Final Review

CMPUT 467: Machine Learning II

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
- Practice Final will be covered on the last day
- Note: 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: 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: 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 and the use of momentum (4.5)
- **Will not be directly tested:**
  - Directional derivatives (3.2)
  - Knowing the updates of specific vector stepsize algorithms
  - Convergence rate formulas

Exercise Question

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

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 $\theta = \mathbf{xw}$
  - with associated transfer function g such that $g(\mathbf{xw})$ 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?

Exercise Question

- Imagine you have multinomial logistic regression implemented. How would you use this code to do binary classification?
- Transform dataset of (x,y) with y in {0,1} or y in {-1,1} to dataset with y in {[1,0], [0,1]}, then call multinonimal logistic regression on this

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 explicitly enforce this condition in our Lagrangian)

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 external & internal CV

Algorithm 3: 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 $\mathcal{D}_{\text{tr}}^{(i)} = \mathcal{D} - \mathcal{D}_{\text{te}}^{(i)}$
    $f_i \leftarrow$ **Learner**$(\mathcal{D}_{\text{tr}}^{(i)})$
    err-f $=$ err-f$+$ error of f_i on $\mathcal{D}_{\text{te}}^{(i)}$
err-f $=$ err-f$/k_{\text{external}}$
$f_i \leftarrow$ **Learner**$(\mathcal{D}_{\text{tr}}^{(i)})$
return f and err-f

Algorithm 4: Learner using cross-validation on a dataset $\mathcal{D}'$

Partition the dataset $\mathcal{D}'$ 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 $\mathcal{D}'^{(j)}_{\text{tr}} = \mathcal{D}' - \mathcal{D}'^{(j)}_{\text{te}}$
        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 retu
Train $f = \text{Alg}(\mathcal{D}, h^*)$
return f

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
  - We have a massive dataset, so can make train, validation and test big
  - And sometimes compromise if our model is very expensive to train, making CV impractical to use

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?
  - Validation is used for hyperparameter selection (role of internal CV)
  - Test is used for evaluation of the final function (role of external CV)
  - CV uses the whole dataset for both evaluation and training, so it actually only makes splits for evaluation, and trains on the entire dataset

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?
- Do we have to use nested cross validation? Can we use a single train-test split externally, and k-fold CV internally?

.

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?
- Do we have to use nested cross validation? Can we use a single train-test split externally, and k-fold CV internally?
  - Yes! The learner uses CV to algorithmically set its hyperparameters. We can evaluate this learner in any way we want (external CV or one train-test split)

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?
- Do we have to use nested cross validation? Can we use a single train-test split externally, and k-fold CV internally?
- Can we use a single train-validation split internally, and CV externally?

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren't there also three datasets (train, validation and test)?
- Do we have to use nested cross validation? Can we use a single train-test split externally, and k-fold CV internally?
- Can we use a single train-validation split internally, and CV externally?
  - Yes! The learner uses a validation set to algorithmically set its hyperparameters. We can evaluate this learner in any way we want.

Exercise Question

Do we have to use nested cross-validation?

- When should we use a single train-validation-test split?
- Why does CV only have train and test partitions? Why aren’t there also three datasets (train, validation and test)?
- Do we have to use nested cross validation? Can we use a single train-test split externally, and k-fold CV internally?
- Can we use a single train-validation split internally, and CV externally?
- Should we do either of these?

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
- **Will not be directly tested:**
  - Knowing specific kernels
  - The advanced remark about the representer theorem

Exercise

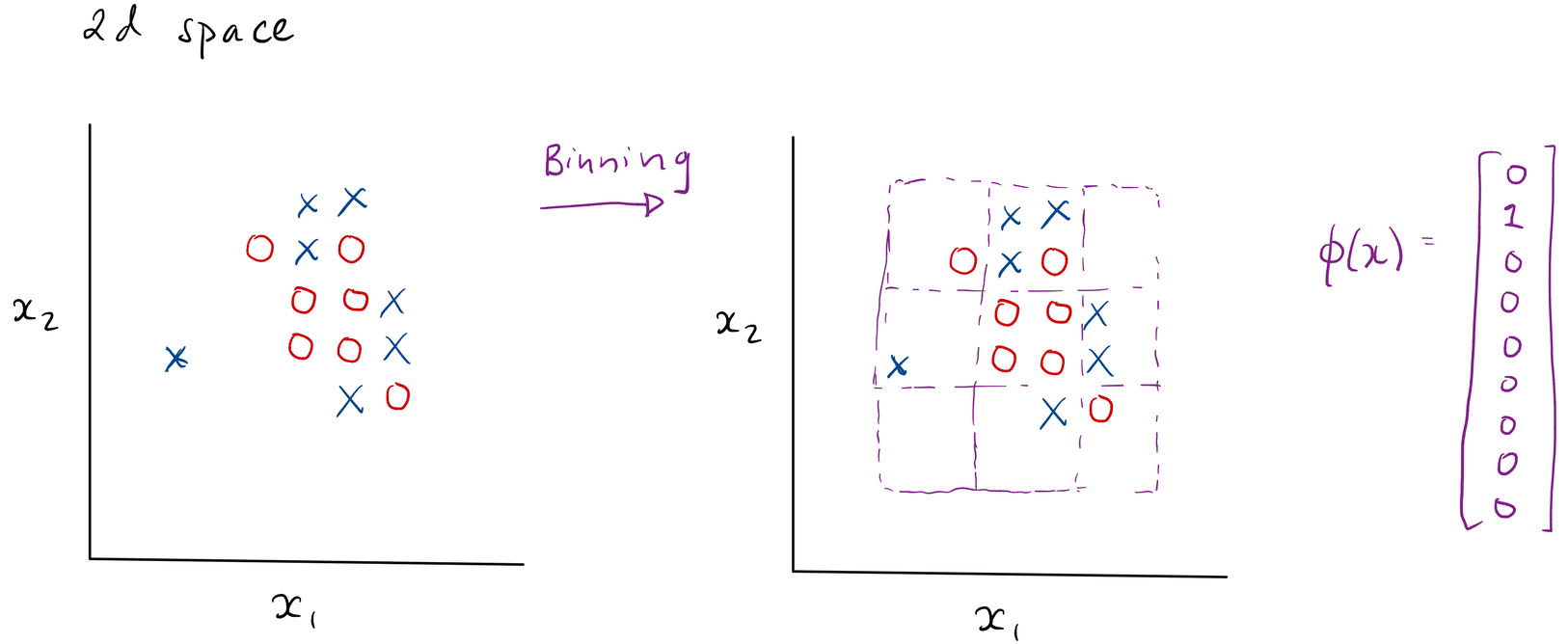
- What are the implications of using l1 regularization with polynomial features?

Exercise

- What are the implications of using l1 regularization with polynomial features?
  - Removes certain elements of the polynomial (e.g., keeps x1^3, removes x1^2, keep x1x2, removes x1^2x2)
  - Could reduce the degree of the polynomial, if removes highest order terms

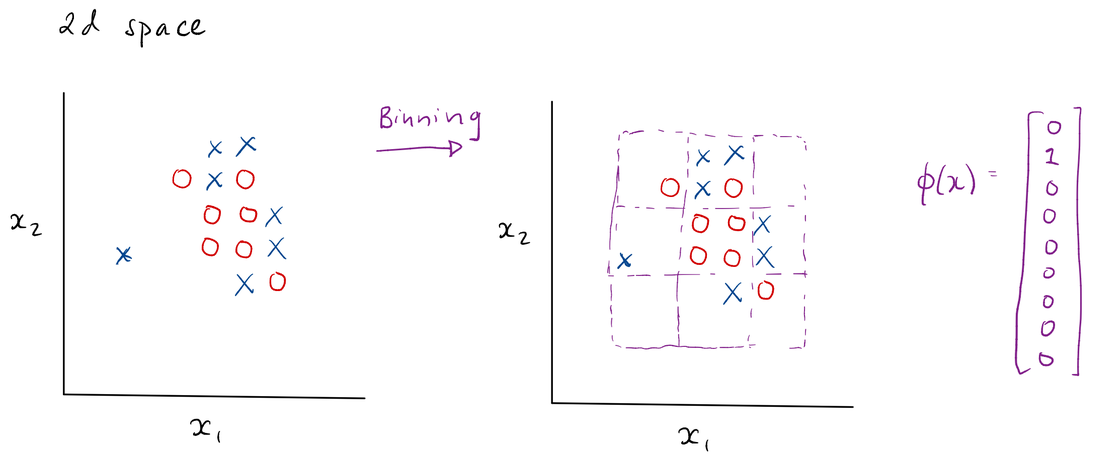
Exercise

- 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?



Exercise

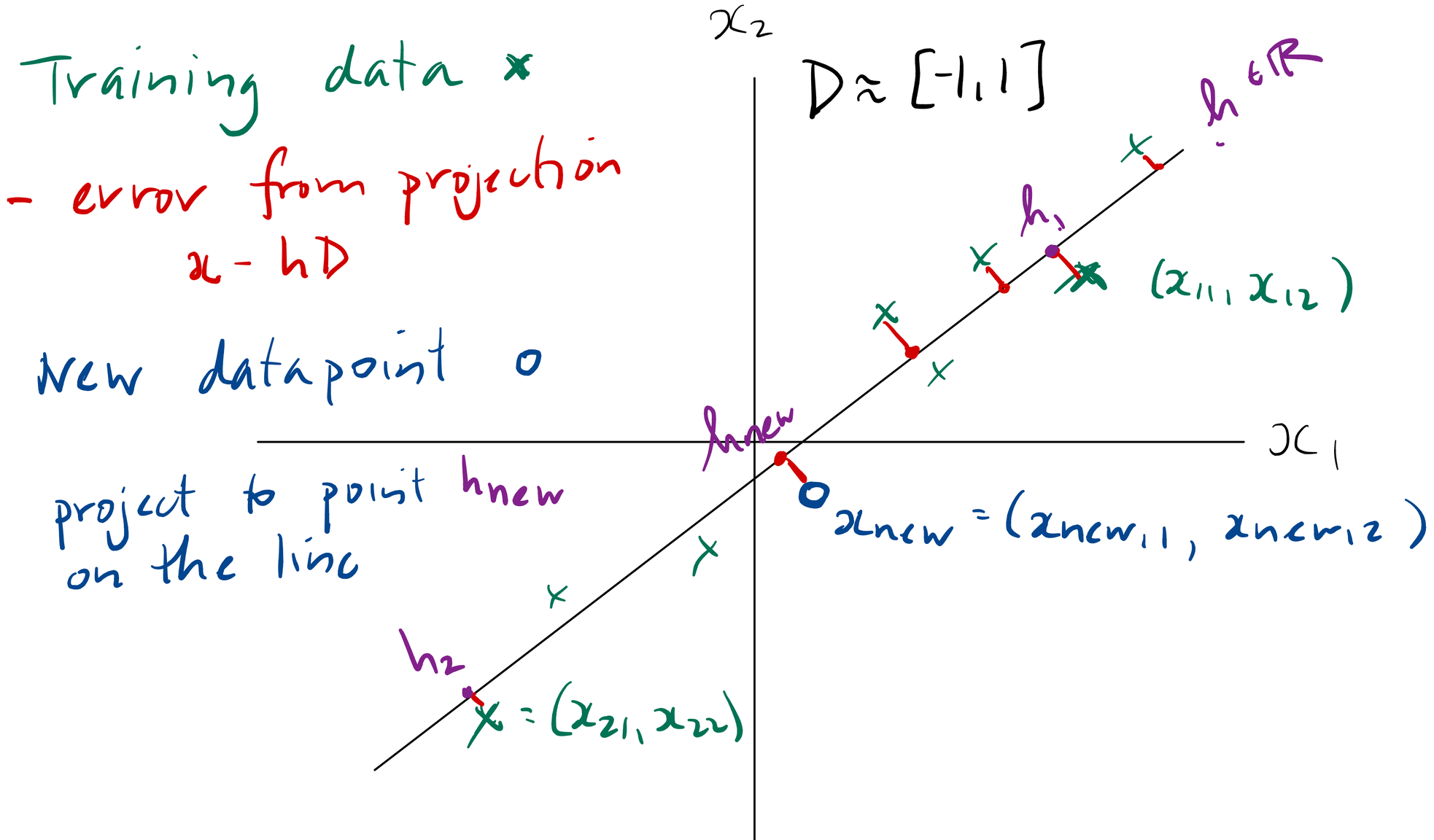
- I didn't give you an example of how projecting to higher dimensions also facilitates regression with simple (linear functions)
- Can you think of a similar example to this one, but for regression?
  - Same phi lets us learn an average y per bin
  - Features duplicated per bin can learn linear functions for each bin, giving piecewise linear functions



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 (was only for intuition about what might be learned)

PCA on training data and new points



Exercise Question

- Imagine we first expand the dimension using a kernel representation, going from 10 features to 5000.
- Imagine we compute the SVD of Phi, and the top 100 singular values are reasonably big, and the rest are relatively small.
- So we decide to use PCA to get 100 features.
- How do we interpret these 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 extract a low-dimensional rep like PCA
  - Can see nonlinear autoencoders as a nonlinear extension of PCA
- Understand that supervised autoencoders add an auxiliary loss
- **Will not be directly tested:**
  - You will not need to derive the gradients for an NN

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)? Why would these be reasonable?

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: 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)

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

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

Chapter 10: 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 $\mathbb{E}\left[(f_{\mathcal{D}}(\mathbf{x}) - f^*(\mathbf{x}))^2\right] = (\mathbb{E}\left[f_{\mathcal{D}}(\mathbf{x})\right] - f^*(\mathbf{x}))^2 + \operatorname{Var}\left[f_{\mathcal{D}}(\mathbf{x})\right].$

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

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 10: 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

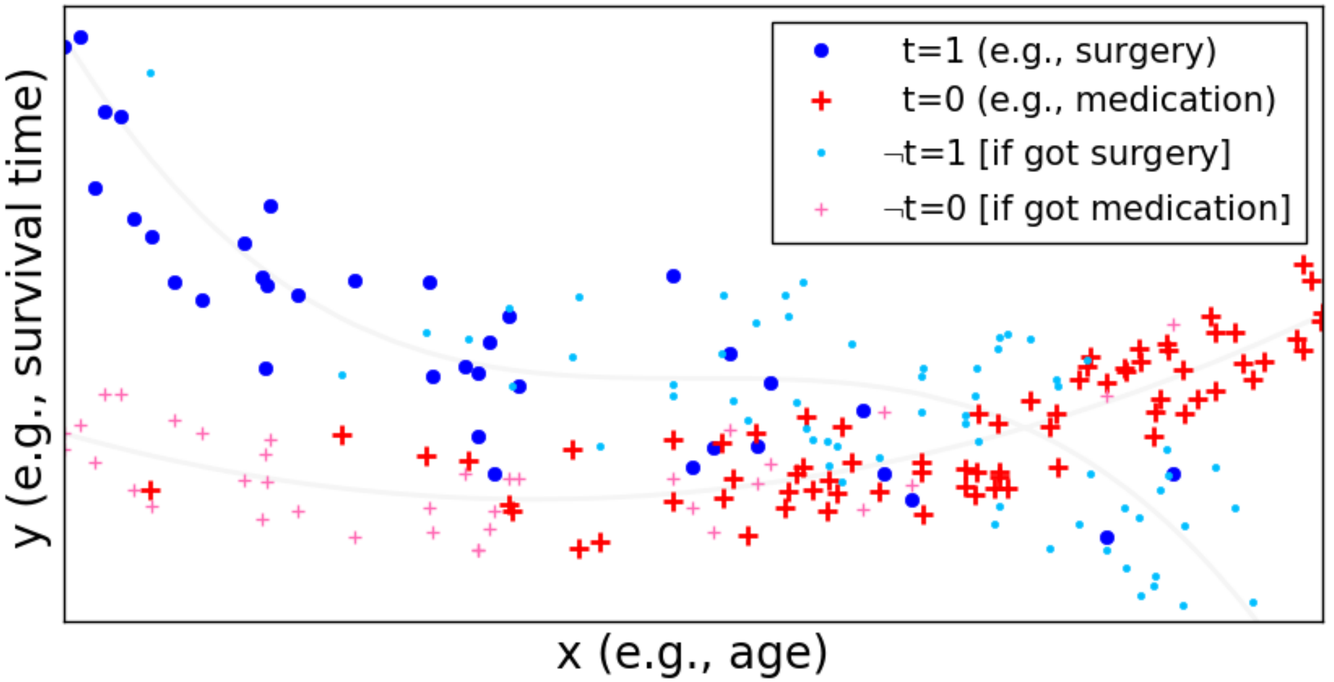


Figure 1.2: An example observational dataset (synthetic). Points in ● represent a patient who actually got surgery ($t = 1$) and indicate their respective *factual* outcome. Points in ● represent patients who in reality got medication but indicate their *counterfactual* outcome had they got surgery ($\neg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

Exercise: What is ptrain and ptest?

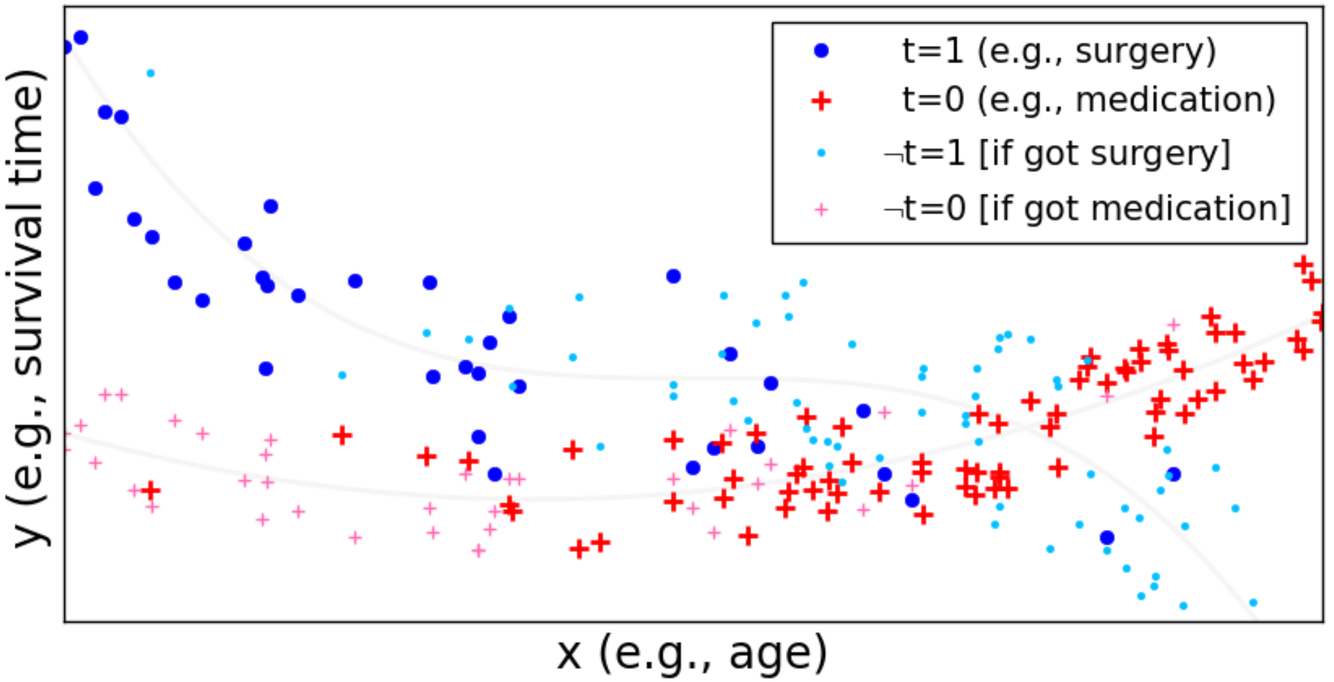


Figure 1.2: An example observational dataset (synthetic). Points in ● represent a patient who actually got surgery ($t = 1$) and indicate their respective *factual* outcome. Points in ● represent patients who in reality got medication but indicate their *counterfactual* outcome had they got surgery ($\neg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

Chapter 10: 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

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

Exercise

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

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

Exercise

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

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

- 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 10: Bias, Variance and Generalization Error (cont)

- Most of the rest of Chapter 10 will not be directly tested
- **Will not be directly tested**
  - Implicit regularization and double descent (10.2)
  - I only expect you to know what covariate shift is; I will not test on understanding how to fix covariate shift (e.g., importance sampling)
  - Nonstationary in p(y|x) (10.3.3)

Chapter 11: Mixture Models

- Understand how we sample from mixture models
- Understand that the EM algorithm consists of (a) the introduction of auxiliary variables z (component) and (b) alternating between updating $p(z_i \mid x_i)$ and parameters $\boldsymbol{\theta}$
- Understand that the E-step updates $p(z_i \mid x_i)$ for fixed $\boldsymbol{\theta}$, and the M-step updates $\boldsymbol{\theta}$ for fixed $p(z_i \mid x_i)$ with each component updated independently using a (weighted) log-likelihood
- **Will not be directly tested:**
  - You do not need to memorize the EM algorithm, but you should be able to recognize key components of it

EM Algorithm

Algorithm 8: EM for Gaussian Mixture Models

Input: number of components m

Initialize $\boldsymbol{\mu}_k^{(0)}$, $\boldsymbol{\Sigma}_k^{(0)}$ and $w_k^{(0)}$ for all $k \in 1$ to m, $t = 0$

while not converged **do**

$$p_t[i,k] = \frac{w_k^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t)})}{\sum_{j=1}^{m} w_j^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_j^{(t)})} \text{ for all } i \in \{1,2,\ldots,n\},\ k \in \{1,2,\ldots,m\}$$

Compute $p_t[k] \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} p_t[i,k]$

for $k \in \{1,2,\ldots,m\}$ **do**

$$w_k^{(t+1)} = p_t[k]$$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{np_t[k]}\sum_{i=1}^{n} p_t[i,k]\mathbf{x}_i$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{np_t[k]}\sum_{i=1}^{n} p_t[i,k](\mathbf{x}-\boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}-\boldsymbol{\mu}_k^{(t+1)})^\top$$

$t \leftarrow t+1$

return $w_k^t, \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}$ for all $k \in \{1,2,\ldots,m\}$

EM Algorithm for any component distribution

Algorithm 9: EM for any component distribution

Input: number of components m, with components distributions $p(\cdot|\boldsymbol{\theta}_1), \ldots, p(\cdot|\boldsymbol{\theta}_m)$

Initialize $\theta_k^{(0)}$ and $w_k^{(0)}$ for all $k \in 1$ to m, $t = 0$

while not converged **do**

$$p_t[i,k] = \frac{w_k^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t)})}{\sum_{j=1}^{m} w_j^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_j^{(t)})} \text{ for all } i \in \{1,2,\ldots,n\},\ k \in \{1,2,\ldots,m\}$$

Compute $p_t[k] \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} p_t[i,k]$

for $k \in \{1,2,\ldots,m\}$ **do**

$$w_k^{(t+1)} = p_t[k]$$

$$\boldsymbol{\theta}_k^{(t+1)} = \text{argmin}_{\boldsymbol{\theta}_k} - \sum_{i=1}^{n} p_t[i,k] \ln p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

$t \leftarrow t + 1$

return $w_k^t, \boldsymbol{\theta}_k^{(t)}$ for all $k \in \{1,2,\ldots,m\}$

Exercise

- What is the algorithm if we use a mixture of exponential distributions? Recall exponential pdf is $p(x) = \lambda \exp(-\lambda x)$.

Algorithm 9: EM for any component distribution

Input: number of components m, with components distributions $p(\cdot|\boldsymbol{\theta}_1), \ldots, p(\cdot|\boldsymbol{\theta}_m)$

Initialize $\theta_k^{(0)}$ and $w_k^{(0)}$ for all $k \in 1$ to m, $t = 0$

while not converged **do**

$p_t[i,k] = \frac{w_k^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t)})}{\sum_{j=1}^m w_j^{(t)} p(\mathbf{x}_i|\boldsymbol{\theta}_j^{(t)})}$ for all $i \in \{1,2,\ldots,n\}$, $k \in \{1,2,\ldots,m\}$

Compute $p_t[k] \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n p_t[i,k]$

for $k \in \{1,2,\ldots,m\}$ **do**

$w_k^{(t+1)} = p_t[k]$

$\boldsymbol{\theta}_k^{(t+1)} = \text{argmin}_{\boldsymbol{\theta}_k} - \sum_{i=1}^n p_t[i,k] \ln p(\mathbf{x}_i|\boldsymbol{\theta}_k)$

$t \leftarrow t+1$

return $w_k^t, \boldsymbol{\theta}_k^{(t)}$ for all $k \in \{1,2,\ldots,m\}$

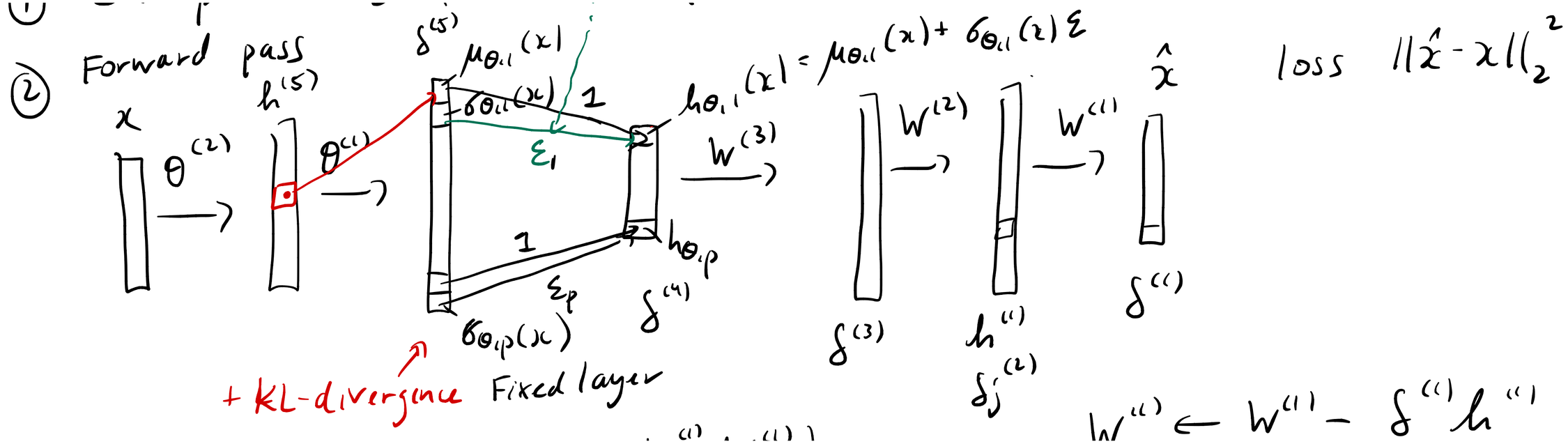
Chapter 12: Generative Models & Data Representations

- 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
  $p(\mathbf{x}\,|\,\mathbf{h}) = \mathcal{N}(f_{\mathbf{W}}(\mathbf{h}), \sigma^2\mathbf{I})$
- Understand how to sample from a VAE
  - Step 1: Sample $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then
  - Step 2: Return $f_{\mathbf{W}}(\mathbf{h})$ (where $f_{\mathbf{W}}$ is the decoder part of the network)

Chapter 12: VAEs (cont)

- Understand our goal is to minimize $\sum_{i=1}^{n} -\ln p(\mathbf{x}_i \,|\, \mathbf{W})$
  - But that this is hard to because $p(\mathbf{x}_i \,|\, \mathbf{W})$ involves an integral over hidden $\mathbf{h}$
- Understand that we learn the encoder $q(\mathbf{h} \,|\, \mathbf{x})$ only as part of the optimization, to help us learn $p(\mathbf{x} \,|\, \mathbf{h})$; we do not need $q(\mathbf{h} \,|\, \mathbf{x})$ itself
- Understand that we derive the VAE objective (the negative ELBO) by starting with $-\ln p(\mathbf{x} \,|\, \mathbf{W}) + D_{KL}(q(\,\cdot\,|\,\mathbf{x}) \,||\, p(\,\cdot\,|\,\mathbf{x}, \mathbf{W}))$ and rearranging terms to get $D_{KL}(q(\,\cdot\,|\,\mathbf{x}) \,||\, \mathcal{N}(0, \mathbf{I})) - \mathbb{E}_{\mathbf{h} \sim q(\cdot|\mathbf{x})}[\ln p(\mathbf{x} \,|\, \mathbf{h}, \mathbf{W})]$
- Understand why we use the reparameterization trick to get the gradient

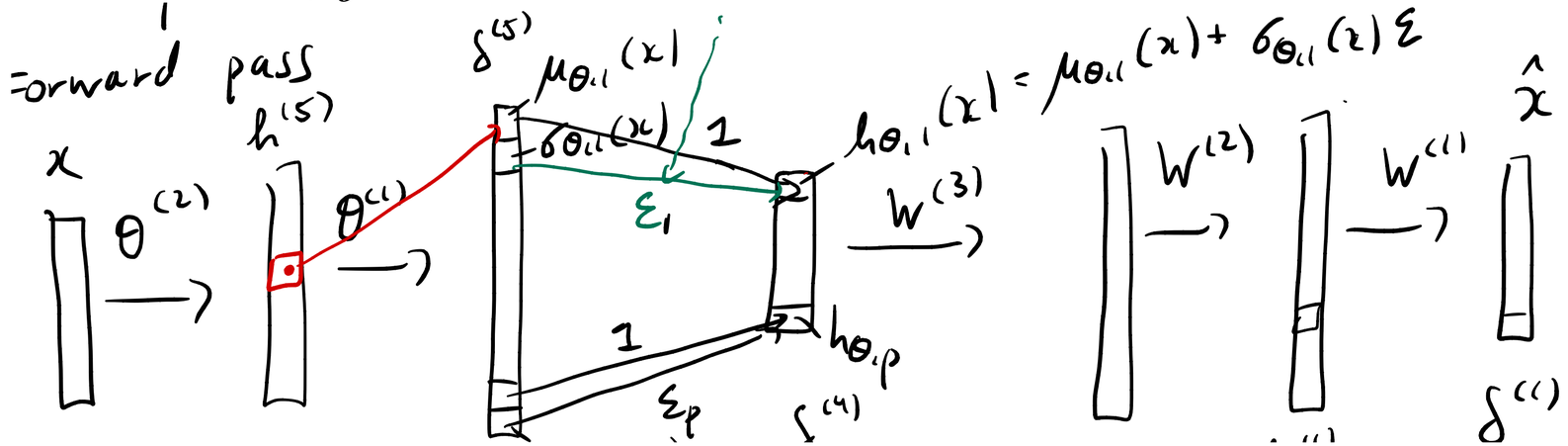
Exercise: which part is q and p?



- Recall we parameterized our encoder as $q(\mathbf{h} \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(f_{\mu,\theta}(\mathbf{x}), f_{\sigma,\theta}(\mathbf{x}))$

Exercise: Reparameterization trick

- Recall we parameterized our encoder as $q(\mathbf{h} \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(f_{\mu,\theta}(\mathbf{x}), f_{\sigma,\theta}(\mathbf{x}))$
- We used reparameterization
  $\mathbb{E}_{\mathbf{h}\sim q(\cdot|\mathbf{x})}[\ln p(\mathbf{x} \mid \mathbf{h}, \mathbf{W})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}[\ln p(\mathbf{x} \mid h_j = \mu_j(\mathbf{x}) + \sigma_j(\mathbf{x})\boldsymbol{\epsilon}_j, \mathbf{W})]$
- Q1: What is the dimension of $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$? How is this sampled?
- Q2: What if we decided we wanted discrete hidden h, like mixtures, and used $q(\mathbf{h} \mid \mathbf{x}, \boldsymbol{\theta}) = \text{softmax}(f_\theta(\mathbf{x}))$? How does the NN change? Can we still use reparam?



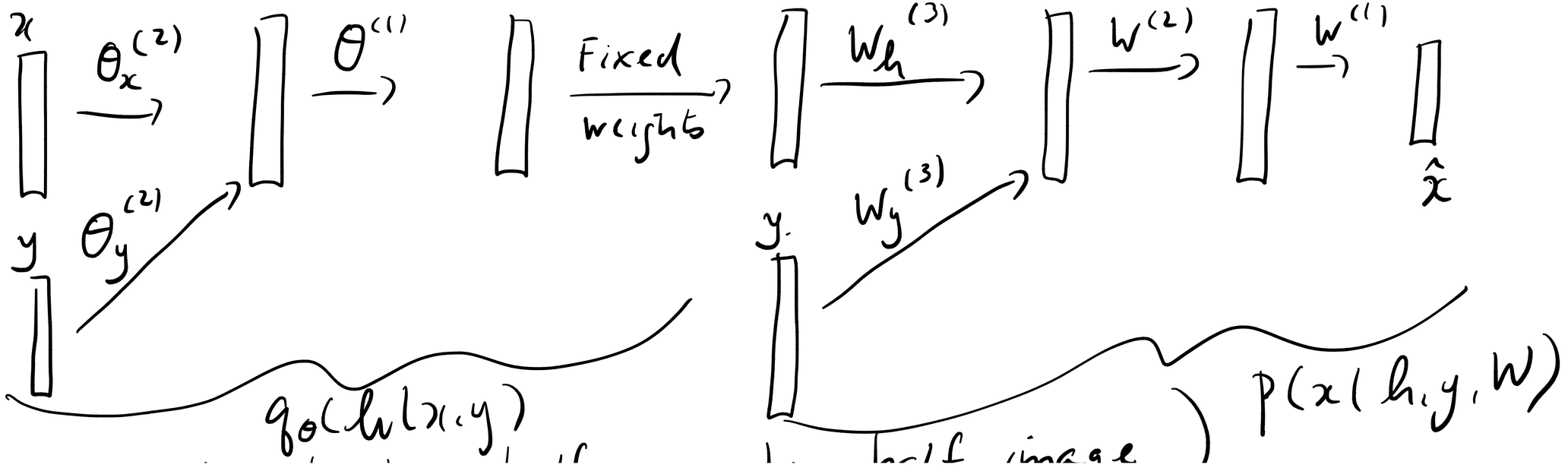
Exercise

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

Exercise: Why do we do this complicated alg with VAEs?

- Can’t we just train an auto-encoder, and then sample h to use the decoder part?

Chapter 12: Conditional VAEs



Chapter 12: VAEs (cont)

- **Will not be directly tested**
  - Memorizing the VAE objective (I will give you formulas)
  - Memorizing the gradient update for the VAE
  - The connection to Expectation-Maximization (9.3)
  - Deriving the gradient with the reparameterization trick
  - Mixture density networks

Chapter 13: Evaluating Generative Models

- Understand goal is to estimate $\hat{\text{GE}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{x_i \in \mathcal{D}_{\text{test}}} -\ln p(\mathbf{x}_i | \boldsymbol{\theta})$
- Understand that can easily compute this for a mixture model, as long as the components are ones where it is easy to compute $-\ln p(\mathbf{x}_i | \boldsymbol{\theta}_k)$
- Understand that this is hard to compute for VAEs, must be estimated
- **Will not be directly tested**
  - importance sampling approach to estimate this GE for VAEs

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)

Exercise

- Can we take a mixture over VAE components?

Exercise

- Can we take a mixture over VAE components?
  - Yes! Why not? Seems fun, you should try it. Go back to the MM algorithms and see if you can figure out how to do it
- Is this still easy to evaluate, in terms of negative log-likelihood?

Chapter 14: Missing Data

- Understand how to do imputation using PCA (matrix factorization), including training with missing data and using the model for new data
- Understand how to do imputation using an autoencoder, including training with missing data and using the model for new data
- Understand the difference between the two stage approach (impute then hand to learning algorithm) versus the direct approach (missingness indicator)
- **Will not be directly tested**
  - Connections to the transductive and semi-supervised settings

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

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 get back the same h's and D?
  - No. But we do if we minimize over $\mathbf{x}_{\mathscr{M}_i}$

More general loss we use

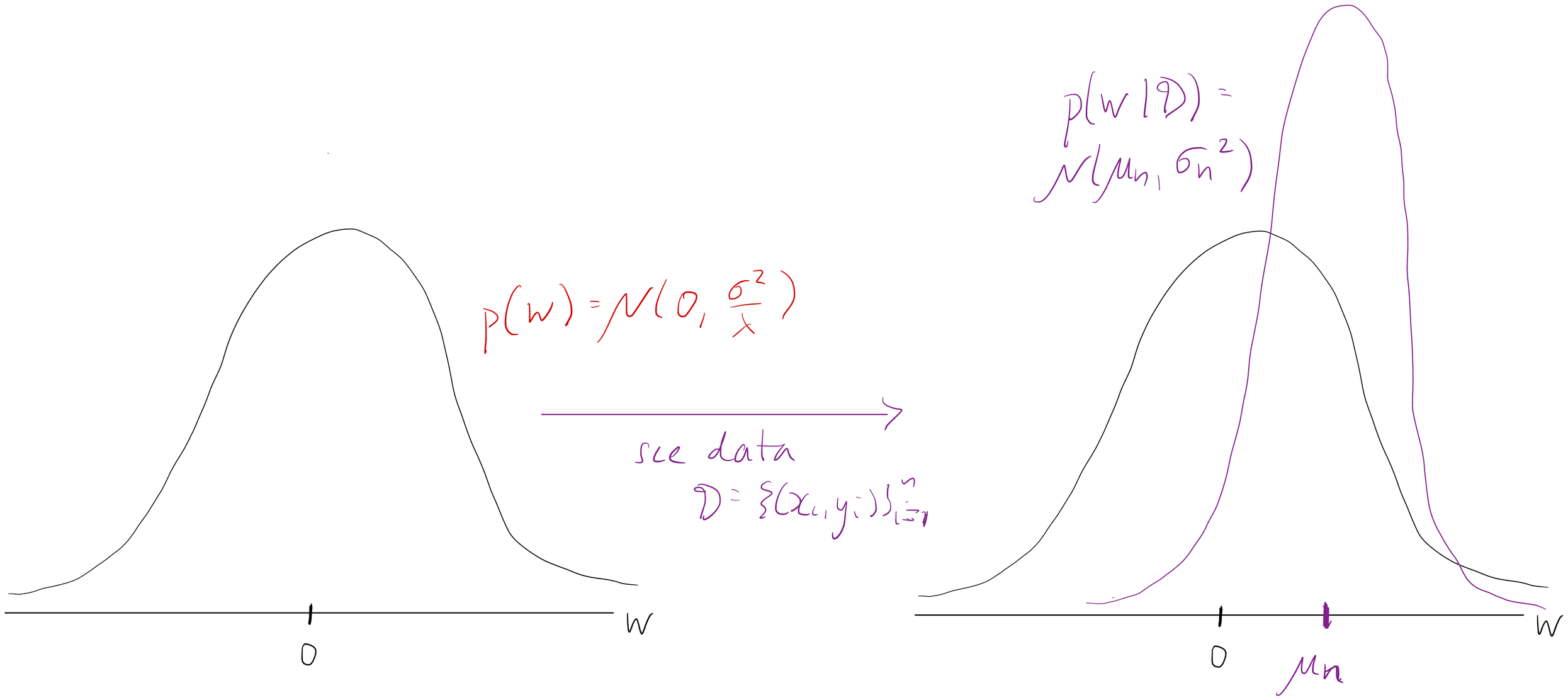
- In PCA we solve for $\min_{\mathbf{x}_{1,\mathscr{M}_1},\ldots,\mathbf{x}_{n,\mathscr{M}_n}} \min_{\mathbf{h}_1,\mathbf{h}_2,\ldots,\mathbf{h}_n \in \mathbb{R}^p, \mathbf{D} \in \mathbb{R}^{p \times d}} \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - \mathbf{h}_i \mathbf{D}_{:j})^2$
- For autoencoder, $\min_{\mathbf{x}_{1,\mathscr{M}_1},\ldots,\mathbf{x}_{n,\mathscr{M}_n}} \min_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - f_{\mathbf{W}}(\mathbf{x}_i))^2$
- We need this more general purpose loss for the autoencoder, because we cannot use objective. Why?

  $\min_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j \in \mathscr{A}_i} (x_{ij} - f_{\mathbf{W}}(\mathbf{x}_i))^2$

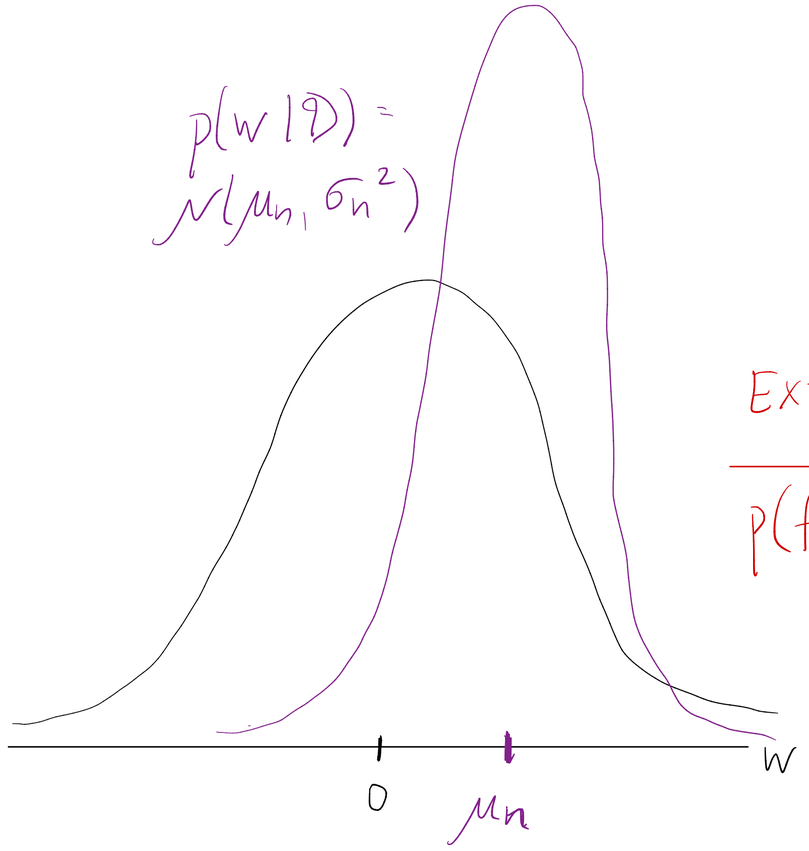
Chapter 15: Uncertainty estimation

- Understand that we might want to know distribution over plausible values of $\mathbf{w}$, given the evidence (data)
- Understand that 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
- **Will not test you on**
  - Memorizing the formulas for Bayesian linear regression
  - The Normal-inverse gamma distribution

Shrinking posterior

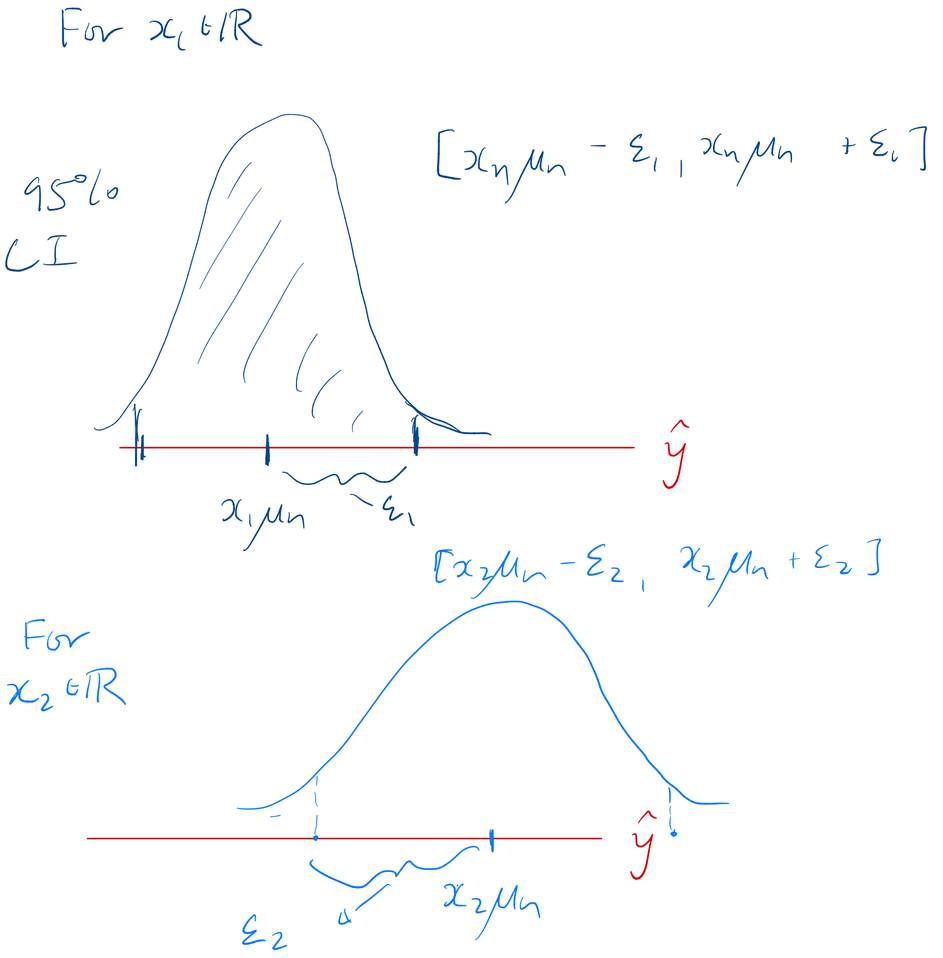


Credible Interval for Predictions



Extract

$p(f_w(x)|D)$



Chapter 15: Nonlinear setting

- Understand that we can use fixed representations + Bayesian linear regression to get nonlinear regressors + credible intervals
- Understand that we can use bootstrap resampling to estimate uncertainty for neural networks
  - Understand that we sample m datasets with replacement from all the training data, and that we train an NN $f_{\mathbf{W}_k}$ from scratch on each dataset k
  - We use this ensemble of NNs to get a set of predictions to compute intervals for our predictions, $\hat{y}_1 = f_{\mathbf{W}_1}(\mathbf{x}), \ldots, \hat{y}_m = f_{\mathbf{W}_m}(\mathbf{x})$
  - Compute Gaussian (or Student's t) confidence interval on $\hat{y}_1, \ldots, \hat{y}_m$
- **Will not test:** smarter confidence interval strategies than Gaussian/Student's t

All later material not tested

- Will not test you Gaussian Processes nor the kernel trick
- Will not test you on handling temporal data (Chapter 16)