# Final, CS-GY 6923

## Basic Info

**Date:** Friday 12/20 from 2pm - 4pm
**Location:** Our usual classroom
**Supplies needed:** Pencil or pen. I will provide scrap paper.
**Supplies allowed:** One double-sided sheet of paper containing whatever you like. Equations, notes, examples, etc. This can be handwritten or typewritten – there are no restrictions.

## Preparation

The best way to prepare is to review your homework and labs. If there are any homework problems you did not understand or fully solve, please ask about them in office hours.

## Question Types

Similar to the midterm:

- Questions similar to the homework.
- True/false, with a short statement to justify your answer.
- "Always, sometimes, never" questions.

# Topics

Below are a list of things you should know, roughly categorized into topics.

## First half of course

We won't be specifically tested on material from the first half of the course.

## Optimization and Gradient Descent

- Why do we need to use search methods to minimize loss functions instead of just figuring out when the gradient equals $\vec{0}$.
- How would you do a brute force search for the optimal parameters $\boldsymbol{\beta}$ of a model? Why would this be slow when $\boldsymbol{\beta}$ has more than a few dimensions?
- The gradient descent update is $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla L(\beta)$ where $\eta$ is the *stepsize*, aka *learning rate*, parameter.

- $\lim_{\eta \to 0} L(\boldsymbol{\beta} + \vec{v}) - L(\boldsymbol{\beta}) = \eta \langle \nabla L(\boldsymbol{\beta}), \vec{v} \rangle$.
- Why is the gradient descent update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla L(\beta)$ guaranteed to decrease $L(\boldsymbol{\beta})$ for a sufficiently small step-size $\eta$?
- Why is gradient descent sometimes called *steepest descent*? Understand why $\nabla L(\boldsymbol{\beta})$ is the "best" update.
- What is the definition of a convex function? Be able to prove or argue why a basic function like $f(x) = x^2$ is convex. What stronger guarantees can we prove for gradient descent when our function is convex? Convergence to local minimum, number of steps depending on parameters of the function.
- What is stochastic gradient descent and what functions can it be applied to? Be able to compute the cost of computing a full gradient and/or a stochastic gradient for various loss functions. SGD trades cheaper iterations for slower convergence.

## Learning Theory

- What is a hypothesis class?
- What is the statistical learning model?
- What is PAC (probably approximately correctly learning)?
- What does it mean when we assume we are in the "realizable setting"?
- What is the union bound? Be able to state and use it.
- You should be able to fully understand the proof of the main sample complexity theorem on Slide 45 Lecture 7.
- Be able to bound the size of finite hypothesis classes like we did on the homework.

## Kernel Methods

- What is a k-NN classifier?
- The inner product is a natural similarity measure between two data points. What are other similarity measures?
- What is a positive semidefinite kernel function? What are some examples?
- Be able to prove a simple kernel is positive semidefinite as we did on the homework.
- Know the reformulated versions of least squares regression and logistic regression that accommodate a non-linear kernel.

## Support Vector Machines

- What is a separating hyperplane? How is it mathematically defined? What is margin?
- Know the hard margin SVM objective. What is the definition of a support vector?
- Soft-margin SVM objective. How does changing the parameter $C$ impact the final separating hyperplane?
- How is the soft-margin objective related to logistic regression?

## Neural Networks Basics

- Understand why neural networks are so powerful (automatic feature learning).
- What are some examples of non-linearities? Why do neural networks always involve a non-linearity between layers? Without a non-linearity, the layers could be collapsed into a single equivalent layer.
- Understand different diagrams of neural networks and what they mean.
- What is the multivariate chain rule? Be able to apply it.
- Be able to carry out a full-backpropagation calculation as in the homework.
- Understand the computational complexity of backpropagation (linear in number of neural network parameters).

## Convolutional Neural Networks

- How is 1D, 2D, and 3D convolution defined? Be able to manually perform a simple convolution or recognize the output of a convolution.
- How can convolutions be used for edge detection? For pattern matching?
- Understand that convolutional networks can be viewed as fully connected networks with additional constraints (some weights forced to equal zero, and weight sharing). Understand how this significantly reduces the number of parameters in the network.

## Transfer Learning and Feature Learning

- What is the goal in transfer learning? What is "one-shot learning"? How would you use a trained neural network for one task to help with learning for another related task?
- Definition of an autoencoder and autoencoder loss.

- What is the "bottleneck" in an autoencoder and why is it necessary?
- What are some possible applications of autoencoders? Feature extraction, data denoising, missing data imputation, data compression, generating realistic synthetic data.

## Principal Component Analysis

- What is the linear autoencoder behind PCA?
- What is the connection between the autoencoder loss for PCA and low-rank approximation?
- What is the definition of the SVD? Understand what each of the three component matrices are, what it means for $U$ and $V$ to be orthogonal, etc.
- How do you find optimal weights for a linear autoencoder using the SVD?
- What are the "principal components" and what are the "loading vectors"?
- Be able to recognize when a dataset might naturally have a good low-rank approximation.
- Understand why the loading vectors approximately preserve distances and inner products of the original data.

## Semantic Embeddings

- What is Latent Semantic Analysis? How does it obtain "word" and "document" vectors? Vectors for similar words should have high inner product (positive) and for dissimilar words should have low inner product (negative or near 0).
- What is the connection between LSA and factorizing the word co-occurrence matrices? How can LSA be generalized to other similarity measures?
- The left/right singular vectors of the word co-occurrence matrix are equal to the right singular vectors of the term-document matrix.
- Why does the word co-occurrence matrix used in LSA have a *symmetric* low-rank approximation? General similarity matrices will not.
- What are node embeddings? How is similarity defined via random walks in a graph?