Final Review

CMPUT 367: Intermediate Machine Learning

Goal of these Slides

- Go over each section of the notes and highlight key concepts
- Additionally highlight what I will and will not test
  - It is in the notes for your knowledge, but hard to directly test
- Not reviewing for Practice Final —> That will be the next session
- Note: like you can see in the Practice Final, the final largely focuses on Chapter 8 onwards. But as usual it builds on your knowledge from earlier chapters

Chapter 1: Intro to ML

- Know the difference between a generative model and predictor (1.1)
- **Will not be directly tested:**
  - Relationship to Statistics and Probability (1.2)
  - The Blessing and Curse of Dimensionality (1.3)
  - SVDs and Eigenvalue decompositions (1.4)
  - You will not need to take gradients

Chapter 2: Intermediate Probability Concepts

- Understand the definition of a multi-dimensional probability (2.1)
- Understand the definition of a mixture of distributions (2.2)
- Know the purpose of the KL divergence (2.3)
- **Will not be directly tested:**
  - Knowing the PMFs or PDFs of specific distributions
  - Specific expectation and variance formulas
  - Remembering the KL divergence formula

Chapter 3: Revisiting Linear Regression

- Understand that Linear Regression and l2-regularized linear regression have closed-form solutions (unlike most GLMs)
- Understand that this let's us characterize the bias and variance of these solutions
- Understand the LR solution is unbiased, if the true function is linear
- Understand LR+l2 is biased, but that asymptotically (as n grows) they reach the same solution
- **Will not be directly tested:**
  - Any specific closed-form solutions; I will give them to you if you need them

Chapter 4: Intermediate Optimization Principles

- Understand multivariate gradient descent, including gradients (4.3) and the role of the Hessian in second-order GD (4.1)
- Understand Stochastic GD (SGD) and the reason to move from full batch GD to mini-batch SGD (4.4)
- Understand the role of vector stepsize algorithms like RMSProp and the use of momentum (4.5)
- **Will not be directly tested:**
  - Properties of the Hessian and directional derivatives (3.2)
  - Knowing the updates of specific vector stepsize algorithms

More Advanced Exercise Question

- How might the size of the dataset n interact with the number of epochs that we need to converge?

More Advanced Exercise Question

- How might the size of the dataset n interact with the number of epochs that we need to converge?
- **Answer**: With a very large dataset, we are doing more updates in each epoch and likely need fewer epochs to converge.

Chapter 5: GLMs

- Understand that Generalized Linear Models (GLMs) allow us to model
  - $p(y \mid \mathbf{x})$ = any natural exponential family distribution with natural parameter $\theta = \mathbf{x}^\top \mathbf{w}$
  - with associated transfer function g such that $g(\mathbf{x}^\top \mathbf{w})$ approximates $\mathbb{E}[Y \mid \mathbf{x}]$
- Understand that multinomial logistic regression is for multi-class classification
- **Will not be directly tested:**
  - Knowing specific GLM updates; if I need you to reason about one I will give it to you
  - The details of exponential family distributions (5.2)

Exercise Question

- Imagine you have multinomial logistic regression implemented. How would you use this code to do binary classification?

Chapter 6: Constrained Optimization

- Understand that we need to use a different approach when we have a constrained optimization (3.5)
- Understand that proximal gradient descent is a reasonably general purpose approach for constrained or non-smooth optimization (3.5)
- **Will not be directly tested:**
  - You do not need to know specific proximal operators
  - You do not need to know about KKT conditions nor how to get the proximal operator for the simplex constraint (3.6)

Exercise for constrained optimization

- Let us revisit the optimization for mixture models
- $$\min_{w_1,\ldots,w_k \geq 0, \sum_{k=1}^{m} w_k = 1} -\sum_{k=1}^{m} d_k \ln w_k \qquad \text{(where } d_k = \sum_{i=1}^{n} p_t[i,k] > 0\text{)}$$
- To solve this, we can be lazy and first just check: does a stationary point give us a feasible solution?

Exercise for constrained optimization (cont.)

$$\frac{\partial}{\partial w_j} \sum_{k=1}^{m} d_k \ln w_k = \sum_{k=1}^{m} d_k \frac{\partial}{\partial w_j} \ln w_k$$

$$= d_j \frac{1}{w_j} = 0$$

Stationary points are plus/minus infty, clearly not a feasible solution (does not satisfy our constraints)
Our lazy step failed.

Exercise for constrained optimization (cont.)

$$\frac{\partial}{\partial w_j} \sum_{k=1}^{m} d_k \ln w_k = \sum_{k=1}^{m} d_k \frac{\partial}{\partial w_j} \ln w_k$$

$$= d_j \frac{1}{w_j} = 0$$

Stationary points are plus/minus infty, clearly not a feasible solution (does not satisfy our constraints)

Our lazy step failed. If the stationary point *had* been a feasible solution (satisfied $w_1, \ldots, w_k \geq 0, \sum_{k=1}^{m} w_k = 1$), then we would be done and wouldn't need to use any fancier optimization approaches

Exercise for constrained optimization (cont)

- Let us now incorporate one of the constraints. We can see our objective actually will not prefer negative weights, so let's first do the sum constraint
- We consider now an equivalent augmented objective (Lagrangian), and see if a stationary point of this objective gives us a solution
- $$L(\mathbf{w}, a) = -\sum_{k=1}^{m} d_k \ln w_k + a\left(\sum_{k=1}^{m} w_k - 1\right)$$
- Solve for $\max_{a\in\mathbb{R}} \min_{\mathbf{w}\in\mathbb{R}^m} L(\mathbf{w}, a)$ we know a solution to this *must* satisfy this constraint, as otherwise w suffers infinite loss

Exercise for constrained optimization (cont)

- $L(\mathbf{w}, a) = -\sum_{k=1}^{m} d_k \ln w_k + a\left(\sum_{k=1}^{m} w_k - 1\right)$ Lets start by solving for w

  $$\frac{\partial}{\partial w_j} L(\mathbf{w}, a) = -\sum_{k=1}^{m} d_k \frac{\partial}{\partial w_j} \ln w_k + a w_j$$

- $$= -d_j \frac{1}{w_j} + a = 0 \implies w_j = \frac{d_j}{a}$$

- We know a solution must have a where $\sum_{k=1}^{m} w_k = \sum_{k=1}^{m} \frac{d_k}{a} = 1 \implies a = \sum_{k=1}^{m} d_k$

Exercise for constrained optimization (cont)

- $$L(\mathbf{w}, a) = -\sum_{k=1}^{m} d_k \ln w_k + a\left(\sum_{k=1}^{m} w_k - 1\right) \qquad w_j = \frac{d_j}{a}$$
- We know a solution must have a where
$$\sum_{k=1}^{m} w_k = \sum_{k=1}^{m} \frac{d_k}{a} = 1 \implies a = \sum_{k=1}^{m} d_k \implies w_j = \frac{d_j}{\sum_{k=1}^{m} d_k}$$
- Feasible solution, since $d_j > 0$ and so $w_j > 0$
- (We didn't need to go explicit enforce this condition)

Chapter 7: Evaluating Generalization Performance

- Understand that cross validation allows us to evaluate a model trained on the entire dataset (without having to have a hold-out test set)
- Understand the k-fold CV algorithm
- Understand the repeated random subsampling (RSS) CV algorithm
- **Will not be directly tested:**
  - The nuances about the bias-variance distinctions for different CV choices

Chapter 7: Evaluating Generalization Performance (cont)

- Know what it means to select hyperparameters
- Understand the utility of CV for hyperparameter selection
- Understand the difference between internal CV and external CV
  - internal CV is for hyperparameter selection and external is to evaluate the algorithm that might use internal CV
- **Will not be directly tested:**
  - Knowing how to pick the set of hyperparameters to be tested with CV

Refresher on internal & external CV

Algorithm 5: Nested cross-validation on a dataset $\mathcal{D}$

Partition the dataset $\mathcal{D}$ into k_{external} folds
Initialize err-f $= 0$
for $i = 1$ to k_{external} **do**
  Set $\mathcal{D}_{\text{te}}^{(i)}$ to the data in fold i
  Set $\mathcal{D}_{\text{tr}}^{(i)} = \mathcal{D} - \mathcal{D}_{\text{te}}^{(i)}$
  // Call the Learner on $\mathcal{D}_{\text{tr}}^{(i)}$; as part of its algorithm, it uses CV to picks hypers
  Partition the dataset $\mathcal{D}_{\text{tr}}^{(i)}$ into k_{internal} folds
  for h in the set of hyperparameters H **do**
    Initialize err$[h] = 0$
    for $j = 1$ to k_{internal} **do**
      Set $\mathcal{D}'^{(j)}_{\text{te}}$ to the data in fold j for dataset $\mathcal{D}_{\text{tr}}^{(i)}$
      Set $\mathcal{D}'^{(j)}_{\text{tr}} = \mathcal{D}_{\text{tr}}^{(i)} - \mathcal{D}'^{(j)}_{\text{te}}$
      Train $f = \text{Alg}(\mathcal{D}'^{(j)}_{\text{tr}}, h)$
      err$[h]$ = err$[h]$+ error for f on $\mathcal{D}'^{(j)}_{\text{te}}$
    err$[h]$ = err$[h]/k_{\text{internal}}$
  Pick $h^* = \text{argmin}_{h \in H}\, \text{err}[h]$
  // Learner done picking its hyperparameter, can now return the learned function
  Train $f = \text{Alg}(\mathcal{D}_{\text{tr}}^{(i)}, h^*)$
  err-f = err-f+ error of f on $\mathcal{D}_{\text{te}}^{(i)}$
err-f = err-f$/k_{\text{external}}$
return f and err-f

Chapter 8: Fixed Representations

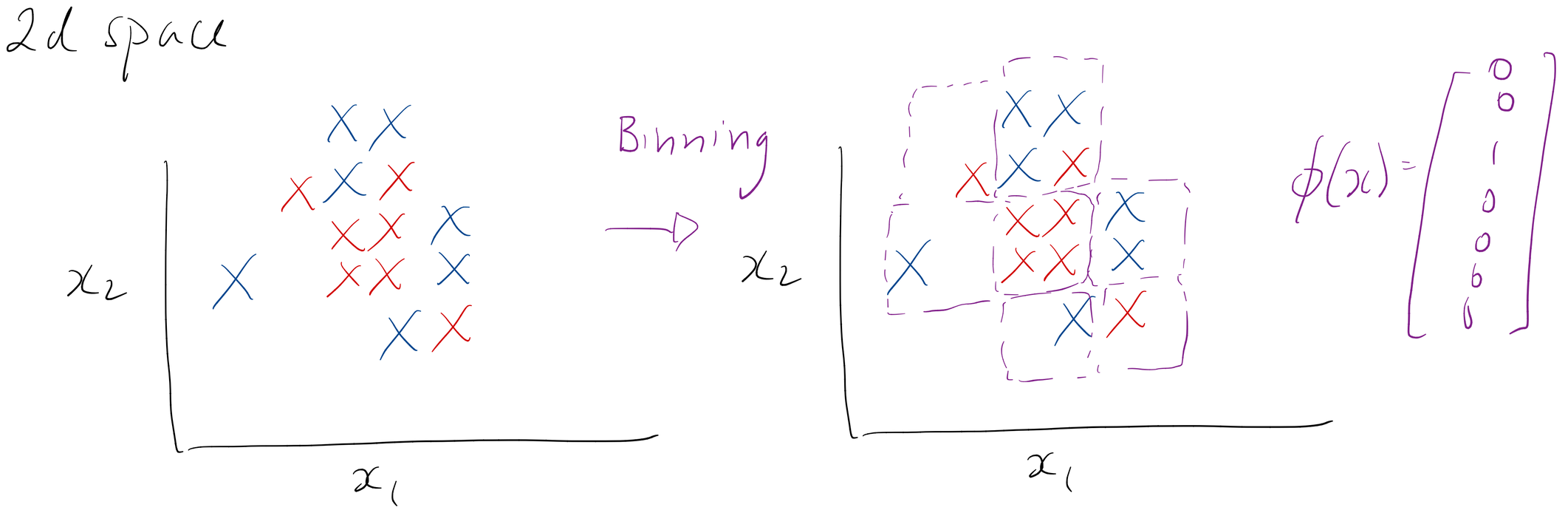
- Understand that projecting to higher dimensions makes data separable (classification) or allows for a simpler function for regression
- Understand that RBF network define features using RBF kernels to a set of centers, with similarity controlled by the width of the RBF
- Understand that Prototype Representations use similarities to prototypes taken from the training dataset
- Understand that l1 does feature selection, and that is more useful when we blow up our feature space using fixed representations
- **Will not be directly tested:**
  - Knowing specific kernels
  - The advanced remark about the representer theorem

Exercise

- What are the implications of using l1 regularization with polynomial features?

Exercise

- I didn't give you an example of how projecting to higher dimensions also facilitates regression with simple (linear functions)
- Can you think of a similar example to this one, but for regression?



Chapter 9: Learned Representations

- Understand that PCA extracts a lower-dimensional representation
- Understand the objective underlying PCA (minimize $\|\mathbf{x} - \mathbf{hD}\|_2^2$ for every x)
- Understand that sparse coding similarly minimizes $\|\mathbf{x} - \mathbf{hD}\|_2^2$, but additionally has an l1 regularizer on h to find a high-dimensional sparse representation
- **Will not be directly tested:**
  - PPCA
  - Algorithms for PCA and sparse coding, such as matrix factorization
  - Interpretations of latent factors (that was only for intuition about what might be learned in a representation)

Advanced Exercise Question

- Imagine we first expand the dimension using a kernel representation, going from 10 features to 5000.
  - Subquestion: why are there 5000 features?
- Then we apply PCA to extract 100 features. How do we interpret what those features are?
- Is it equivalent to PCA or l1-regularization to get to 100 features?

Chapter 9: Learned Representations

- Understand types of transformation on the input given by a neural network
- Understand that backpropagation is gradient descent
- Understand that linear autoencoders also extract a low-dimensional representation like PCA
  - Can see nonlinear autoencoders as a nonlinear extension of PCA
- **Will not be directly tested:**
  - You will not need to derive the gradients for an NN
  - You will not be tested on supervised autoencoders

Exercise Question

- We discussed that many transformations consist of (1) linear weighting followed by (2) nonlinear activation (differentiable almost everywhere)
- What are some other activations we could consider using in a network, beyond the three we discussed (ReLU, sigmoid, tanh)?

Exercise Question

- Now imagine that we want to get a new representation with 5000 features using NNs.
- How would we do that? (there is more than one answer to this question)

Exercise Question

- Write down the set of functions F1 obtained using a kernel representation with kernel k, and a random subset of 100 points from the training data as centers (assume $\mathcal{X}$ is the space of all possible inputs $\mathbf{x}$)
- Write down the set of functions F2 obtained using an NN with two hidden layers each of size 256, with ReLu activations, for regression

Exercise Question

- Write down the set of functions F1 obtained using a kernel representation with kernel k, and a random subset of 100 points from the training data as centers (assume $\mathcal{X}$ is the space of all possible inputs $\mathbf{x}$)
- Write down the set of functions F2 obtained using an NN with two hidden layers each of size 256, with ReLu activations, for regression
- If I told you that F1 is a subset of F2, what does that mean? Which class has higher complexity (or capacity)?
- How do you know one is a subset of the other? Is F1 a subset of F2 here?

Chapter 10: Mixture Models

- Understand that the EM algorithm consists of (a) the introduction of auxiliary variables z and (b) alternating between updating $p(z_i \mid x_i)$ and parameters $\boldsymbol{\theta}$
- Understand that the M-step updates $\boldsymbol{\theta}$ for fixed $p(z_i \mid x_i)$ and the E-step updates $p(z_i \mid x_i)$ for fixed $\boldsymbol{\theta}$
- **Will not be directly tested:**
  - The MLE solution for Multivariate Gaussians (4.1)
  - You do not need to memorize the EM algorithm, but you should be able to recognize key components of it

Exercise

- How would you use k-fold CV to pick the number of centers for a GMM?

Exercise

- How would you use k-fold CV to pick the number of centers for a GMM?
- Answer: You would decide on the set of numbers to select from,
  e.g., H = {2, 4, 8, 16}
- After partitioning the data into k folds, for each hyper m in H and each fold f
  - Learn the GMM phat on all but fold f
  - Evaluate on fold f, by computing the negative log likelihood on the data
    sum_(x in fold f) -ln phat(x)

Chapter 11: Generative Models & Data Representations

- Understand that both PPCA and VAEs make the assumption that
- $p(\mathbf{x}) = \int p(\mathbf{x}\,|\,\mathbf{h})p(\mathbf{h})d\mathbf{h}$ with $p(\mathbf{h}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Understand that PPCA assumes a linear relationship between $\mathbf{x}$ and $\mathbf{h}$
  $p(\mathbf{x}\,|\,\mathbf{h}) = \mathcal{N}(\mathbf{hD}, \sigma^2\mathbf{I})$
- And that VAE generalizes to a nonlinear relationship, using NN $f_{\mathbf{W}}$ to give
  $p(\mathbf{x}\,|\,\mathbf{h}) = \mathcal{N}(f_{\mathbf{W}}(\mathbf{h}), \sigma^2\mathbf{I})$

Chapter 11: Generative Models (cont)

- Understand that we learn the encoder $q(\mathbf{h} \mid \mathbf{x})$ only as part of the optimization, to help us learn $p(\mathbf{x} \mid \mathbf{h})$; we do not need $q(\mathbf{h} \mid \mathbf{x})$ itself
- Understand how to sample from a VAE
  - Step 1: Sample $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then
  - Step 2: query the decoder part of the VAE network $f_{\mathbf{W}}(\mathbf{h})$
- **Will not be directly tested**
  - Knowing the VAE objective (the elbo loss)
  - The connection to Expectation-Maximization (9.3)
  - The reparameterization trick and the gradient update for the VAE

Exercise

- The last slide said: To sample from a VAE
  - Step 1: Sample $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then
  - Step 2: query the decoder part of the VAE network $f_{\mathbf{W}}(\mathbf{h})$
- But why don't we sample $\mathbf{h}$ from $q(\mathbf{h} \mid \mathbf{x})$?

Chapter 12: Bias, Variance and Generalization Error

- Understand that the generalization error of a function f is the error in expectation across all possible datapoints (expected cost)

$$\mathrm{GE}(f) = \mathbb{E}[(f(X) - Y)^2] = \underbrace{\mathbb{E}[(f(X) - f^*(X))^2]}_{\text{reducible error}} + \underbrace{\mathbb{E}[(f^*(X) - Y)^2]}_{\text{irreducible error}} \qquad (12.1)$$

- GE is about a specific function f, rather than a function class that outputs $f_{\mathscr{D}}$ that varies with data

Chapter 12: Bias, Variance and Generalization Error

- Understand that we can reason about function $f_{\mathscr{D}}$ as a random variable, where randomness comes from the underlying dataset
- Understand that we can reason about the generalization error of functions from a function class, by considering the bias and variance of this $f_{\mathscr{D}}$
- Understand that reducible error of $f_{\mathscr{D}}$ decomposes into bias and variance
- For a specific x, we have $\mathbb{E}\left[\left(f_{\mathcal{D}}(\mathbf{x}) - f^*(\mathbf{x})\right)^2\right] = \left(\mathbb{E}\left[f_{\mathcal{D}}(\mathbf{x})\right] - f^*(\mathbf{x})\right)^2 + \operatorname{Var}\left[f_{\mathcal{D}}(\mathbf{x})\right].$

$$\mathbb{E}[(f_{\mathcal{D}}(\boldsymbol{X}) - f^*(\boldsymbol{X}))^2] = \mathbb{E}_{\boldsymbol{X}}\left[\left(\mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})] - f^*(\boldsymbol{X})\right)^2 + \operatorname{Var}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})]\right] \tag{12.2}$$

Exercise Question

- We wrote F1 the set of function using a kernel representation and F2 using an NN. We thought about the case where F1 is a subset of F2
- Do you think F1 or F2 has higher bias?
- Do you think F1 or F2 has higher variance?
- Why is this reasoning useful? Can’t we just measure generalization error of our actual learned function using a test set or cross validation?

Chapter 12: Bias, Variance and Generalization Error

- Understand the definition of covariate shift
  - $p_{\text{train}}(\mathbf{x}, y) = p(y \mid \mathbf{x}) p_{\text{train}}(\mathbf{x}) \neq p(y \mid \mathbf{x}) p_{\text{test}}(\mathbf{x}) = p_{\text{test}}(\mathbf{x}, y)$

A more realistic example of covariate shift

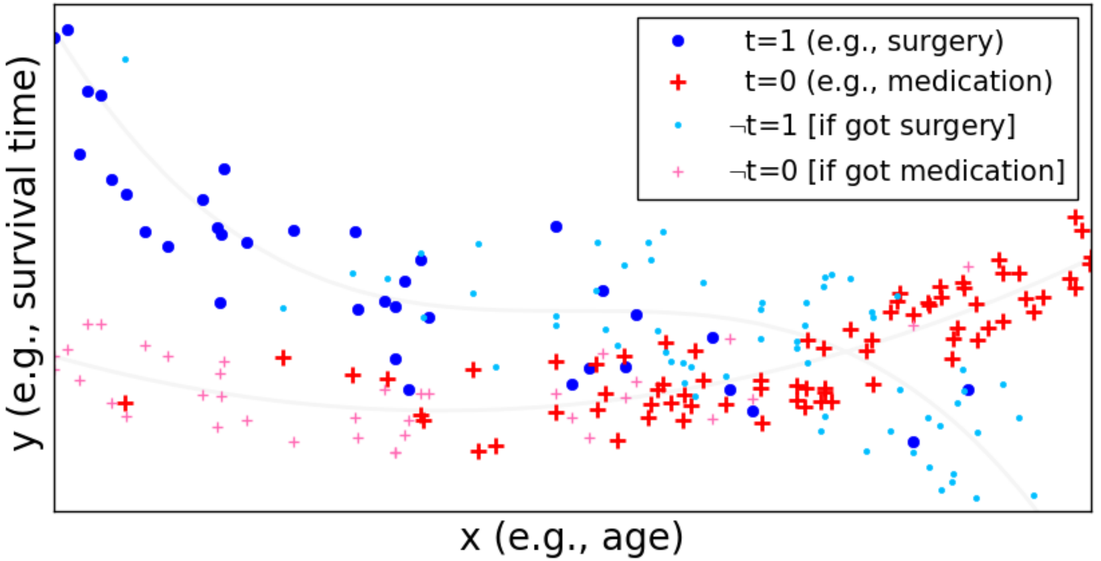


Figure 1.2: An example observational dataset (synthetic). Points in ● represent a patient who actually got surgery ($t = 1$) and indicate their respective *factual* outcome. Points in • represent patients who in reality got medication but indicate their *counterfactual* outcome had they got surgery ($\neg t = 1$).

Exercise: What is ptrain and ptest?

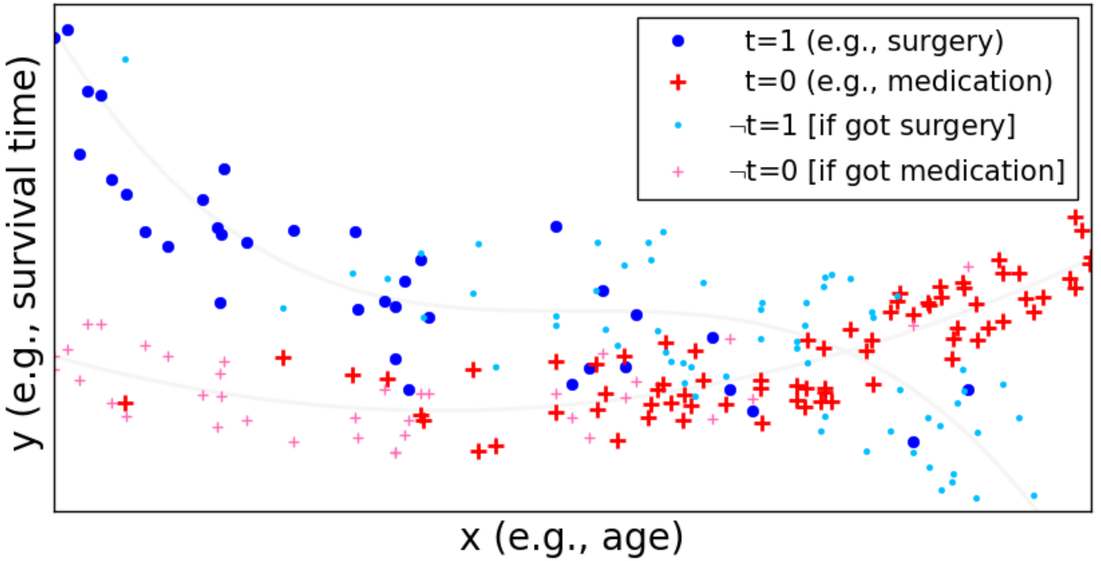


Figure 1.2: An example observational dataset (synthetic). Points in ● represent a patient who actually got surgery ($t = 1$) and indicate their respective *factual* outcome. Points in • represent patients who in reality got medication but indicate their *counterfactual* outcome had they got surgery ($\neg t = 1$).

Chapter 12: Bias, Variance and Generalization Error

- Understand the definition of covariate shift
  - $p_{\text{train}}(\mathbf{x}, y) = p(y \mid \mathbf{x}) p_{\text{train}}(\mathbf{x}) \neq p(y \mid \mathbf{x}) p_{\text{test}}(\mathbf{x}) = p_{\text{test}}(\mathbf{x}, y)$
- Understand that our definition for GE stays the same
  - still about deployment data, but before $p_{\text{train}}(\mathbf{x}, y) = p_{\text{test}}(\mathbf{x}, y)$ so we simply called them both p

$$\text{GE}(f) = \mathbb{E}_{p_{\text{test}}}[(f(X) - Y)^2] = \int_{\mathcal{X}} p_{\text{test}}(\mathbf{x}) \mathbb{E}[(f(\mathbf{x}) - Y)^2 | X = \mathbf{x}] d\mathbf{x} \qquad (12.3)$$

Exercise

- When we talk about bias-variance, in expectation across inputs, how does this change under covariate shift?

$$\mathbb{E}[(f_{\mathcal{D}}(\boldsymbol{X}) - f^*(\boldsymbol{X}))^2] = \mathbb{E}_{\boldsymbol{X}}\left[(\mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})] - f^*(\boldsymbol{X}))^2 + \mathrm{Var}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})]\right] \tag{12.2}$$

Exercise

- When we talk about bias-variance, in expectation across inputs, how does this change under covariate shift?

$$\mathbb{E}[(f_{\mathcal{D}}(\boldsymbol{X}) - f^*(\boldsymbol{X}))^2] = \mathbb{E}_{\boldsymbol{X}}\left[(\mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})] - f^*(\boldsymbol{X}))^2 + \mathrm{Var}_{\mathcal{D}}[f_{\mathcal{D}}(\boldsymbol{X})]\right] \tag{12.2}$$

- Expectation over datasets assumes $\mathscr{D} \sim p_{\text{train}}$
- Expectation over X assumes $\mathbf{x} \sim p_{\text{test}}$
- (Before both were sampled from the same distribution p)
- Why is this the new definition?

Chapter 12: Bias, Variance and Generalization Error (cont)

- Most of the rest of Chapter 12 will not be directly tested
- **Will not be directly tested**
  - 12.2 on High probability bounds
  - I only expect you to know what covariate shift is; I will not test on understanding how to fix covariate shift
  - We will not talk about nonstationary in p(y|x) (12.3.3)
  - Comments about the impacts of covariate shift on bias-variance (12.4)

Chapter 13: Convergence Rates

- You learned that norm of the gradient reduces at a rate of 1/t for gradient descent
- You learned that (expected) norm of the gradient reduces at a rate of 1/t for stochastic gradient descent too! Even though it uses a much noisier gradient
  - But you converge to a region around the stationary point, proportional to the magnitude of this noise
- You also gained some insight into how to pick the mini-batch size
- **I will not test you on any of this chapter**

Note about stopping conditions

- We reasoned that these algorithms converge or stop within a finite number of iterations (of order O(1/epsilon) for gradient magnitude epsilon).
- But for SGD we do not necessarily measure the gradient norm and decide to stop
  - At least this would be too expensive to do every iteration
- For GD, it is common to check the norm of the gradient as a stopping criterion
- Our analysis just showed us that we get to within such a region in O(1/epsilon) steps, for SGD

Chapter 14: Missing Data

- Understand how to do imputation using PCA (matrix factorization)
- Understand what is means to do multiple imputation and why we want to do it
- Understand the Missing at Random assumption
- **Will not be directly tested**
  - Connections to the transductive and semi-supervised settings
  - You do not need to understand how to compute $p(\mathbf{x}_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}})$
  - 14.2.2 about difficulties with NNs

Exercise: PCA (matrix completion)

- In PCA we solve for $\min_{\mathbf{h}_1,\mathbf{h}_2,\ldots,\mathbf{h}_n \in \mathbb{R}^p, \mathbf{D} \in \mathbb{R}^{p \times d}} \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - \mathbf{h}_i \mathbf{D}_{:j})^2$
- In PCA with missing data, $\min_{\mathbf{h}_1,\mathbf{h}_2,\ldots,\mathbf{h}_n \in \mathbb{R}^p, \mathbf{D} \in \mathbb{R}^{p \times d}} \sum_{i=1}^{n} \sum_{j \in \mathscr{A}_i} (x_{ij} - \mathbf{h}_i \mathbf{D}_{:j})^2$
- Why didn't we just set $\mathbf{x}_{\mathscr{M}_i} = \mathbf{0}$ (set unavailable values to zero) and call PCA? We will set get back the h's and D. How is this different?

Missing at Random

- Define RV $I_{\mathscr{M}}$ that is 0 or 1. It is 1 if indices $\mathscr{M}$ are missing and 0 if they are not missing
  - Why is this a random variable?
- MAR = Conditional independence between $I_{\mathscr{M}}$ and $\mathbf{x}_{\mathscr{M}}$, given $\mathbf{x}_{\mathscr{A}}$
- $p(\mathbf{x}_{\mathscr{M}}, I_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}}) = p(\mathbf{x}_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}}) p(I_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}})$
- Conditional independence implies $p(\mathbf{x}_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}}, I_{\mathscr{M}}) = p(\mathbf{x}_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}})$

Exercise

- Imagine you do PCA on the data to get $\mathbf{D}$
- And you do PPCA to get $\tilde{\mathbf{D}}$ and σ^2 where $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \tilde{\mathbf{D}}\tilde{\mathbf{D}}^\top + \sigma^2\mathbf{I})$
- We talked about how we can use the PPCA solution, to get $p(\mathbf{x}_\mathscr{M} \mid \mathbf{x}_\mathscr{A})$ from $p(\mathbf{x})$, and so sample from $p(\mathbf{x}_\mathscr{M} \mid \mathbf{x}_\mathscr{A})$
- Why can't we do multiple imputation with the PCA solution?

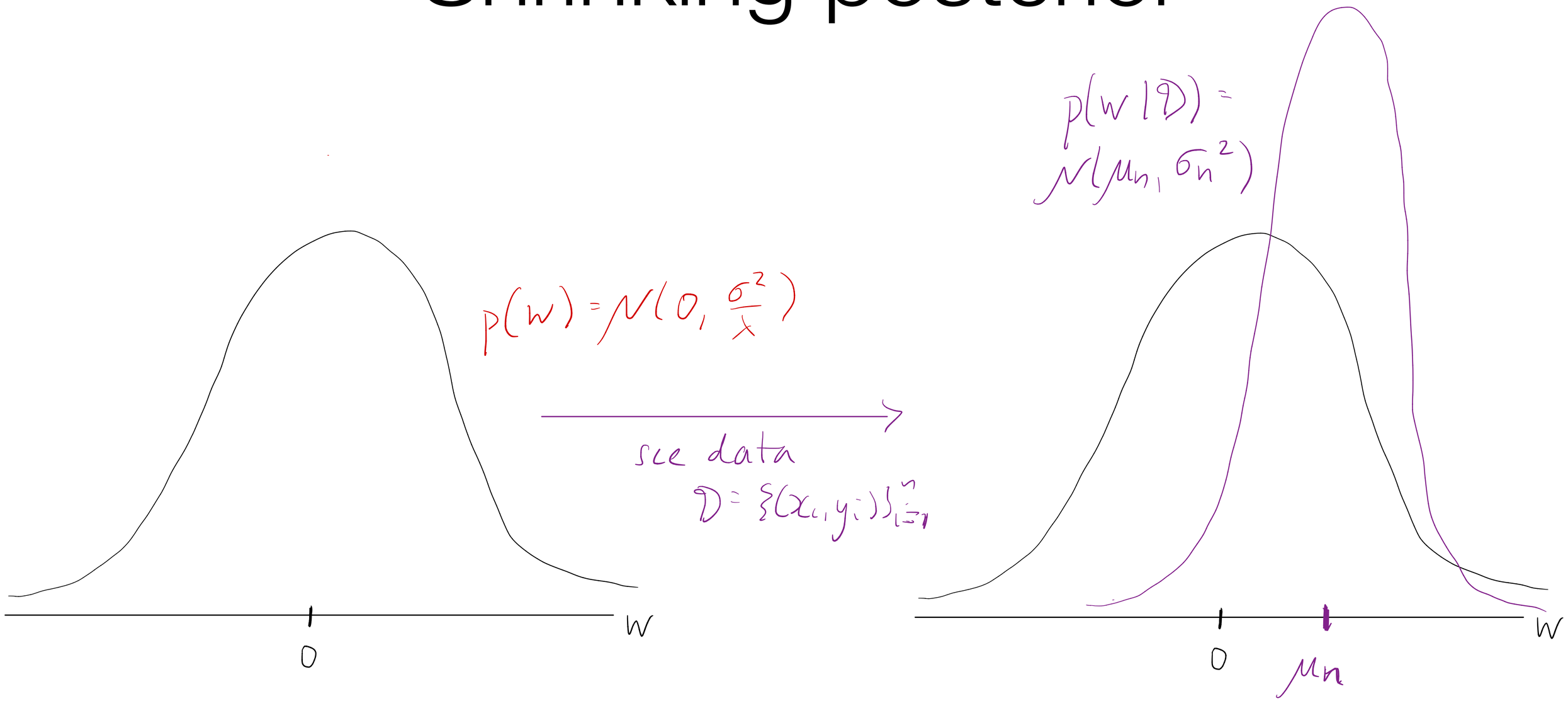
Exercise Question

- Obtaining $p(y \mid \mathbf{x}_{\mathscr{A}}) = \int p(\mathbf{x}_{\mathscr{M}} \mid \mathbf{x}_{\mathscr{A}}) p(y \mid \mathbf{x}_{\mathscr{A}}, \mathbf{x}_{\mathscr{M}}) d\mathbf{x}_{\mathscr{M}}$ is hard in general
- But, under some conditions, it is actually easy. Consider a case where we have two binary features $\mathbf{x} = [x_1, x_2]$, namely $x_1, x_2 \in \{0,1\}$ and
  $p(y \mid x_1 = 0, x_2 = 0) = \mathcal{N}(\mu_{00}, \sigma^2)$, $p(y \mid x_1 = 0, x_2 = 1) = \mathcal{N}(\mu_{01}, \sigma^2)$
  $p(y \mid x_1 = 1, x_2 = 0) = \mathcal{N}(\mu_{10}, \sigma^2)$, $p(y \mid x_1 = 1, x_2 = 1) = \mathcal{N}(\mu_{11}, \sigma^2)$
- To get $p(y \mid x_1)$ we just need to learn $p(x_2 \mid x_1)$ since
  $p(y \mid x_1) = \sum_{x_2 \in \{0,1\}} p(x_2 \mid x_1) p(y \mid x_1, x2)$. How do we get $p(x_2 \mid x_1)$?

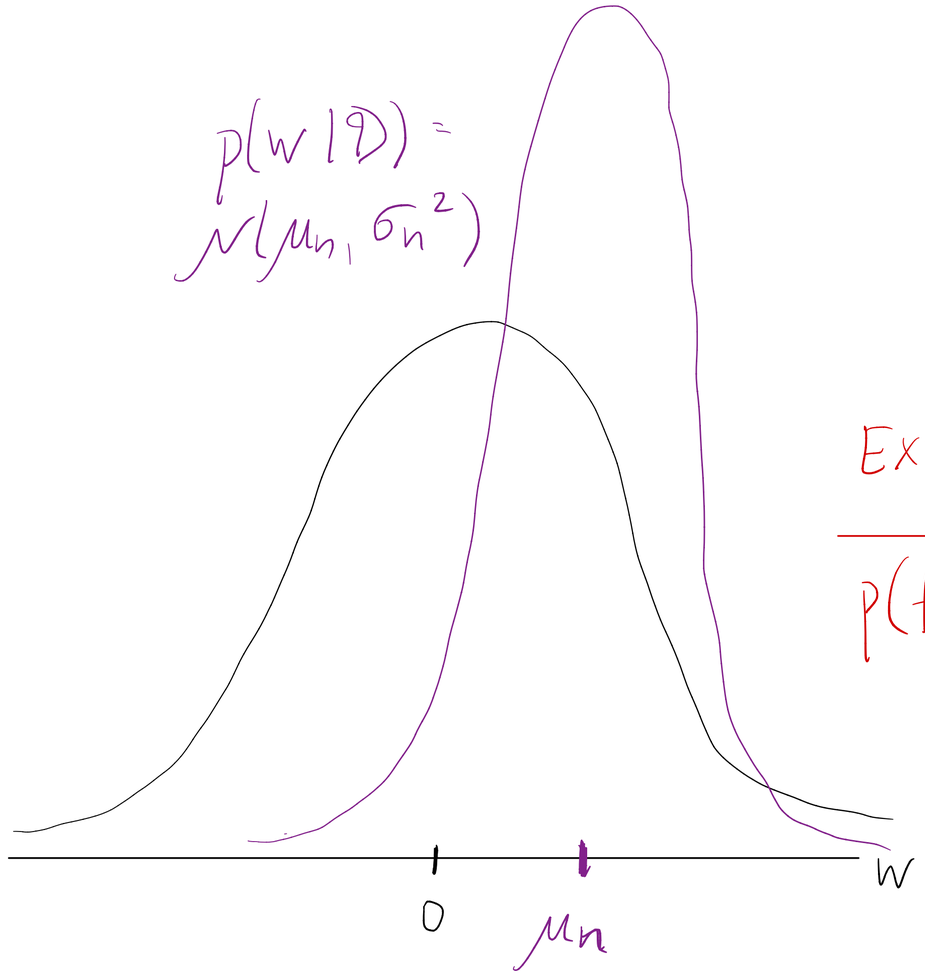
Chapter 15: Bayesian (linear) regression

- Understand that we might want to know distribution over plausible values of $\mathbf{w}$, given the evidence (data)
- This allows us to also obtain a distribution over our predictions, and so construct credible intervals $[f_{\mathbf{w}}(\mathbf{x}) - \epsilon, f_{\mathbf{w}}(\mathbf{x}) + \epsilon]$
- Understand why the posterior and credible interval shrink with growing n

Shrinking posterior



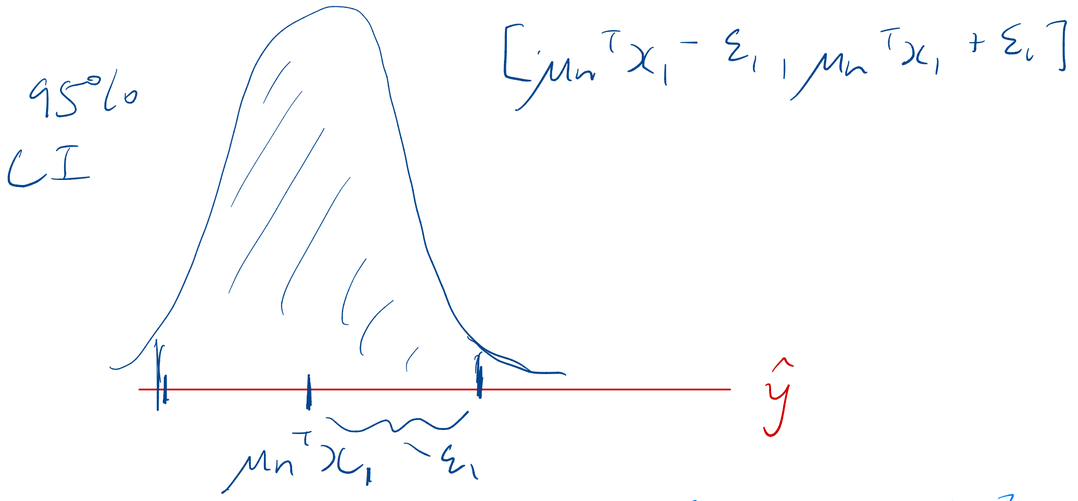
Credible Interval for Predictions



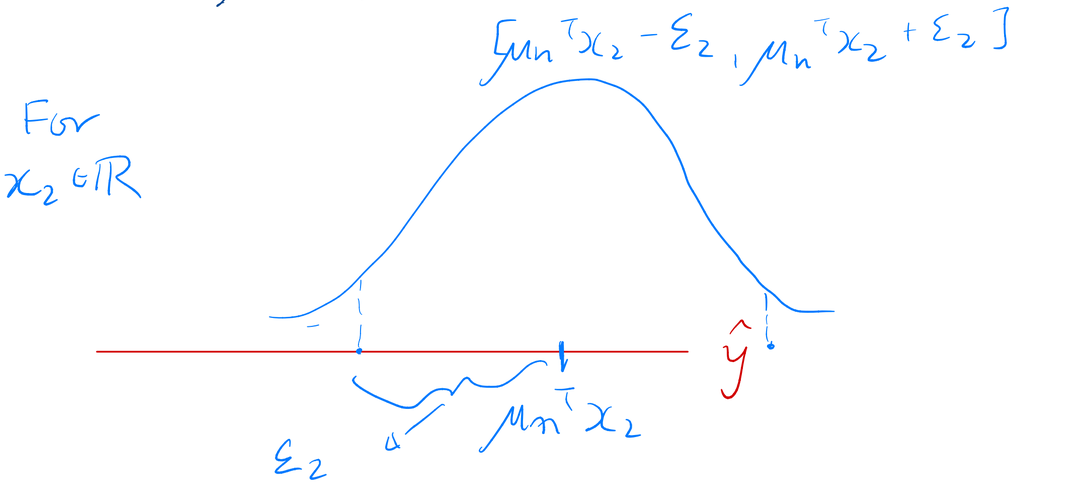
Extract

$p(f_w(x)|D)$

For $x_1 \in \mathbb{R}$



For $x_2 \in \mathbb{R}$



Added blurb to notes

To reason about this a bit more formally, let us define

$$\mathbf{C}_n \doteq \frac{1}{n}(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \quad \text{where } \mathbf{X}^\top \mathbf{X} = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top.$$

Notice that $\mathbf{C}_n \to \mathbb{E}[\mathbf{X}\mathbf{X}^\top]$ as $n \to \infty$ (as we get more and more data). Further, because we have $\lambda > 0$, we know that $\mathbf{C}_n$ is invertible for each n. Therefore, assuming that $\mathbb{E}[\mathbf{X}\mathbf{X}^\top]$ is invertible, we know that $\mathbf{x}^\top \mathbf{C}_n^{-1} \mathbf{x} \to c_x$ as $n \to \infty$ for $c_x = \mathbf{x}^\top \mathbb{E}[\mathbf{X}\mathbf{X}^\top]^{-1}\mathbf{x}$. We can write $\mathbf{\Sigma}_n = n^{-1}\mathbf{C}_n^{-1}$, giving

$$\mathbf{x}^\top \mathbf{\Sigma}_n \mathbf{x} = \mathbf{x}^\top \left(n^{-1}\mathbf{C}_n^{-1}\right) \mathbf{x} = n^{-1} \left(\mathbf{x}^\top \mathbf{C}_n^{-1} \mathbf{x}\right) \to 0.$$