Midterm Review

CMPUT 367: Intermediate Machine Learning

Comments

- Midterm on Chapters 1 - 9 (up to and including neural networks)
- The goal of the exam is to test (a) did you understand the basic ideas and (b) can you apply that understanding
- Answers can be relatively short, say at most 5 sentences
- I have added a bit more detail to the Practice Midterm, to better match the level of detail I would give the Midterm

Chapters 1-5

- Covered in Quiz Review
- Should be comfortable with
  - Generalized Linear Models
  - Basic Optimization concepts, including first and second order gradient descent, SGD, vector stepsizes and momentum
  - Basic matrix operators, including having weights that are matrices, matrix multiplication and SVD
  - The role of l2 regularization (in any GLM) and the bias-variance trade-off in linear regression
  - Understand why (and when) we might use SGD and GD, as well as first-order versus scond-order GD

A few comments from quiz errors

- SGD means mini-batch SGD (not batch = 1)
  - GD or Batch GD means using the whole dataset
  - Stochastic GD (SGD) means using a stochastic estimate of the gradient with a mini-batch of size b
- There are only four GLMs we discussed: linear regression (Gaussian), Poisson regression (Poisson), logistic regression (Bernoulli) and multinomial logistic regression (multiclass with a multinomial)
  - Know these four
- Understand how to use the models we learned

How do we use GLMs?

- In a GLM we learn E[Y | x], which fully characterizes our p(y | x)
  - Bernoulli, Poisson and Multinomial all only have one key parameter, which is E[Y]
  - Gaussian has mean and variance, but we assume the variance is fixed and that we not learning it; so its key param is also only E[Y]
- How do we use GLMs for prediction?
  - Mode of p(y|x) is a reasonable answer
  - Mean E[Y | x] is also a reasonable answer

Why mode or mean?

- We will suffer a cost for our prediction
  - recall: we want to minimize expected cost
- If we picked a squared cost, then the best choice was E[Y | x]
- If we picked a 0-1 cost, then the best choice was argmax p(y | x) (mode)

Chapter 6:
Constrained Optimization

- Optimization of the form $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for smooth c, nonsmooth r
- Example: c is squared errors and r is box constraints
- Smooth means differentiable everywhere

Questions

- For the optimization $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$, what is a c and what is r for linear regression + l1 regularization?
- For the optimization $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$, what is a c and what is r for **logistic regression** + l1 regularization?
- For the optimization $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$, what is a c and what is r for linear regression + l2 regularization + l1 regularization?

Chapter 6: Constrained Optimization

- Optimization of the form $\min_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for smooth c, nonsmooth r
- Re-derived the gradient descent update, with this nonsmooth r —> Ended up with proximal gradient descent
- Proximal update:

  [Descend] $\tilde{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t)$

  [Project] $\mathbf{w}_{t+1} = \text{prox}_{\eta_t r}(\tilde{\mathbf{w}}_{t+1})$

- where $\text{prox}_{\eta_t r}(\tilde{\mathbf{w}}_{t+1}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - \tilde{\mathbf{w}}_{t+1}\|_2^2 + \eta_t r(\mathbf{w})$

Chapter 6: Constrained Optimization

- Optimization of the form $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for smooth c, nonsmooth r
- Proximal update: $\mathbf{w}_{t+1} = \text{prox}_{\eta_t r}(\mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t))$
- **Do not need to know**
  - Specific proximal operators; just need to know where to use the given proximal operator
  - How to use vector stepsizes or momentum; we only did scalar stepsizes
  - I will not get you to derive solutions with Lagrangians

Chapter 6: Constrained Optimization

- Optimization of the form $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for smooth c, nonsmooth r
- Proximal update: $\mathbf{w}_{t+1} = \text{prox}_{\eta_t r}(\mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t))$
- **You should know**
  - That we used proximal gradient descent for l1 regularization
  - That we do not always have closed-form solutions for the proximal operator, and sometimes have to solve a simple optimization to get the projection step (proximal operator), as in Section 6.3

Exercise: l1 regularization and independent features

- Imagine we have a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top$
- Imagine y is independent of x_2 and dependent on x_6
- Imagine we have 1000 samples and d = 30
- If we use l1-regularization, what might happen?
- If we don't use any regularization, what might happen?

Exercise: l1 regularization and independent features

- Imagine we have a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top$
- Imagine y is independent of x_2 and dependent on x_6
- Imagine we have 1000 samples and d = 30
- Now further imagine x_8 is accidentally a repeated feature, $x_8 = x_6$. Now what might happen when we use l1-regularization? Is the weight on x_8 likely to be zero?
- What about l2-regularization?

Chapter 7: Estimating GE and Cross Validation

- Goal is to estimate generalization error (GE) for a learned function f

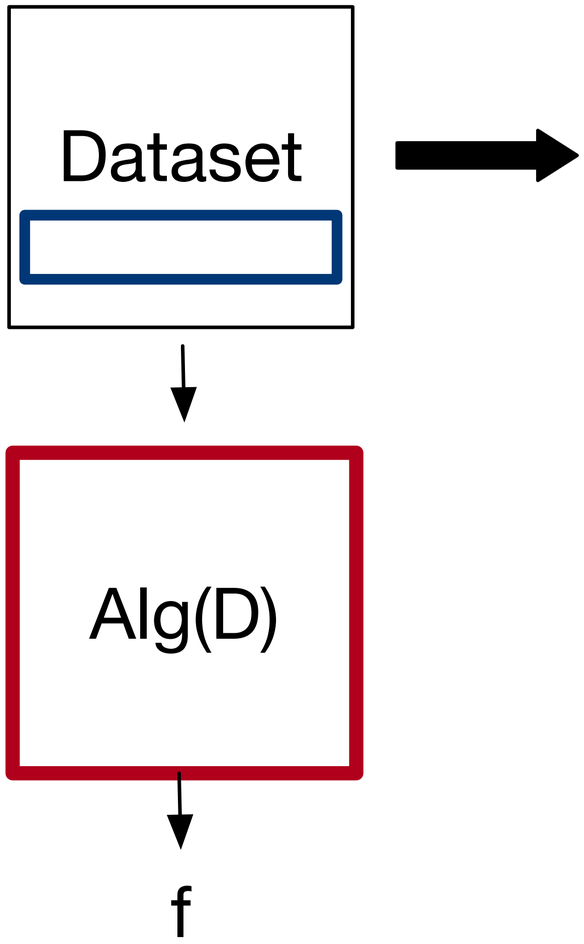
Question about GE

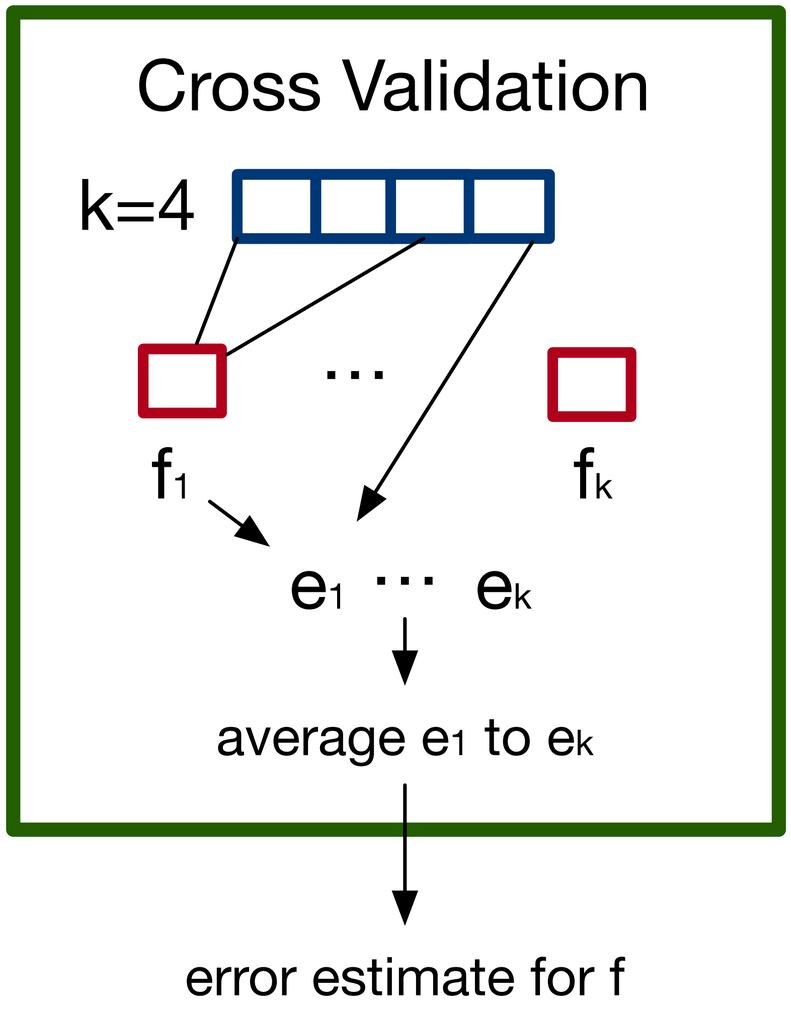
- What is the generalization error for a linear regression model?
- What is the generalization error for a logistic regression model?
- What is the generalization error for a multinomial logistic regression model?
- [Extra Q] What is the generalization error for a Poisson regression model?

Chapter 7: Estimating GE and Cross Validation

- Goal is to estimate generalization error (GE) for a learned function f
- Having a training and testing split can be data inefficient
- Cross-validation lets us use the training data for training and evaluation

Cross validation





Step 1: Learn f on the entire dataset

Step 2: Do CV to estimate the GE for f

Step 2 consists of
1. Get k partitions of the dataset, to get k training and test splits

2. For every i = 1 to k,
train $f_i = \text{Alg}(\mathscr{D}_{tr}^{(i)})$ and
compute error e_i on $\mathscr{D}_{test}^{(i)}$

3. Get average error $\frac{1}{k}\sum_i e_i$

k-fold vs RSS

- Partition means disjoint subsets that cover the data
- k-fold is one way to get partitioning
  - Partition data into k folds/chunks
  - Each fold is set to a test dataset, the training is union of the remaining folds
- Repeated random subsampling (RSS) is another way to get a partitioning
  - Randomly sample points for test dataset (without replacement), and set the rest to the training set
  - Have to specify percentage for test p and number repeats k

Selecting k

- We decided interim k (e.g., k = 10) was generally good. Why?
- We say k = 2 is likely problematic. Why?
- Why might k = n be problematic?

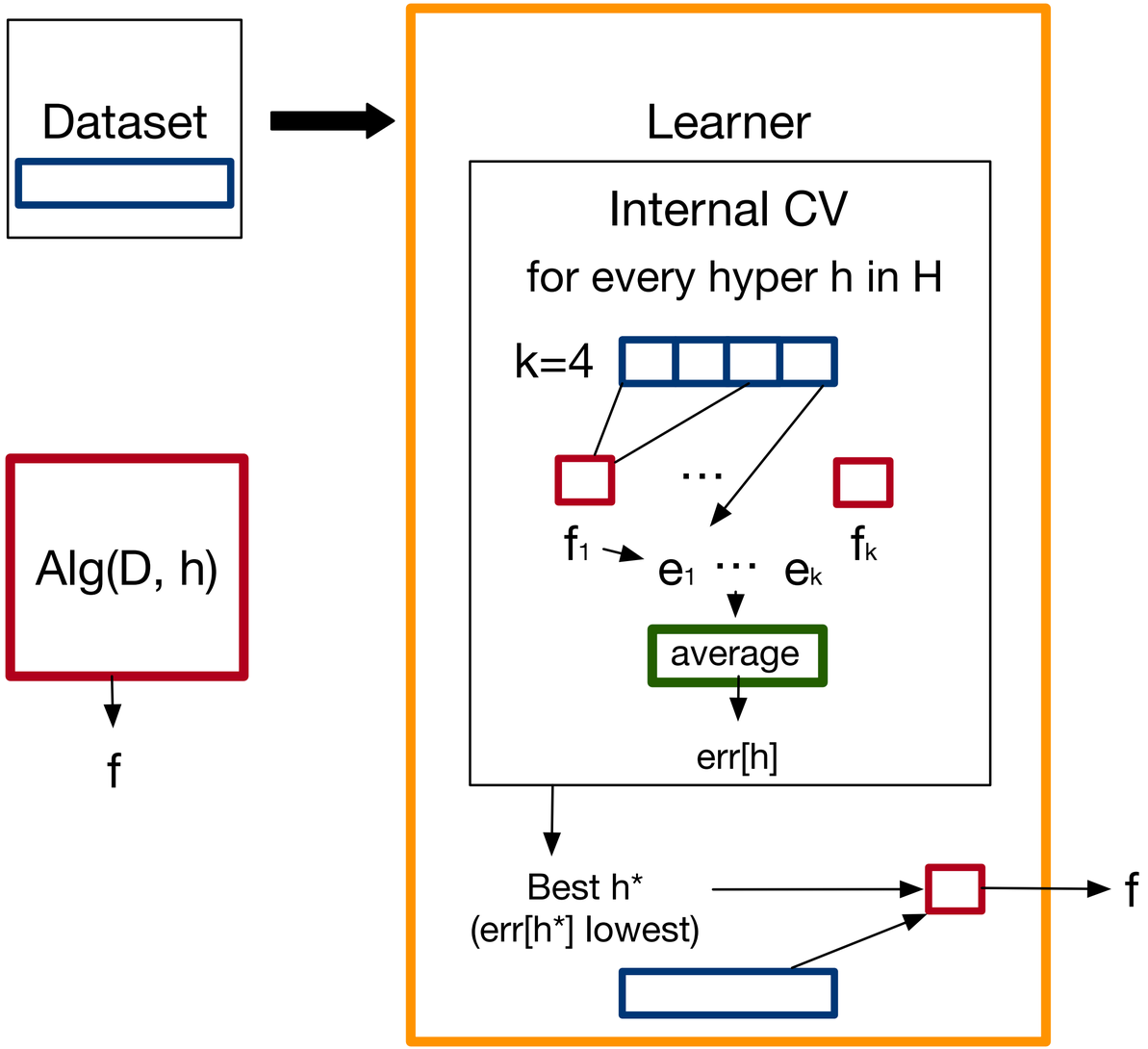
Chapter 7: Estimating GE and Cross Validation

- Goal is to estimate generalization error (GE) for a learned function f
- Having a training and testing split can be data inefficient
- Cross-validation let's us use the training data for training and evaluation
- k-fold and RSS as two partitioning approaches
- **You do not need to know**
  - All the sources of bias and variance in CV, just know that our estimator is biased and that the choice of k (and p) can impact bias and variance

Chapter 7: CV for hyperparameter selection

- Our estimate of (GE) is a good criteria to pick hyperparameters

CV for hyper selection



Chapter 7: CV for hyperparameter selection

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produce by Learner
- Can use training / validation set to evaluate it
  - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
  - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
  - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$

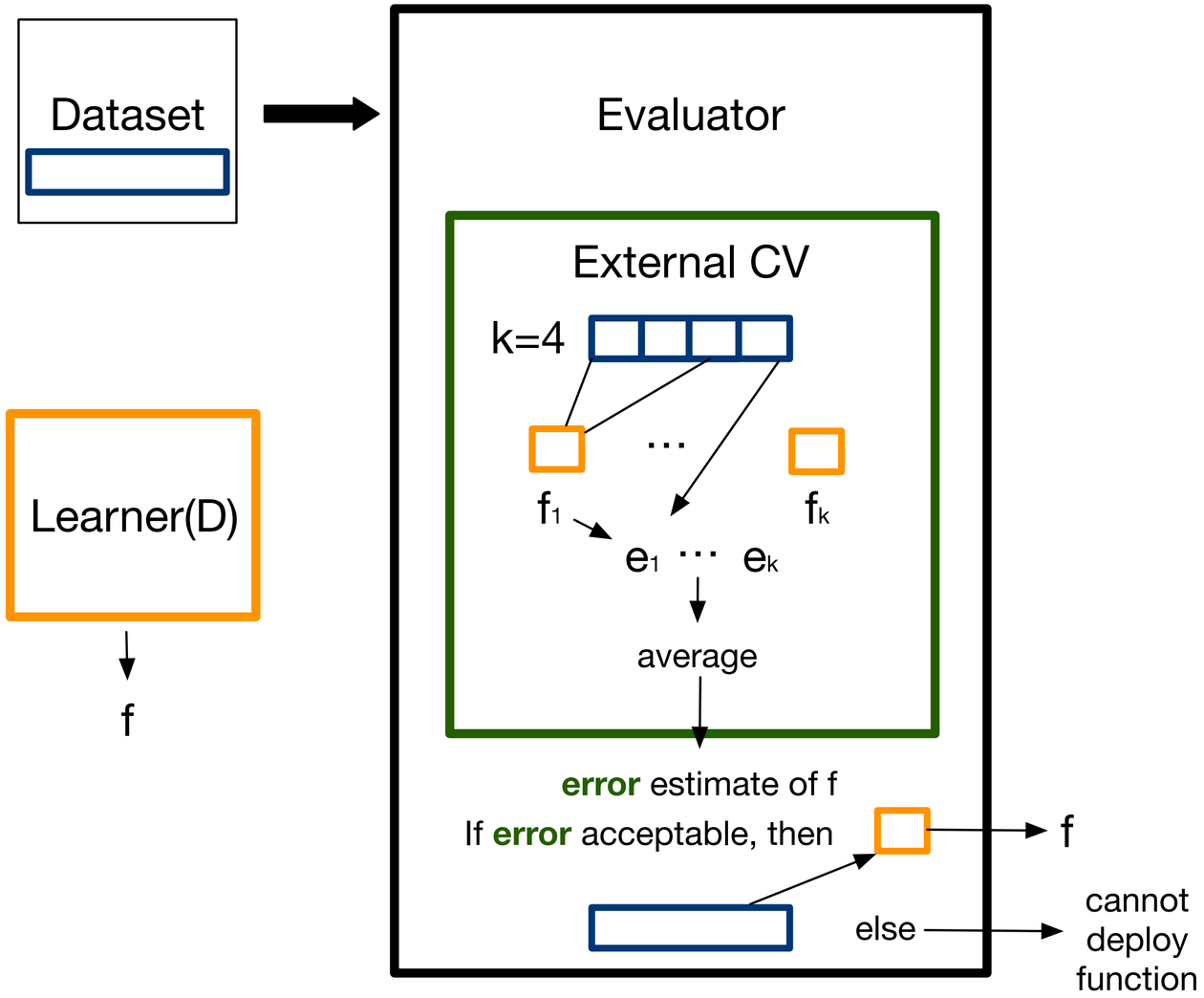
Chapter 7: CV for hyperparameter selection

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produce by Learner
- Can use training / validation set to evaluate it
  - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
  - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
  - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$
- What is the issue with this approach?

Chapter 7: CV for hyperparameter selection

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produce by Learner
- Can use training / validation set to evaluate it
  - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
  - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
  - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$
- What is the issue with this approach? Data inefficient, let's use CV!

Nest Cross-Validation



Step 1: Learn f on the entire dataset

Step 2: Do CV to estimate the GE for f

Step 2 consists of
1. Get k partitions of the dataset, to get k training and test splits

2. For every i = 1 to k,
train $f_i = \text{Alg}(\mathscr{D}_{tr}^{(i)})$ and
compute error e_i on $\mathscr{D}_{test}^{(i)}$

3. Get average error $\frac{1}{k}\sum_i e_i$

Chapter 8: Fixed Representations

- We discussed polynomials, RBF Networks and Prototype representations
- Question: what is the difference between RBF Networks and Prototype representations that use an RBF kernel?

Chapter 8: Fixed Representations

- We discussed polynomials, RBF Networks and Prototype representations
- Question: what is the difference between RBF Networks and Prototype representations that use an RBF kernel?
- Answer: we can see this Prototype Rep + RBF kernel as an instance of an RBF network where the centers are prototypes

Chapter 8: Fixed Representations

- We discussed polynomials, RBF Networks and Prototype representations
- We discussed how l1 regularization allows us to subselect prototypes
- **You do not need to know**
  - Any representability results for these functions
  - You just need to know that they let us learn nonlinear functions

Exercise: why don't we use proximal methods with l0 regularization?

- l0 regularization ($\|\mathbf{w}\|_0$) counts number of entries that are non-zero
- We could set r to l0 for the following optimization $\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$

Exercise: why don't we use proximal methods with l0 regularization?

- l0 regularization ($\|\mathbf{w}\|_0$) counts number of entries that are non-zero
- We could set r to l0 for the following optimization $\min_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$
- But hard to solve for $\text{prox}_{\eta_t \ell_0}(\tilde{\mathbf{w}}_{t+1}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - \tilde{\mathbf{w}}_{t+1}\|_2^2 + \eta_t\|\mathbf{w}\|_0$
- Proximal GD doesn't solve all of our problems, only those where the proximal operator is easy to compute

Chapter 9
Learning Latent Factors

- Understand that PCA extracts a lower-dimensional representation $\mathbf{h}$ for $\mathbf{x}$
- Understand that sparse coding extracts a higher-dimensional, sparse representation $\mathbf{h}$
- Understand that for both we are trying to solve $\mathbf{x} \approx \mathbf{hD}$
- For both we try to minimize $||\mathbf{x} - \mathbf{hD}||_2^2$ for all $\mathbf{x}$, but for sparse coding we additionally add a sparsity regularizer to $\mathbf{h}$, namely $||\mathbf{h}||_1$

Exercise about PCA

- We minimized $\sum_i \|\mathbf{x}_i - \mathbf{h}_i\mathbf{D}\|_2^2$ for PCA, with $\mathbf{h}_i \in \mathbb{R}^p$ for $p < d$
- What would happen if we let $p \geq d$?
- Why isn't this a problem for sparse coding?

Chapter 9
Learning Latent Factors

- Understand that PCA extracts a lower-dimensional representation $\mathbf{h}$ for $\mathbf{x}$
- Understand that sparse coding extracts a higher-dimensional, sparse representation $\mathbf{h}$
- **You do not need to know**
  - The exact formulas for the optimizations; I will give them to you. But you should know how to reason about minimizing them
  - You do not need to know the probabilistic PCA solution, nor the closed-form PCA solution

Exercise Question

- Imagine we have 5000 datapoints for a problem with d = 10
- 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?

Chapter 9:
Learning Neural Networks

- Understand types of transformation on the input given by a neural network
  - series of linear functions composed with simple activations
- Understand that backpropagation is gradient descent
- Understand that linear autoencoders also extract a low-dimensional representation like 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: NN choices

- An NN with three layers transforms the inputs as
  - $f_{\mathbf{w}}(\mathbf{x}) = f_1(f_2(f_3(\mathbf{x}\mathbf{W}^{(3)})\mathbf{W}^{(2)})\mathbf{W}^{(1)})$ for weights $\mathbf{w}$ composed of $\mathbf{W}^{(3)}, \mathbf{W}^{(2)}, \mathbf{W}^{(1)}$
- Can think of this NN as learning p(y | x) with key parameter $\theta(\mathbf{x}) = \mathbf{h}^{(1)}\mathbf{W}^{(1)}$ for $\mathbf{h}^{(1)} = f_2(f_3(\mathbf{x}\mathbf{W}^{(3)})\mathbf{W}^{(2)})$
- We pick a GLM loss for the output that matches the targets
  - e.g., what if the output is a binary 0,1 variable? What is f1?
  - e.g., what if the output is ordinal 0, 1, 2, 3, 4, 5, .., 100? What is f1?

Exercise: NN vs PCA

- An NN with three layers transforms the inputs as
  - $f_{\mathbf{w}}(\mathbf{x}) = f_1(f_2(f_3(\mathbf{x}\mathbf{W}^{(3)})\mathbf{W}^{(2)})\mathbf{W}^{(1)})$ for weights $\mathbf{w}$ composed of $\mathbf{W}^{(3)}, \mathbf{W}^{(2)}, \mathbf{W}^{(1)}$
- Can think of this NN as learning p(y | x) with key parameter $\theta(\mathbf{x}) = \mathbf{h}^{(1)}\mathbf{W}^{(1)}$ for $\mathbf{h}^{(1)} = f_2(f_3(\mathbf{x}\mathbf{W}^{(3)})\mathbf{W}^{(2)})$
- Can think of $\mathbf{h}^{(1)}$ as the new representation of $\mathbf{x}$. How do we extract the new representation for a new $\mathbf{x}_{\text{new}}$?
- How do we do this for PCA?