Final Exam Topics

CMPUT 267: Basics of Machine Learning

Chapters 1 - 12

Goal of these Slides

- Highlight key concepts to be tested
- Additionally highlight what I will not test
  - It is in the notes for your knowledge, but hard to directly test
- Do a couple of exercises along the way

Probability

- Understand the following concepts
  - **random variables**
  - **joint** and **conditional probabilities** for continuous and discrete random variables
  - **probability mass functions** and **probability density functions**
  - **independence** and conditional independence
  - **expectations** for continuous and discrete random variables
  - **variance** for continuous and discrete random variables

Probability (2)

- Know how to represent a problem probabilistically
- Use a provided distribution
  - I will always remind you of the density expression for a given distribution
- Apply **Bayes' Rule** to derive probabilities
- **Will not be directly tested:**
  - I will not expect you to know specific pdf and pmfs

Estimators

- Understand the following concepts
  - **estimators**
  - **bias**
  - **consistency**
  - how to show that an estimator is/is not biased
  - how to derive an expression for the variance of an estimator
  - how to show that an estimator is/is not consistent
  - when the use of a **biased estimator** is **preferable**

Estimators (2)

- Apply **concentration inequalities** to derive **confidence bounds**
- Define **sample complexity**
- Understand how concentration inequalities can be used to characterize the sample complexity of an estimator
- Explain when a given concentration inequality can/cannot be used
- **Will not be directly tested**
  - You do not need to know concentration inequality formulas

Estimators (3)

- Understand the **sample average estimator** and its properties
  - unbiased estimator, characterize variance
- Understand the **maximum likelihood estimator** (MLE)
- Understand the **MAP estimator**, and contrast to MLE
- **Will not be directly tested**
  - You will not need to derive parameters for MLE and MAP on the exam

Estimators (4)

- Understand that MAP and MLE are **point estimates**, and the **Bayesian** estimator maintains the full posterior p(w | D)
- Understand the role of **conjugate priors** priors
- **Will not be directly tested**
  - Do not need to know specific conjugate priors

Optimization

- Represent a problem as an optimization problem
- Solve an optimization problem by finding **stationary points**
- Define **first-order gradient descent**
- Define **second-order gradient descent**
- Define **step size** and **adaptive step size**
- Explain the role and importance of step sizes in first-order gradient descent
- **Will not be directly tested**
  - Specific stepsize adaptation algorithms

Stochastic gradient descent

- If $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$, then we can be more computationally efficient by using a stochastic approximation to the gradient on each step
- Each update consists of taking a mini-batch $\mathscr{B}$ and updating with
- $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \frac{1}{b} \sum_{i \in \mathscr{B}} \nabla c_i(\mathbf{w}_t)$
- Contrast to **gradient descent** update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \frac{1}{n} \sum_{i=1}^{n} \nabla c_i(\mathbf{w}_t)$
- Question: why is it called **stochastic** gradient descent?

Prediction

- Describe the differences between **regression** and **classification**
- Understand the **optimal classification predictor** for a given **cost**
- Understand the **optimal regression predictor** for a given cost
- Describe the difference between **irreducible** and **reducible error**
- **Will not be directly tested**
  - Deriving optimal predictors
  - Multi-label vs multi-class classification

Linear Regression

- Understand that we assume $p(y \mid \mathbf{x})$ is Gaussian and that the resulting MLE objective corresponds to the sum of squared errors $\sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2$
- Understand the computational cost of the **gradient descent** and **stochastic gradient descent** solutions to linear regression
- Represent a **polynomial regression** problem as linear regression
- **Will not be directly tested**
    - Do not need to know the closed-form solution with matrices

Generalization Error

- Describe the difference between **test error** and **generalization error**
- Explain why **training error** is a **biased estimator** of generalization error
- Describe how to **estimate generalization** error given a dataset
- Understand that we can use **statistical significance tests** to compare two models
- **Will not be directly tested**
    - Specific statistical significance tests

Regularization

- Understand that regularization constrains the solutions to mitigate overfitting
- Understand that L2-regularized linear regression is the **MAP objective with a Gaussian prior**
- Describe the effects of the **regularization hyperparameter** λ
- Understand that l1 regularization does feature selection
- **Will not be directly tested**
  - The Laplace distribution
  - Deriving the MAP solution

Bias-Variance Tradeoff

- Explain the implications of the **bias-variance decomposition** for estimators
- Describe the advantages and disadvantages of the MAP estimator for linear regression (Gaussian prior)
- Explain how the choice of **hypothesis class** can affect the bias and variance of **predictions**
- **Will not be directly tested**
    - Do not need to know the bias and variance formulas of the MLE and MAP estimators for linear regression

Bias of MAP goes to zero with more samples

- The objective was $\sum_{i=1}^{n} \frac{1}{2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2$ or equivalently

$$c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2$$

- i.e, $c_i(\mathbf{w}) = \frac{1}{2}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2n}\sum_{j=1}^{d} w_j^2$ for $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$

- We scale the weight on the penalty by $\frac{1}{n}$, which goes to zero as n gets big

Question

- Which of the following estimators have higher bias and variance?
- MLE for linear regression ($\lambda = 0$)
- MAP for linear regression with $\lambda = 0.1$
- MAP for linear regression with $\lambda = 0.5$

Logistic Regression

- Define linear classifier, sigmoid function, logistic regression
- Understand that we parameterize $p(y=1\,|\,\mathbf{x}) \approx \sigma(\mathbf{x}^\top \mathbf{w})$
- Understand that the objective (cross-entropy) and update underlying logistic regression is different from linear regression
- Understand that we estimate $p(y=1\,|\,\mathbf{x})$, and predict $\arg\max_{y\in\{0,1\}} p(y\,|\,x)$
- Understand that we can extend logistic regression just like linear regression, to use polynomials, l2 regularization and l1 regularization
- **Will not be directly tested**
    - You do not need to memorize the logistic regression update

Bayesian linear regression

- Understand the goals of obtaining $p(\mathbf{w} \mid \mathscr{D})$ for regression
- For the scalar case: Know that $p(w)$ is Gaussian, the conjugate prior for Gaussian likelihood $p(y \mid x) = \mathcal{N}(xw, \sigma^2)$, giving Gaussian posterior $p(w \mid \mathscr{D})$
- More generally, for vector case, know that the Gaussian prior is also the conjugate prior for the Gaussian likelihood
- Understand that having $p(\mathbf{w} \mid \mathscr{D})$ lets us reason about uncertainty in our predictions $p(\mathbf{x}^\top \mathbf{w} \mid \mathscr{D})$
- **Will not be directly tested**
- You do not need to know the explicit formula for the posteriors

Question

- Assume we are in the realizable setting, namely we are learning a linear function with no intercept and the true function is linear with no intercept
  - So the true $p(y\,|\,x) = \mathcal{N}(xw^*, \sigma^2)$
- Assume we have the posterior $p(w\,|\,\mathcal{D}) = \mathcal{N}(\mu_n, \sigma_n^2)$ (and we also had a Gaussian prior $p(w) = \mathcal{N}(0,\sigma_0^2)$)
- A 95% credible interval is $[\mu_n - \epsilon, \mu_n + \epsilon]$ for $\epsilon = 1.96\sigma_n$
- Does this mean $w^* \in [\mu_n - \epsilon, \mu_n + \epsilon]$, with 95% probability? (Hint: consider if the answer changes for big or small σ_0^2)

Some Case Studies

- AKA how does anything we learned here connect to the real world?
- (And obviously none of this will be tested)

Historical Example:
US Postal Service (1990)

- Problem: automatically sort mail based on destination, by reading the handwritten zip code on the envelopes
- Strategy:
- 1. Snap a picture of the envelope front
- 2. Segment the image, extracting first the zip code and then each digit in the zip code
- 3. Input the segmented digit x into the classifier f(x) to get a prediction of the class from {0,1,2,3,4,5,6,7,8,9}

Step 3 is what we are doing

- The input x is a non-color image, with entries either 0 or 1 representing a black pixel (writing, dirt) and 0 representing a white pixel (no writing)
- The image is 2d, but can be flattened into a vector input
  - e.g., 30x30 pixel image becomes a vector of size 900 (d = 900)
- Our goal is to learn p(y | x) so that we can predict
- $$f(\mathbf{x}) = \arg \max_{y \in \{0,1,\ldots,9\}} p(y \mid \mathbf{x})$$

Multi-class Classification

- Need to use multinomial logistic regression instead of logistic regression
- Idea is very similar. Learn weights $\mathbf{w}_k$ for each class to predict
- $\hat{p}(y = k \mid \mathbf{x}) \propto \sigma(\mathbf{x}^\top \mathbf{w}_k)$
- Pick the class k where $\sigma(\mathbf{x}^\top \mathbf{w}_k)$ is the highest
- Small nuance: we normalize predictions so that $\sum_{y \in \{0,1,\ldots,9\}} \hat{p}(y \mid \mathbf{x}) = 1$

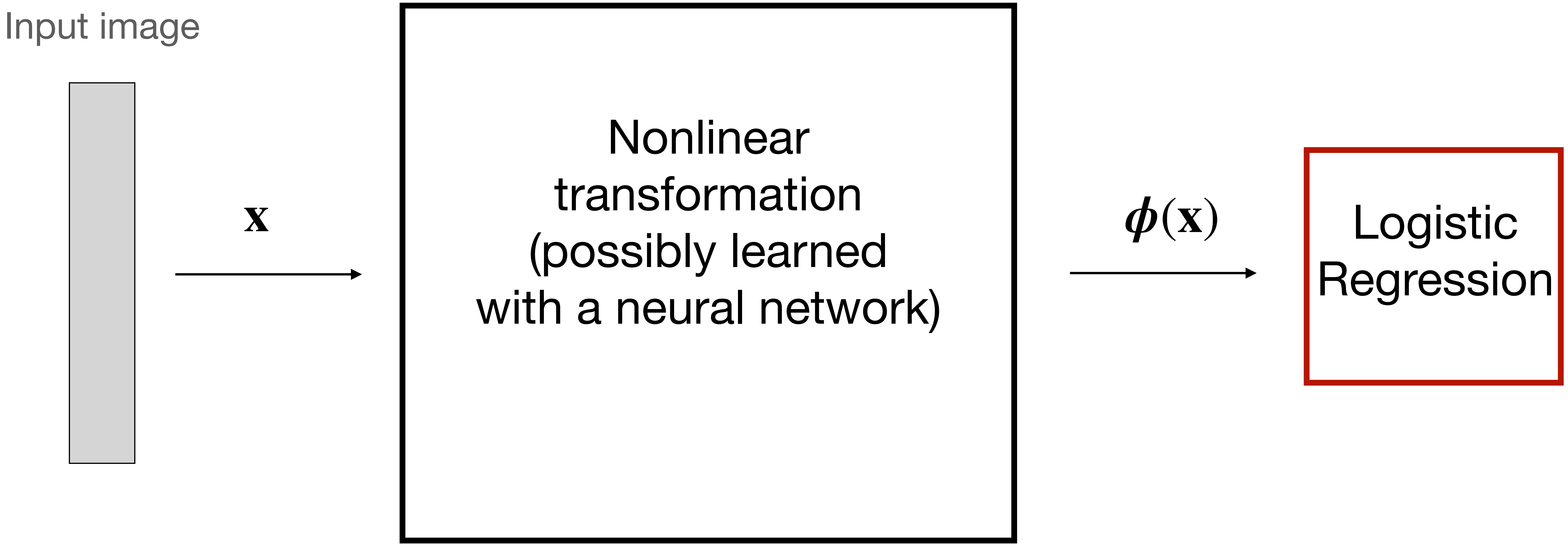
Moving from linear to nonlinear

- Is it sensible to learn a linear function of the image?
- What is an alternative? Do polynomials make sense here?

Nonlinearity beyond polynomials

- The general concept behind polynomial regression is that we
  - mapped $\mathbf{x}$ to a new set of features $\boldsymbol{\phi}(\mathbf{x})$
  - learning a linear function on $\boldsymbol{\phi}(\mathbf{x})$ gives us a nonlinear function on $\mathbf{x}$
- This general concept can be applied with many nonlinear functions, not just polynomials
- Other examples: radial basis functions, Fourier basis, wavelets, neural networks

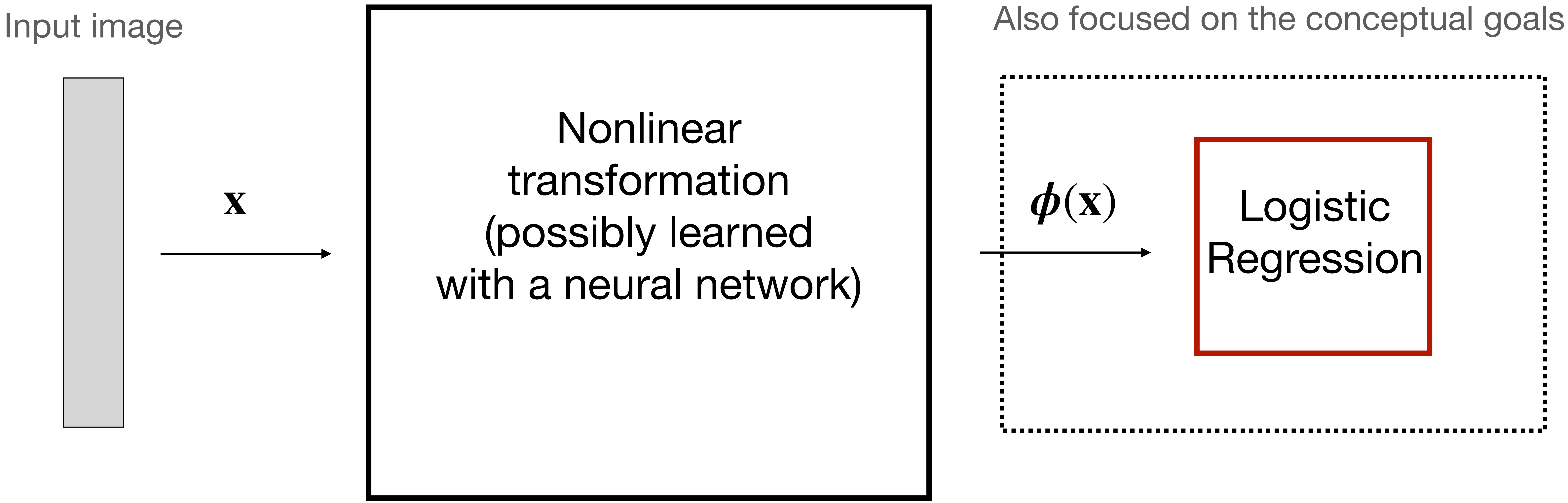
General idea



General idea

This course was focused on the underlying probabilistic concepts for this part, which stays the same for more complex models

Also focused on the conceptual goals for $\phi(x)$



General idea

A huge part of machine learning
is about how to get these
nonlinear transformations
(up next in future ML courses)

Input image

$\xrightarrow{\mathbf{x}}$

Nonlinear
transformation
(possibly learned
with a neural network)

$\xrightarrow{\boldsymbol{\phi}(\mathbf{x})}$

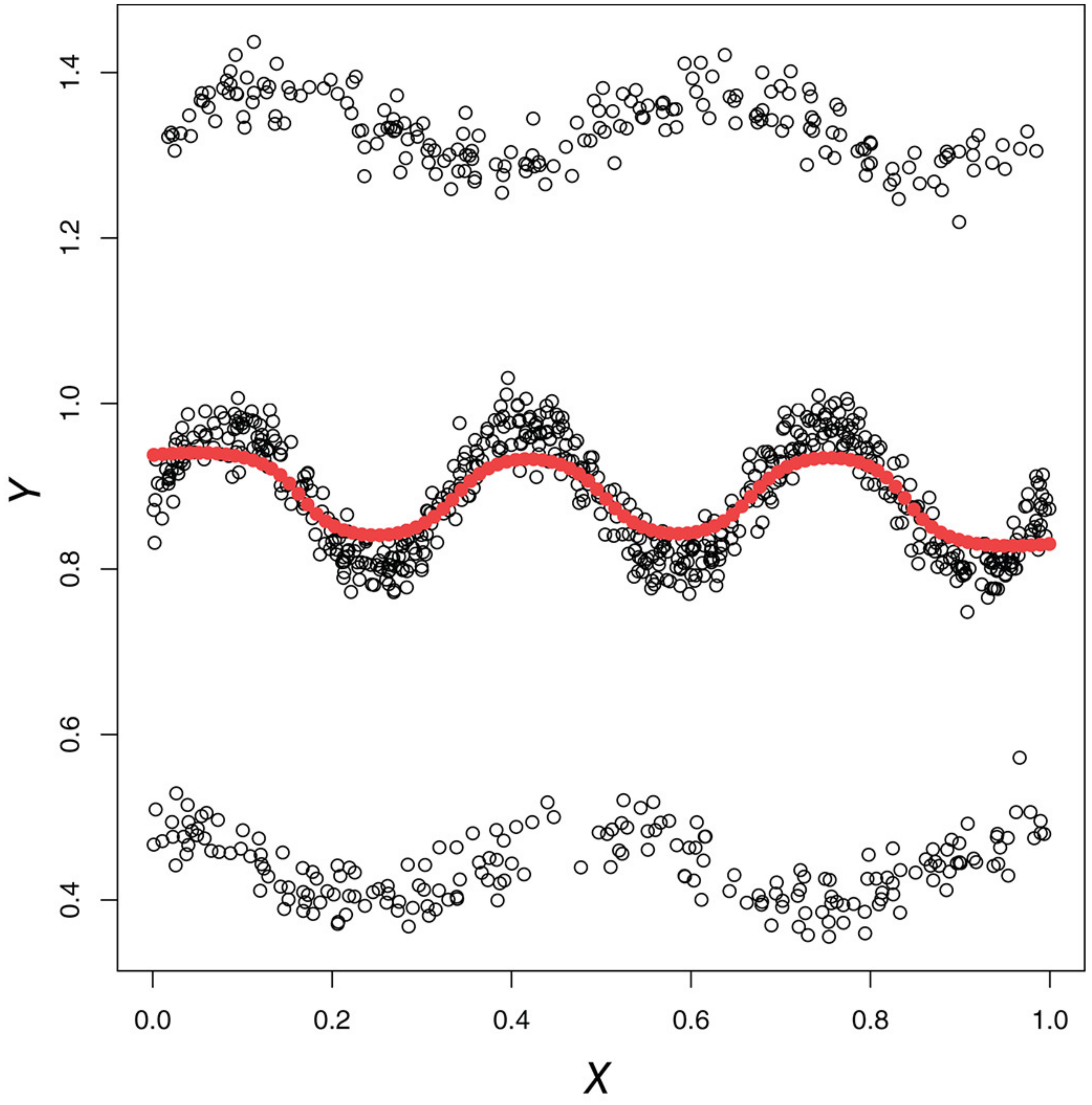
Logistic
Regression

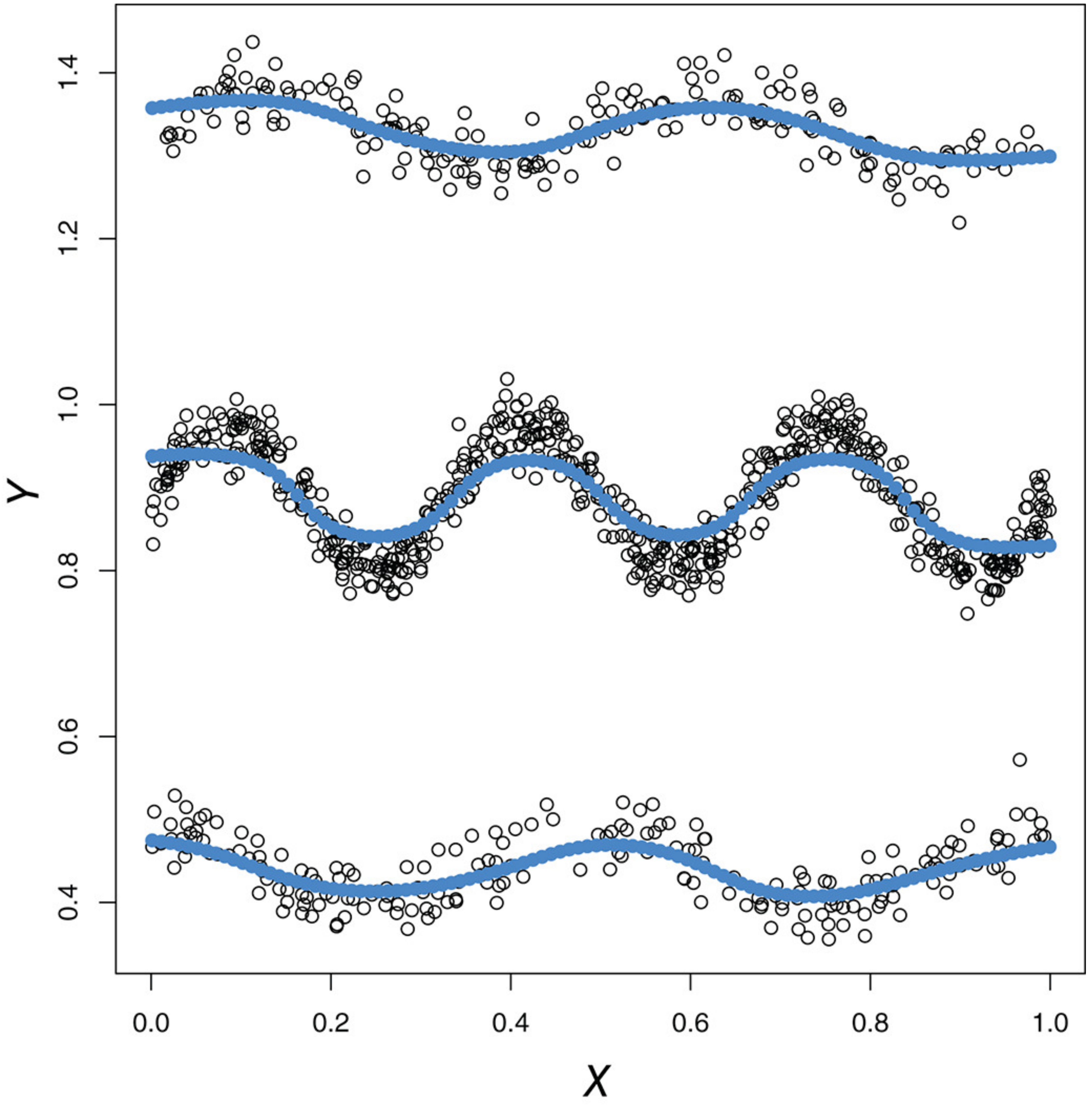
Fun Case Study 2

- A big part of machine learning is also learning more complex distributions
  - Mixture models and modal regression
  - Generative Models
- Same concepts about finding parameters from the distribution, using maximum likelihood objectives
  - but the distributions are just more complex than Gaussians and Gammas

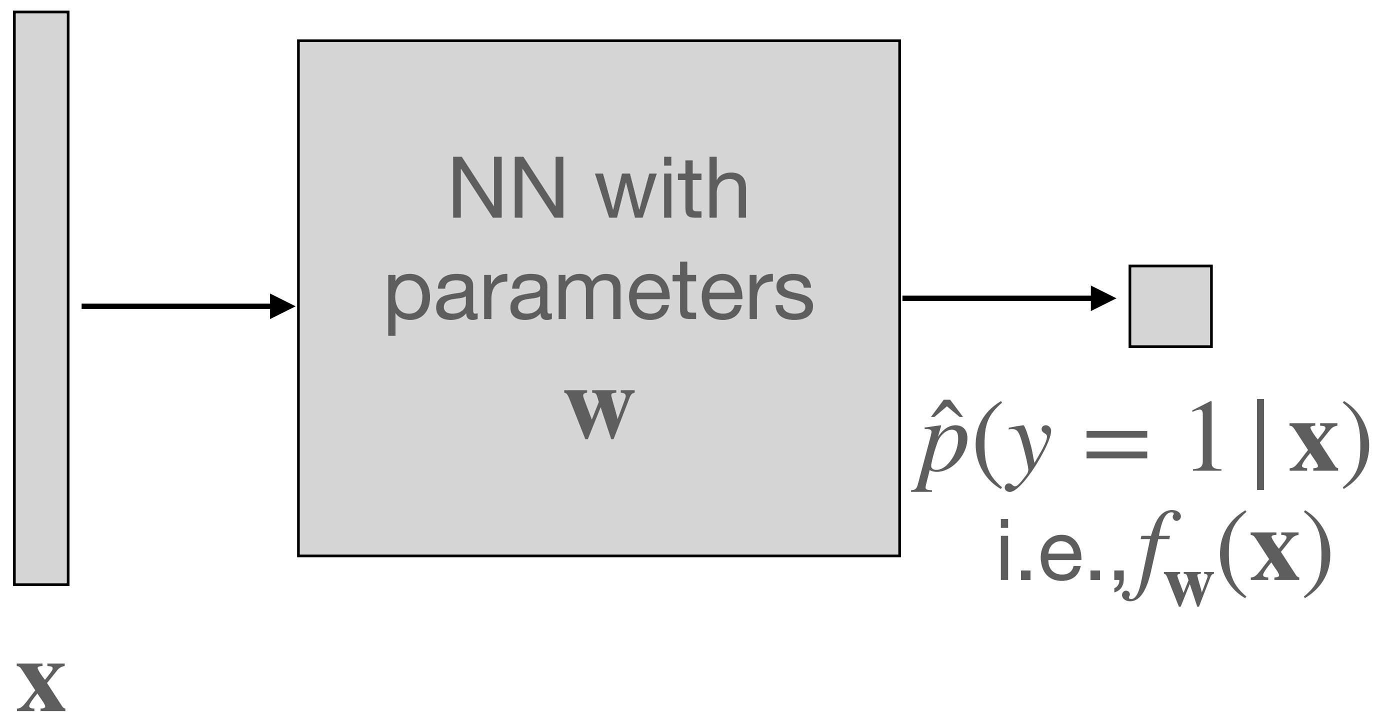
Example: Modal Regression

p(y|x) has is multimodal
it has three modes
When making predictions
it can be useful to know
that the central mode is
most likely but that these
other two very different
outcomes can occur





Case Study 3



- Let's apply our knowledge. Imagine you are learning a neural network $f_{\mathbf{w}}(\mathbf{x}) \approx p(y = 1 \mid \mathbf{x})$ to do binary classification
    - Say to predict whether a patient has a disease, based on 100 attributes about them (age, medical info, etc.)
- Just like polynomial logistic regression, you use the cross-entropy
- $c_i(\mathbf{w}) = -y_i \ln f_{\mathbf{w}}(\mathbf{x}_i) - (1 - y_i)\ln(1 - f_{\mathbf{w}}(\mathbf{x}_i))$
- Imagine you get 70% accuracy on training and 60% on test. Not great.
- What could be the problem? (Let's brainstorm together)

Case Study 3

- Imagine you get 70% accuracy on training and 60% on test. Not great.
- What could be the problem and how might we check if it is a problem? (Let's brainstorm together)
  - Optimization issues:
    - Nonconvex objective, maybe got stuck in a local min. **Test**: run multiple times from different random starts and check variability
    - Not enough epochs. **Test**: look a cross-entropy curve on training data. Has it flattened? Or is it still decreasing and you are not a stationary point?

Case Study 3

- Imagine you get 70% accuracy on training and 60% on test. Not great.
- What could be the problem and how might we check if it is a problem? (Let's brainstorm together)
  - Optimization issues (nonconvexity, did not converge)
  - Model complexity issues: Overfitting or underfitting? Doesn't seem like much overfitting, but there is a little. **Test**: compare to a simpler linear model as a baseline, and also test a more complex model (bigger NN)

Case Study 3

- Imagine you get 70% accuracy on training and 60% on test. Not great.
- What could be the problem and how might we check if it is a problem? (Let's brainstorm together)
    - Optimization issues (nonconvexity, did not converge)
    - Model complexity issues
    - Partial observability: true $p(y = 1 \mid \mathbf{x}) = 0.7$. Inherent error (even on training) is 70%, cannot be overcome.