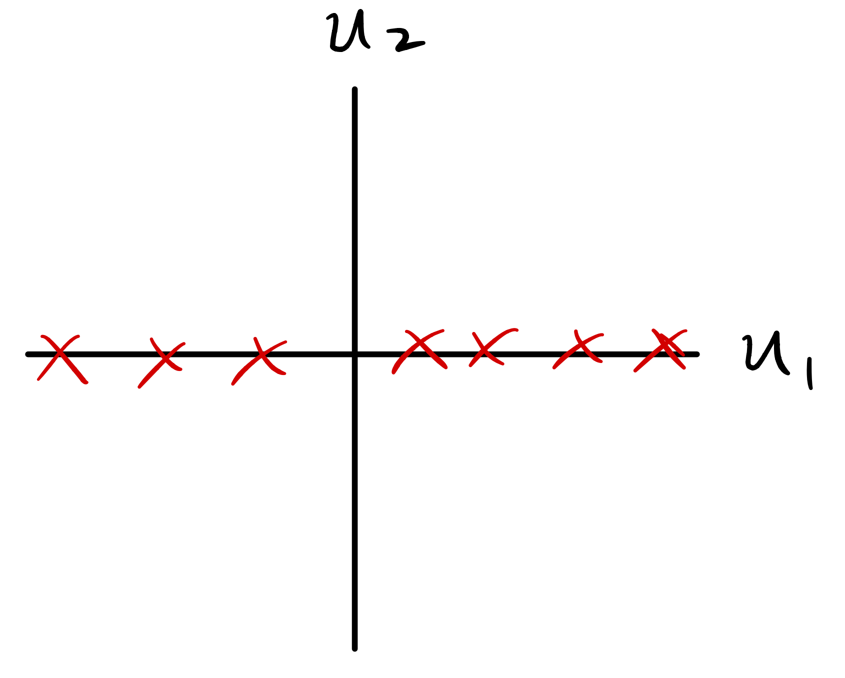
The effect on predictions

Rotate

$\xrightarrow{\hspace{2em}}$

$\hat{X} = XV$

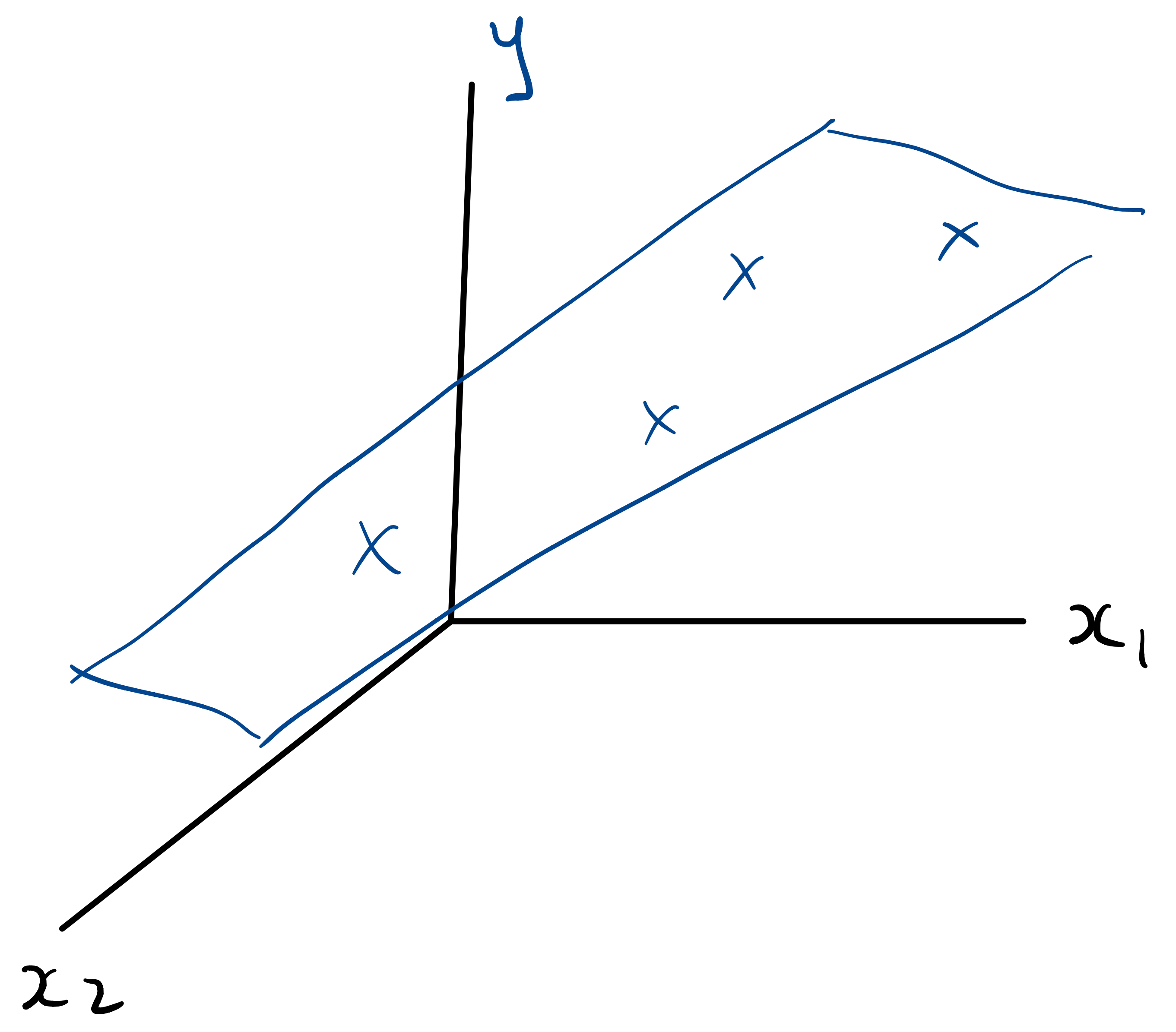
$= \sigma_1 u_1$



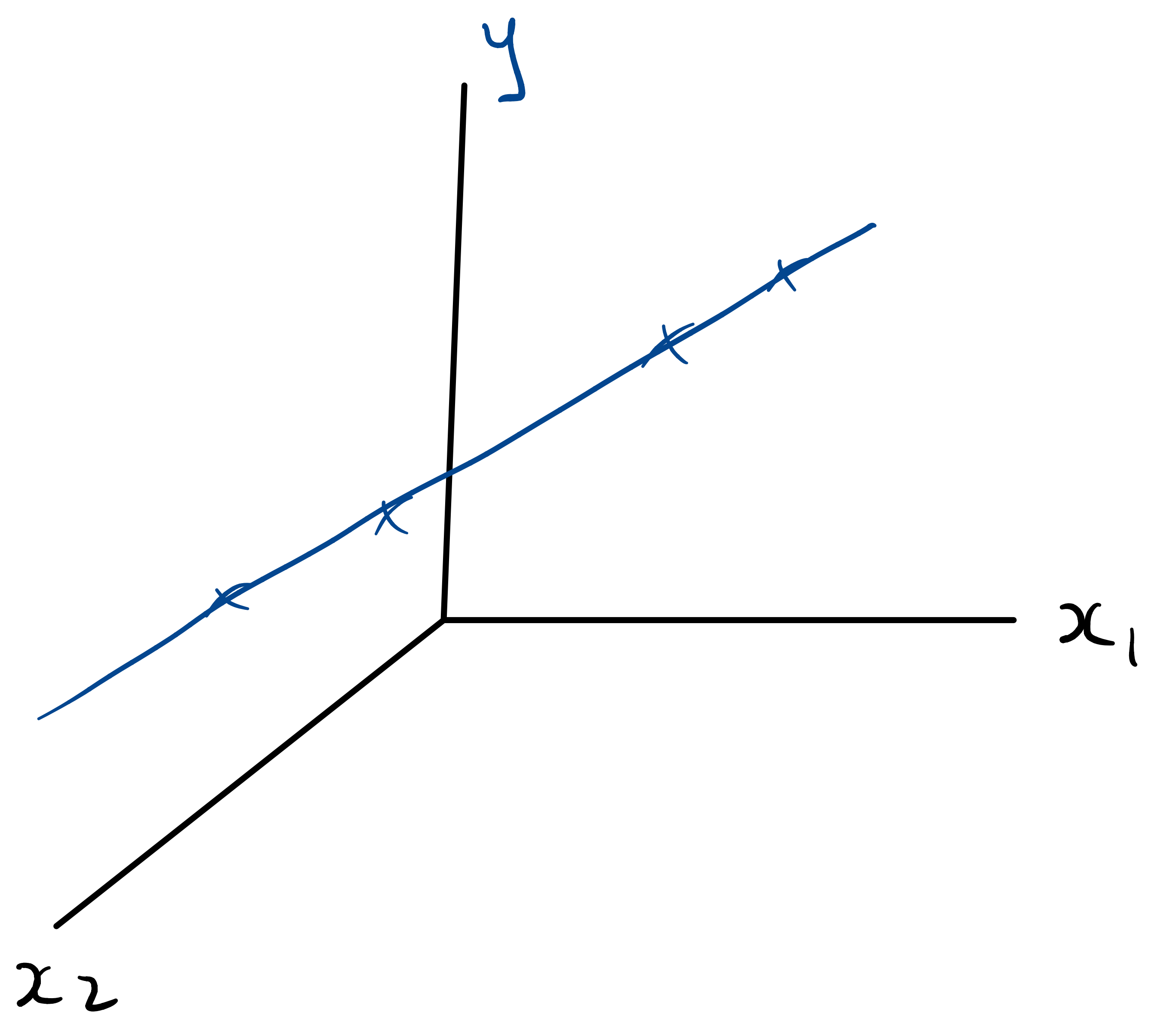
All points only vary in one dimension

- $\hat{y} = \mathbf{X}\mathbf{w} = \tilde{\mathbf{X}}\tilde{\mathbf{w}}$ 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 that it projects vectors to
- For a matrix with one singular value that is zero, it projects all vectors to one dimension lower (a plane in dimension n-1 inside the large n-dimensional space)

Class Poll

- Should I change the inputs to be row vectors?
- Reasoning: $\mathbf{X} \in \mathbb{R}^{n \times d}$ has each sample as a row, $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$
- When we write $\mathbf{X}\mathbf{w}$ for $\mathbf{w} \in \mathbb{R}^{d \times 1}$, we are taking each row and computing $\mathbf{x}_i\mathbf{w}$
- Usually we assume $\mathbf{x}_i \in \mathbb{R}^{d \times 1}$ so we write $\mathbf{x}_i^\top \mathbf{w}$ and say the rows of $\mathbf{X}$ consist of $\mathbf{x}_i^\top$
- But when we move to NNs and learning representations, convenient to treat $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$as a row vector and write $\mathbf{x}_i\mathbf{W}$
  - Other plus: the minimal ink principles, it removes a bunch of transposes
- **Issue: is this going to confuse you a lot?** To see $\mathbf{x}_i\mathbf{w}$ instead of $\mathbf{x}_i^\top \mathbf{w}$

Eigenvalue Decomposition

- A square symmetric matrix $\mathbf{M} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$
  - $\boldsymbol{\Lambda}$ is a diagonal matrix
  - Notice this like an SVD, where the second rotation is $\mathbf{U}$ again
- Computing the inverse is now easy: $\mathbf{M}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^\top$
- 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 $\mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$