Reinforcement Learning

CMSC 678
UMBC
Announcement 1: Exam 2

Friday May 18th, 1pm-3pm

You can bring any of your own hard-copy notes (or course material photocopies)

you don’t need to turn them in

<table>
<thead>
<tr>
<th>Allowed</th>
<th>Not Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Your notes</td>
<td>• Your computer/phone</td>
</tr>
<tr>
<td>• CIML, ESL, UML, ITILA photocopies/printouts</td>
<td>• Your friend’s notes</td>
</tr>
<tr>
<td>• Slide printouts</td>
<td>• Any full (bound) textbooks</td>
</tr>
<tr>
<td>• Cheat sheets that you’ve written (or typed) yourself</td>
<td>• Printouts of non-course materials (e.g., printouts from Coursera’s ML course)</td>
</tr>
<tr>
<td>• Notes you’ve transcribed from online sources (w/ citation)</td>
<td></td>
</tr>
</tbody>
</table>

(don’t assume you’ll have time to continuously flip through your notes)
Announcement 2: Final Project

Due: Wednesday May 23, 11:59 AM

Turn in the report, code, (best) model(s), and any new data
Recap from last time...
Ensembles

“Wisdom of the crowd:" groups of people can often make better decisions than individuals

Reuse previous classifiers

Boosting — a method that takes classifiers that are only slightly better than chance and learns an arbitrarily good classifier
Voting Multiple Classifiers

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging
Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Option 2: Bootstrap aggregation (bagging) resampling

Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from $D$

Train classifiers on each dataset and average their predictions

Q: What can go wrong with option 1?

A: Small sample $\rightarrow$ poor performance

Given a dataset $D$...

get new datasets $D$ by random sampling with replacement from $D$
Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size $k$ and pick the best among these.

Train decision trees of depth $d$.

Average results from multiple randomly trained trees.

**Q:** What’s the difference between bagging decision trees and random forests?

**A:** Bagging $\rightarrow$ highly correlated trees (reuse good features).
Boosting takes a poor learning algorithm (weak learner) and turns it into a good learning algorithm (strong learner).

Intuition behind AdaBoost: study for an exam by taking past exams

1. Take the exam
2. Pay less attention to questions you got right
3. Pay more attention to questions you got wrong
4. Study more, and go to step 1
There’s an entire book!

Reinforcement Learning

agent

environment

Robot image: openclipart.org

Reinforcement Learning

agent \rightarrow \text{take action} \rightarrow \text{environment}
Reinforcement Learning

agent

take action

get new state and/or reward

environment

Robot image: openclipart.org

Reinforcement Learning

agent

take action

get new state and/or reward

environment

Robot image: openclipart.org
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, p, \gamma)\)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \( (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma) \)

- set of possible states
- set of possible actions
- get new state and/or reward
- environment
- take action
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, p, \gamma)\)

- set of possible states
- set of possible actions
- reward of (state, action) pairs

Robot image: openclipart.org

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, p, \gamma)\)

- set of possible states
- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \( (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma) \)

- \( \mathcal{S} \): set of possible states
- \( \mathcal{A} \): set of possible actions
- \( \mathcal{R} \): reward of (state, action) pairs
- \( p \): state-action transition distribution
- \( \gamma \): discount factor

Environment

Agent

Take action

Get new state and/or reward
Robot in a room

Goal: what’s the strategy to achieve the maximum reward?

actions: UP, DOWN, LEFT, RIGHT

UP
80% move UP
10% move LEFT
10% move RIGHT

reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

Slide courtesy Peter Bodík
Robot in a room

<table>
<thead>
<tr>
<th>START</th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **actions**: UP, DOWN, LEFT, RIGHT

- **UP**
  - 80% move UP
  - 10% move LEFT
  - 10% move RIGHT

- Reward: +1 at [4,3], -1 at [4,2]
- Reward: -0.04 for each step

**states**: current location  
**actions**: where to go next  
**rewards**

**what is the solution?** map each state to an action
**Robot in a room**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STRT</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>+1</strong></td>
</tr>
<tr>
<td></td>
<td><strong>-1</strong></td>
<td></td>
</tr>
</tbody>
</table>

**actions:** UP, DOWN, LEFT, RIGHT

**UP**
- 80% move UP
- 10% move LEFT
- 10% move RIGHT

Reward: +1 at [4,3], -1 at [4,2]
Reward: -0.04 for each step

**states:** current location
**actions:** where to go next
**rewards**

what is the solution? map each state to an action

Slide courtesy Peter Bodík
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- state-action transition distribution
- discount factor

Start in initial state $s_0$
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)\)

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- state-action transition distribution
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to ...
choose action \(a_t\)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: 

\((S, A, R, \pi, \gamma)\)

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- state-action transition distribution
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to ...:
choose action \(a_t\)
“move” to next state \(s_t \sim \pi(\cdot|s_{t-1}, a_t)\)
Markov Decision Process: Formulating Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to \(\ldots\):
  - choose action \(a_t\)
  - “move” to next state \(s_t \sim \pi(\cdot| s_{t-1}, a_t)\)
  - get reward \(r_t = R(s_t, a_t)\)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to ...:

- choose action \(a_t\)
- “move” to next state \(s_t \sim \pi(\cdot | s_{t-1}, a_t)\)
- get reward \(r_t = R(s_t, a_t)\)

objective:

\[
\max_{\pi} \sum_t \gamma^t r_t
\]
Markov Decision Process: Formulating Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- state-action transition distribution
- discount factor

Start in initial state \(s_0\)

for \(t = 1\) to ...:

choose action \(a_t\)

“move” to next state \(s_t \sim \pi(\cdot | s_{t-1}, a_t)\)

get reward \(r_t = R(s_t, a_t)\)

objective:

\[
\max_{\pi} \sum_t \gamma^t r_t
\]

“solution” \(\pi^* = \arg\max_{\pi} \mathbb{E} \left[ \sum_t \gamma^t r_t ; \pi \right] \)
Step Rewards Change the Optimal Solution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>←</td>
<td>↑</td>
<td>↑</td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td></td>
<td>↑</td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td>←</td>
<td>←</td>
<td>←</td>
</tr>
</tbody>
</table>

-0.04

Examples courtesy Peter Bodík
Step Rewards Change the Optimal Solution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td></td>
<td></td>
<td>-0.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>→</td>
<td></td>
<td></td>
<td>-2</td>
</tr>
</tbody>
</table>
### Step Rewards Change the Optimal Solution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-0.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-2</td>
</tr>
</tbody>
</table>

Examples courtesy Peter Bodík
Step Rewards Change the Optimal Solution

Examples courtesy Peter Bodík
State Representation

Task: pole-balancing

state representation?

move car left/right to keep the pole balanced
State Representation

Task: pole-balancing

state representation
  position and velocity of car
  angle and angular velocity of pole

what about *Markov property*?
State Representation

Task: pole-balancing

state representation
  position and velocity of car
  angle and angular velocity of pole

what about Markov property?
  would need more info
  noise in sensors, temperature, bending of pole

move car left/right to keep the pole balanced
Designing Rewards

robot in a maze
   episodic task, not discounted, +1 when out, 0 for each step

chess
   GOOD: +1 for winning, -1 losing
   BAD: +0.25 for taking opponent’s pieces
      high reward even when lose
Designing Rewards

robot in a maze
  episodic task, not discounted, +1 when out, 0 for each step

chess
  GOOD: +1 for winning, -1 losing
  BAD: +0.25 for taking opponent’s pieces
    high reward even when lose

rewards
  rewards indicate what we want to accomplish
  NOT how we want to accomplish it
Designing Rewards

robot in a maze
   episodic task, not discounted, +1 when out, 0 for each step

chess
   GOOD: +1 for winning, -1 losing
   BAD: +0.25 for taking opponent’s pieces
       high reward even when lose

rewards
   rewards indicate what we want to accomplish
   NOT how we want to accomplish it

shaping
   positive reward often very “far away”
   rewards for achieving subgoals (domain knowledge)
   also: adjust initial policy or initial value function
Value functions

state value function: $V_\pi(s)$

expected return when starting in $s$ and following $\pi$

$$V_\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, \pi \right]$$
Value functions

state value function: $V_\pi(s)$
expected return when starting in $s$
and following $\pi$

$$V_\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, \pi \right]$$

state-action value function: $Q_\pi(s, a)$
expected return when starting in $s$,
performing $a$, and following $\pi$

$$Q_\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, a_0 = a, \pi \right]$$
Value functions

state value function: $V_\pi(s)$
- expected return when starting in $s$ and following $\pi$

$$V_\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, \pi \right]$$

state-action value function: $Q_\pi(s, a)$
- expected return when starting in $s$, performing $a$, and following $\pi$

$$Q_\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi \right]$$

useful for finding the optimal policy
- can estimate from experience
- pick the best action using $Q_\pi(s, a)$
Value functions

state value function: $V_\pi(s)$

- expected return when starting in $s$ and following $\pi$

$$V_\pi(s) = \mathbb{E}\left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, \pi \right]$$

state-action value function: $Q_\pi(s, a)$

- expected return when starting in $s$, performing $a$, and following $\pi$

$$Q_\pi(s, a) = \mathbb{E}\left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, a_0 = a, \pi \right]$$

useful for finding the optimal policy

can estimate from experience

pick the best action using $Q_\pi(s, a)$
Optimal value functions

\[ Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, a_0 = a, \pi \right] \]
Optimal value functions

**definition**

\[ Q^*(s,a) = \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, a_0 = a, \pi \right] \]

**satisfies**

\[ Q^*(s,a) = \mathbb{E}_{s',a'} \left[ r + \gamma \max_{a'} Q^*(s',a') ; s_0 = s, a_0 = a, \pi \right] \]

*Bellman equation*
Optimal value functions

\[ Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t ; s_0 = s, a_0 = a, \pi \right] \]

definition

\[ Q^*(s, a) = \mathbb{E}_{s', \alpha'} \left[ r + \gamma \max_{a'} Q^*(s', a') ; s_0 = s, a_0 = a, \pi \right] \]

satisfies

Bellman equation

idea: if you know the best action to take, then the best strategy maximizes the overall expected reward
Overview: Learning Strategies

Dynamic Programming

Q-learning

Monte Carlo approaches
Dynamic programming

use value functions to structure the search for good policies
Dynamic programming

use value functions to structure the search for good policies

policy evaluation: compute $V^\pi$ from $\pi$

policy improvement: improve $\pi$ based on $V^\pi$
Dynamic programming

use value functions to structure the search for good policies

- policy evaluation: compute $V^\pi$ from $\pi$
- policy improvement: improve $\pi$ based on $V^\pi$

start with an arbitrary policy
repeat evaluation/improvement until convergence
Policy evaluation/improvement

policy evaluation: $\pi \rightarrow V_\pi$

Bellman equations define a system of equations could solve, but will use iterative version

$$V_{k+1}(s) = \sum_a \pi(s, a) \sum_{k'} P_{ss'}^a \left[ r_{ss'}^a + \gamma V_k(s') \right]$$

start with an arbitrary value function $V_0$, iterate until $V_k$ converges

policy improvement: $V_\pi \rightarrow \pi'$

$$\pi'(s) = \arg \max_a Q^\pi(s, a)$$

$$= \arg \max_a \sum_{s'} P_{ss'}^a \left[ r_{ss'}^a + \gamma V^\pi(s') \right]$$

$\pi'$ either strictly better than $\pi$, or $\pi'$ is optimal (if $\pi = \pi'$)
Q-learning

previous algorithms: on-policy algorithms
start with a random policy, iteratively improve
converge to optimal

Q-learning: off-policy
use any policy to estimate Q

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \]

Q directly approximates Q* (Bellman optimality equation)
independent of the policy being followed
only requirement: keep updating each (s,a) pair
Q-learning

previous algorithms: on-policy algorithms
  start with a random policy, iteratively improve
  converge to optimal

Q-learning: off-policy
  use any policy to estimate $Q$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Q directly approximates $Q^*$ (Bellman optimality equation)
  independent of the policy being followed
  only requirement: keep updating each (s,a) pair
Deep/Neural Q-learning

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

neural network       desired optimal solution
Deep/Neural Q-learning

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

- neural network
- desired optimal solution

Q: What’s a good loss function?
Deep/Neural Q-learning

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

neural network  
desired optimal solution

Q: What’s a good loss function?  
A: Squared expectation loss
Monte Carlo policy evaluation

don’t need full knowledge of environment (just (simulated) experience)

want to estimate $V^\pi(s)$
Monte Carlo policy evaluation

Don’t need full knowledge of environment (just (simulated) experience)

Want to estimate $V^\pi(s)$

Expected return starting from $s$ and following $\pi$

Estimate as average of observed returns in state $s$

$V^\pi(s) \approx \frac{(2 + 1 - 5 + 4)}{4} = 0.5$
Maintaining exploration

key ingredient of RL

deterministic/greedy policy won’t explore all actions
  don’t know anything about the environment at the beginning
  need to try all actions to find the optimal one

maintain exploration
  use soft policies instead: $\pi(s,a)>0$ (for all $s,a$)

$\epsilon$-greedy policy
  with probability $1-\epsilon$ perform the optimal/greedy action
  with probability $\epsilon$ perform a random action

will keep exploring the environment
slowly move it towards greedy policy: $\epsilon \rightarrow 0$
RL Slides Credit

https://people.eecs.berkeley.edu/~jordan/courses/294-fall09/lectures/reinforcement/slides.pptx
Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)
Machine Learning Framework: Learning

- instance 1
- instance 2
- instance 3
- instance 4

Machine Learning Predictor

Evaluator

Gold/correct labels

score

give feedback to the predictor

instances are typically examined independently

Extra-knowledge
Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Learn different ways that success and progress can be measured in ML
Machine Learning Framework: Learning

Instances are typically examined independently.

Gold/correct labels give feedback to the predictor.

Sometimes: the function we can learn $\neq$ the function we want to learn.
Optimize Empirical Risk of Surrogate Loss

\[ \argmin_h \sum_{i=1}^{N} \ell(y_i, h_\theta(x_i)) \quad \text{empirical risk minimization} \]

\[
\nabla_\theta F = \sum_i \frac{\partial \ell(y_i, \hat{y} = h_\theta(x_i))}{\partial \hat{y}} \nabla_\theta h_\theta(x_i) \quad \text{approximate loss in a computable way}
\]
Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)
Learn different ways that success and progress can be measured in ML
Relate to statistics, AI [671], and specialized areas (e.g., NLP [673] and CV [691])
Implement ML programs
Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Learn different ways that success and progress can be measured in ML

Relate to statistics, AI [671], and specialized areas (e.g., NLP [673] and CV [691])

Implement ML programs

Read and analyze research papers

Practice your (written) communication skills
Remember from the first day:
A Terminology Buffet

Classification

Regression

Clustering

the **task**: what kind of problem are you solving?

**Fully-supervised**

**Semi-supervised**

**Un-supervised**

the **data**: amount of human input/number of labeled examples

Probabilistic  Neural

Generative  Memory-based

Conditional  Exemplar

Spectral  ...

the **approach**: how any data are being used
A Terminology Buffet

Classification

Regression

Clustering

Fully-supervised

Semi-supervised

Un-supervised

what we covered through exam 1...

the task: what kind of problem are you solving?

the data: amount of human input/number of labeled examples

the approach: how any data are being used
Course Overview (Part 1)

Basics of Probability

Requirements to be a distribution ("proportional to", ∝)
Definitions of conditional probability, joint probability, and independence
Bayes rule, (probability) chain rule
Expectation (of a random variable & function)

Empirical Risk Minimization

Gradient Descent
Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

Data set splits: training vs. dev vs. test
Classification: Posterior decoding/MAP classifier
Classification evaluations: accuracy, precision, recall, and F scores
Regression (vs. classification)
Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
Clustering: high-level goal/task, K-means as an example
Tradeoffs among clustering evaluations

Linear Models

Basic form of a linear model (classification or regression)
Perceptron (simple vs. other variants, like averaged or voted)
When you should use perceptron (what are its assumptions?)
Perceptron as SGD

Maximum Entropy Models

Meanings of feature functions and weights
How to learn the weights: gradient descent
Meaning of the maxent gradient

Neural Networks

Relation to linear models and maxent
Types (feedforward, CNN, RNN)
Learning representations (e.g., "feature maps")
What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
How to learn: gradient descent, backprop
Common activation functions
Neural network regularization

Dimensionality Reduction

What is the basic task & goal in dimensionality reduction?
Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

Feature expansion and kernels
Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
Non-separability & slack
Sub-gradients
A Terminology Buffet

**Classification**

- Fully-supervised
- Semi-supervised
- Un-supervised

**Regression**

- Probabilistic
- Generative
- Neural

**Clustering**

- Conditional
- Memory-based
- Exemplar

---

**the task**: what kind of problem are you solving?

**the data**: amount of human input/number of labeled examples

**the approach**: how any data are being used

---

*what we covered through exam 1...*  
*what we covered after exam 1...*
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution (“proportional to”, \(\propto\))
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., “feature maps”)
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

leveraging large, unannotated data
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution (“proportional to”, \(\propto\))
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/ MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Course Overview (Part 2)
- leveraging large, unannotated data
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution ("proportional to", \(\propto\))
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Dataset splits: training vs. dev vs test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution ("proportional to", $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)

Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models
- Meaning: what’s a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn’t)
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution ("proportional to", $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)

Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models
- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn’t)

Distributions as Graphs
- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)

Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models
- Meaning: what’s a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn’t)

Distributions as Graphs
- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference
- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution (“proportional to”, \(\propto\))
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)

Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models
- Meaning: what’s a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn’t)

Distributions as Graphs
- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference
- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods

Structured Prediction
- General formulation
- Porting existing approaches: non-separability, slack, and sub-gradients
- Learning vs. inference
- General inference technique: ILP
Course Overview (Part 2)

Basics of Probability
- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization
- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)
- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models
- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)

Perceptron as SGD

Maximum Entropy Models
- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks
- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction
- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs
- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models
- Meaning: what’s a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn’t)

Distributions as Graphs
- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference
- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods

Structured Prediction
- General formulation
- Porting existing approaches: non-separability, slack, and sub-gradients
- Learning vs. inference
- General inference technique: ILP

Ensembles & RL
- Combining approaches