

Machine Learning

Advanced Topic (Part 2) and Review

DSC 240

March 11, 2025

Instructor: Prof. Yu-Xiang Wang

Announcement

- In-Class Final Quiz on Thursday
 - Prepare by solving problems in this lecture
 - Two cheat sheets allowed.
 - The focus is on the second half of the course.
- Course Evaluation form
 - 2% participation bonus for everyone if we get above 75%
 - 36.4% now. Keep it up!

Last time

- Intuition behind PCA for Dimension-reduction
 - If finding the best low-dimensional subspace to approximate the (centered covariance matrix)
 - The optimization problem can be solved using SVD
- Advanced topic: Generalization theory
 - Rademacher complexity: the ability for the hypothesis to fit random labels.

$$\widehat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[\max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \sigma_i h(x_i) \right]$$

Advance topics: Which topic should I cover?

- Generalization?
 - Covered in Lecture 9 and then a bit more in Lecture 17 (last week)
 - Rademacher complexity
- Deep Learning?
- Online Learning?
 - Very briefly explained (stock portfolio example)
 - Closest topic that we covered: Perceptron algorithm.
- Reinforcement Learning?

Recap: Intuition behind Rademacher complexity in binary classification

$$\widehat{\text{Rad}}(\mathcal{H}) = \mathbb{E} \left[\max_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \sigma_i h(x_i) \right]$$

Recap: Small Rademacher Complexity implies generalization

Theorem: With probability at least $1 - \delta$, for $\forall h \in \mathcal{H}$,

$$\left| \hat{R}(h) - R(h) \right| \leq \text{Rad}(\mathcal{H}) + \sqrt{\frac{\log(2/\delta)}{n}}$$

Useful lemmas

$$\widehat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2 \log |\mathcal{H}|}{n}}$$

$$\widehat{\text{Rad}}(\mathcal{H}) \leq \sqrt{\frac{2 \text{VCdim}(\mathcal{H}) \log n}{n}}$$

Shattering and VC-Dimension

- **Shattering:** We say \mathcal{H} **shatters** a set of points $S \subset \mathcal{X}$ if it can **realizes any labeling of the points** in S .
- **VC-dimension:** $\text{VCdim}(\mathcal{H})$ is the largest number of points in $S \subset \mathcal{X}$ such that \mathcal{H} **shatters** S .
- **Example 1:** $\mathcal{X} = \mathbb{R}$ and $h \in \mathcal{H}$ satisfies
 - $h(x) = 1$ if $x > \tau$ and $h(x) = -1$ otherwise.
 - VC-dimension = ?

More examples of VC-dimensions from the last lecture (not covered)

Summary: Generalization Theory

- The generalization error depends on the “Complexity” of the hypothesis class.
- The more the hypothesis class can overfit to random labels, the more it is prone to overfitting
 - Rademacher Complexity
 - VC Theory

Today

- Advanced Topics (Part 2)
 - Reinforcement Learning
 - Some remarks on Deep learning
- Final Review

“Reinforcement Learning” pioneers just won Turing Award.



Association for Computing Machin... @TheOfficialA... · Mar 5

Meet the recipients of the 2024 ACM A.M. **Turing Award**, Andrew G. Barto and Richard S. Sutton! They are recognized for developing the conceptual and algorithmic foundations of reinforcement learning. Please join us in congratulating the two recipients! <https://bit.ly/4hpdsbD>

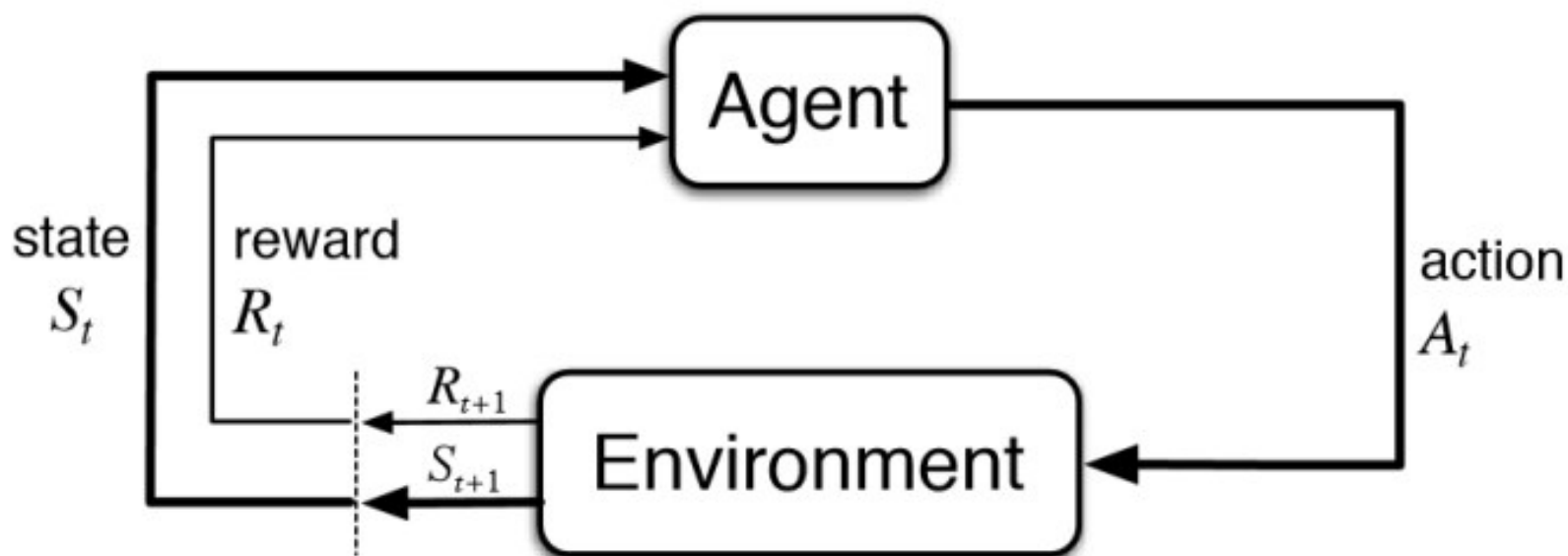


Applications in AI:

AlphaGo
AlphaGoZero
AlphaZero
MuZero

Aligning LLMs to human values

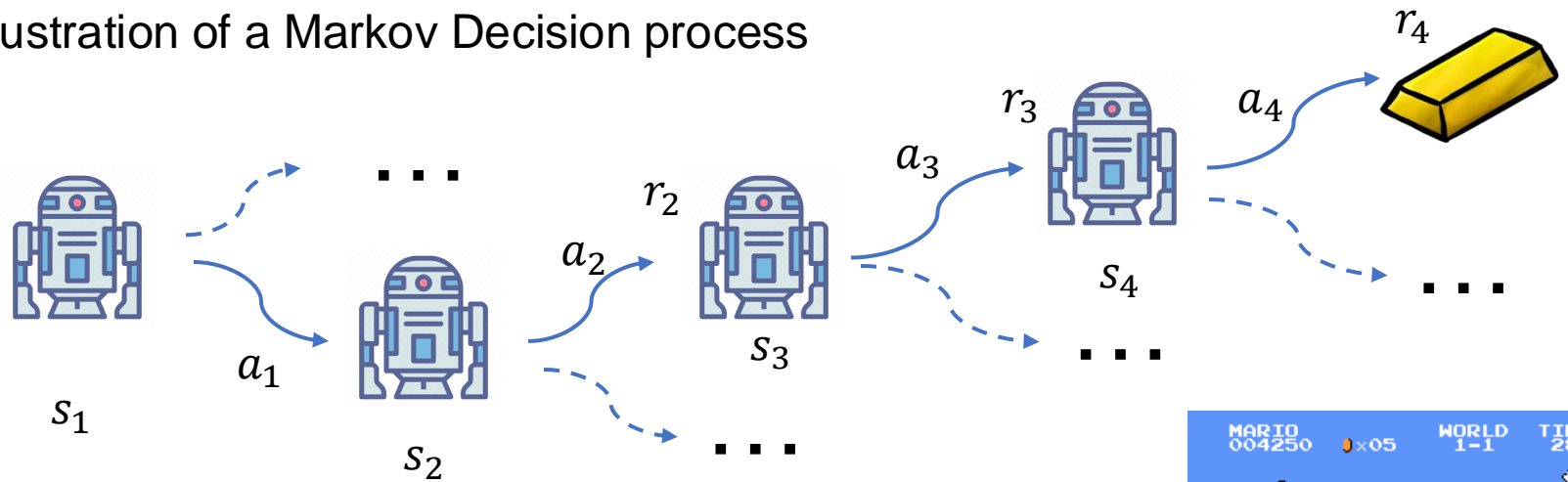
An RL agent learns **interactively** through the **feedbacks** of an environment.



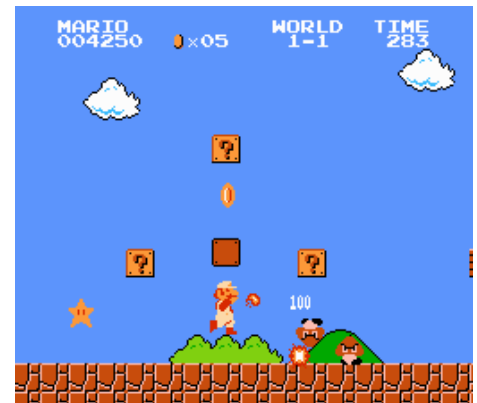
- Learning how the world works (dynamics) and how to maximize the long-term reward (control) at the same time.

RL agent aims at learning a **decision policy** rather than a **prediction rule** (as in supervised learning).

Illustration of a Markov Decision process



Your decision influences where you go next.



Reinforcement learning problem setup

- State, Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O}$$

- Policy:

- When the state is observable: $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- Or when the state is not observable

$$\pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A}$$

- Learn the best policy that maximizes the expected reward

- Finite horizon (episodic) RL:

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[\sum_{t=1}^T R_t \right]$$

T: horizon

- Infinite horizon RL:

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \right]$$

γ : discount factor

RL for robot control



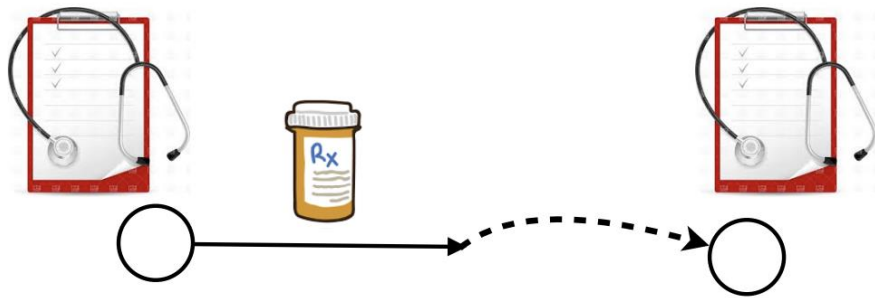
- States: The physical world, e.g., location/speed/acceleration and so on.
- Observations: camera images, joint angles
- Actions: joint torques
- Rewards: stay balanced, navigate to target locations, serve and protect humans, etc.

RL for Inventory Management

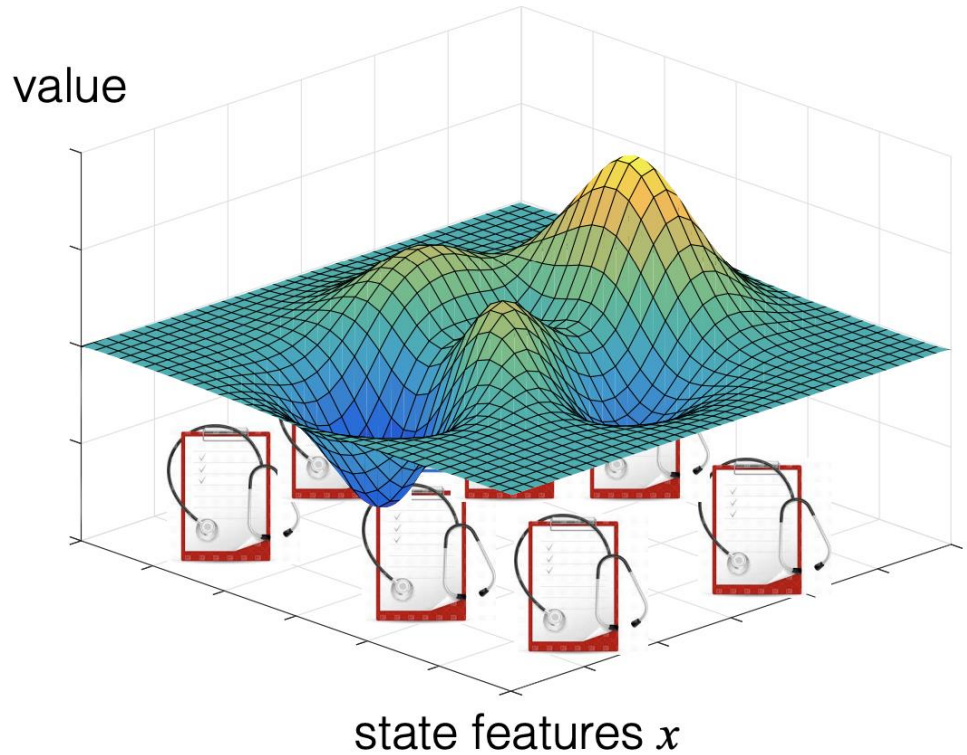


- State: Inventory level, customer demand, competitor's inventory
- Observations: current inventory levels and sales history
- Actions: amount of each item to purchase
- Rewards: profit

RL for Adaptive medical treatment



- State: diagnosis
- Action: treatment
- Reward: progress in recovery



(example / illustration due to Nan Jiang)

Example: Supervised learning vs RL in movie recommendation

- Bob is described by a feature vector
 - $s = [\text{Previous movies watched} / \text{Rating} / \text{Written reviews}]$
- Supervised learning predicts how likely Bob will click on “aliens vs predators”
- Reinforcement learning aims at controlling Bob
 - So in the future, Bob will develop a taste for “aliens vs predators” (e.g., from having watched “aliens” and “predators” both).

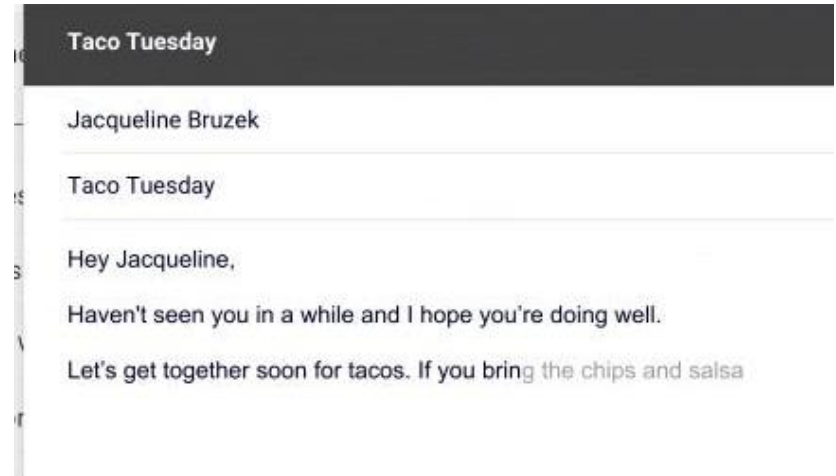
A broader view: Let's consider a few other machine learning tasks

- Hospitals need to decide **who to test** based on symptoms and other patient attributes



- Train a classifier on historic records to predict the test outcome.
- The accuracy is high on a holdout set!

- Large tech wants to improve user experience on their popular email service



- Train a **large language model** with user data to **complete sentences**
- It seems to work great!

What could go wrong?

Every machine learning problem is secretly a control (or RL) problem

- If I test patients using the new rule, the distribution of patients receiving the test will be different!
- Should I still trust my classifier?

- If I deploy the new “Guess what you will write” prompt, what users will enter may change!
- Is the model fulfilling its own prophecy?

The ultimate goal is NOT prediction, but to:
minimize disease transmission / maximize user
experience!

Reinforcement learning is very challenging

- The agent needs to:
 - Learn the state-transitions ----- How the world works
 - Learning the costs / rewards ----- Cost of actions
 - Learning how to search ----- Come up with a good decision policy
- All at the same time

To appropriately teach RL basics I need at least 3 lectures. It's easily a whole course to develop more thorough understanding.

- Markov Decision Processes:
 - Dynamics are given no need to learn. planning only.
- RL algorithms
 - Model-based RL vs Model-free RL
 - Temporal difference learning
 - Function approximation
- Exploration
 - Bandits: Explore-Exploit in simple settings
 - RL: Explore-Exploit in Learning MDPs

How do I cover these in half a lecture? I will focus on key ideas and leverage what you've learned throughout the ML course.

- Value function and Bellman equation
- Mean Square Bellman Error and Q-learning
- The need for Function approximation
- A demo of RL for PACMAN

Reward function and Value functions

- Immediate reward function $r(s,a)$

- **expected immediate** reward

$$r(s, a) = \mathbb{E}[R_1 | S_1 = s, A_1 = a]$$

$$r^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)}[R_1 | S_1 = s]$$

- state value function: $V^\pi(s)$

- **expected long-term** return when starting in s and following π

$$V^\pi(s) = \mathbb{E}_\pi[R_1 + \gamma R_2 + \dots + \gamma^{t-1} R_t + \dots | S_1 = s]$$

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

- **expected long-term** return when starting in s , performing a , and following π

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_1 + \gamma R_2 + \dots + \gamma^{t-1} R_t + \dots | S_1 = s, A_1 = a]$$

Optimal value function and the MDP planning problem

$$V^*(s) := \sup_{\pi \in \Pi} V^\pi(s)$$

$$Q^*(s, a) := \sup_{\pi \in \Pi} Q^\pi(s, a).$$

Goal of MDP planning:

Find π^* such that $V^{\pi^*}(s) = V^*(s) \quad \forall s$

Approximate solution:

π is ϵ -optimal if $V^\pi \geq V^*(s) - \epsilon \mathbf{1}$

Bellman optimality equations characterizes the optimal policy

$$V^*(s) = \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V^*(s')]$$

- system of n non-linear equations
 - solve for $V^*(s)$
 - easy to extract the optimal policy
-
- having $Q^*(s, a)$ makes it even simpler

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

- Bellman Optimality Equation with the Q-functions

Mean Square Bellman Error Minimization

$$Q(s, a) = \mathbb{E}[r(s, a) + \gamma \max_{a'} Q(s', a')]$$

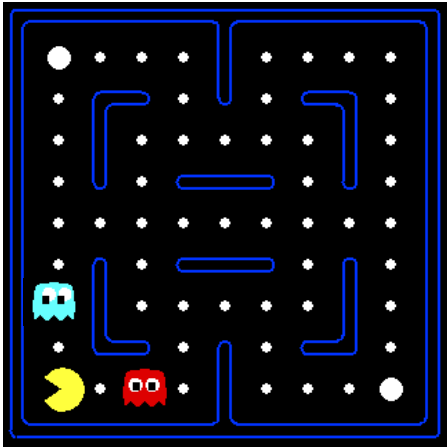
- In every round of the MDP, we start at **state s**, take **action a**, receives **reward r** and then arrive at **states s'**
 - Receive one data point quadruple: (s, a, r, s')
- Take derivative of the square Bellman error.

How do we represent the value functions?

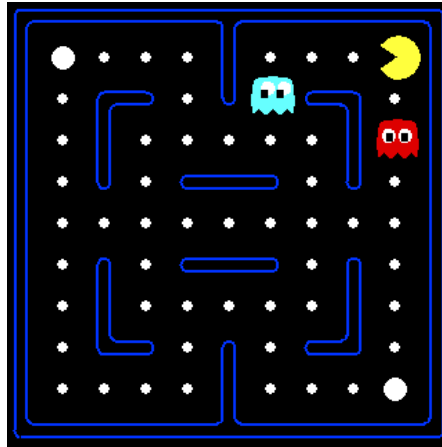
1. A look-up table of state and actions
2. A linear function of of “features” of state and action pairs
3. A neural network of that takes the state and action (features) as input.

In practice, the “Look-up table” idea is terrible.
For example, in the PACMAN environment

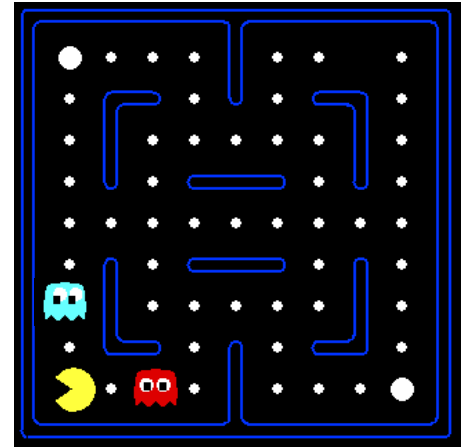
Let’s say we discover
through experience
that this state is bad:



In naïve q-learning,
we know nothing
about this state:

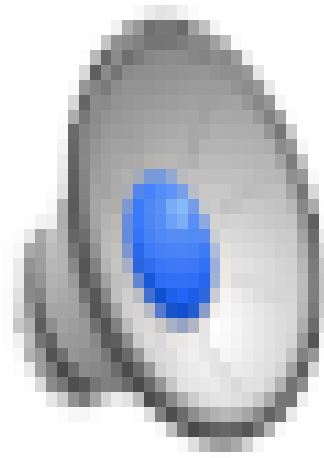


Or even this one!

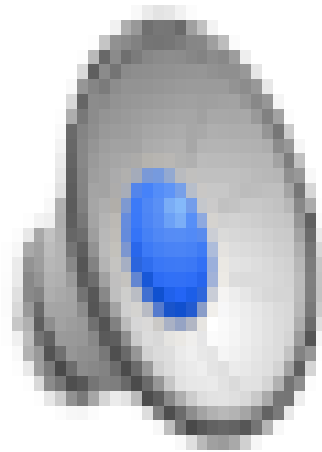


(From Dan Klein and Pieter Abbeel)

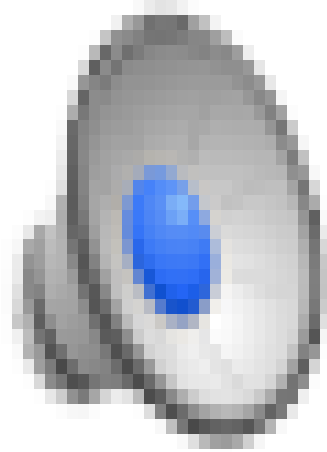
Video of Demo Q-Learning Pacman – Tiny – Watch All



Video of Demo Q-Learning Pacman – Tiny – Silent Train

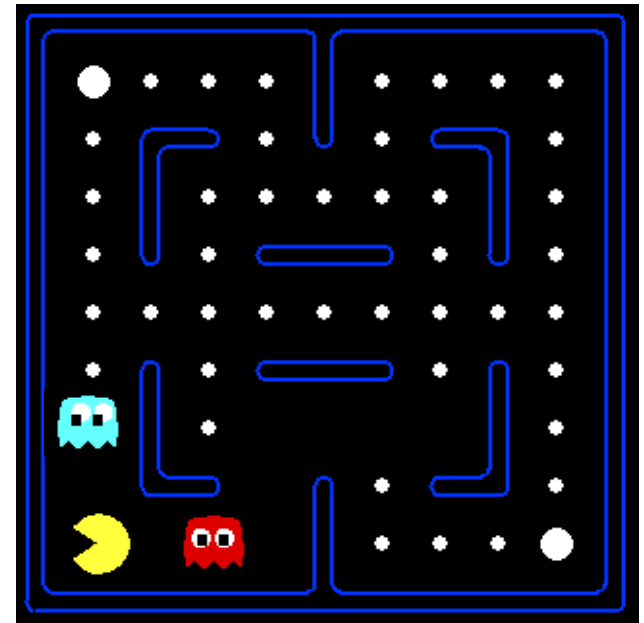


Video of Demo Q-Learning Pacman – Tricky –
Watch All



Idea: Function approximation. Let's come up with informative features.

- Solution: describe a state using a vector of features (properties)
 - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
 - Example features:
 - Distance to closest ghost
 - Distance to closest dot
 - Number of ghosts
 - $1 / (\text{dist to dot})^2$
 - Is Pacman in a tunnel? (0/1)
 - etc.
 - Is it the exact state on this slide?
 - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Updating a linear value function

- Original Q learning rule tries to reduce prediction error at s, a :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

- Instead, we update the weights to try to reduce the error at s, a :

$$\begin{aligned} w_i &\leftarrow w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \partial Q_w(s,a) / \partial w_i \\ &= w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] f_i(s,a) \end{aligned}$$

Updating a linear value function

- Original Q learning rule tries to reduce prediction error at s, a :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

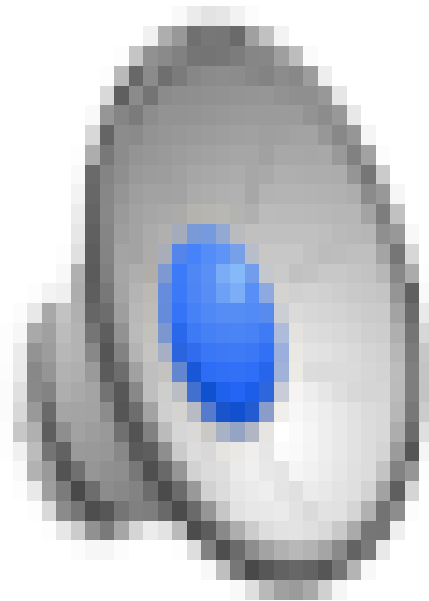
- Instead, we update the weights to try to reduce the error at s, a :

$$\begin{aligned} w_i &\leftarrow w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \partial Q_w(s,a) / \partial w_i \\ &= w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] f_i(s,a) \end{aligned}$$

- Qualitative justification:

- Pleasant surprise: increase weights on positive features, decrease on negative ones
- Unpleasant surprise: decrease weights on positive features, increase on negative ones

PACMAN Q-Learning (Linear function approx.)



Ideas of deep reinforcement learning

- Use neural networks for approximating the value function
- Use neural networks to describe policies.
- Many other tricks.

Watch the talk by David Silver on AlphaGo and AlphaZero:
<https://youtu.be/x5Q79XCxMVc?si=OkLueb2UI8LN1HoS>

Today

- Advanced Topics (Part 2)
 - Reinforcement Learning
 - Some remarks on Deep learning
- Final Review

On Deep Learning

- We have covered a fair bit of it in Lecture 15 and HW4 Q1.
- Some useful facts about deep learning to know:
 1. It is a drop-in replacement for the linear “score function” with a nonlinear one.
 2. It is a feature expansion technique but with learned features.
 3. It comes You should think of it as a “Programming Language” for constructing hypothesis classes that are suitable for your problem.

You can use neural network for all kinds of ML problems that we learned: classification, regression, clustering, dimension reduction etc..

- Neural networks provide a **learnable function approximation**
- Different kinds of NNs architecture (like LEGO blocks) are designed to address different challenges in different kind of problems:
 - Feedforward neural network
 - Recurrent neural network
 - Boltzmann machine
 - Convolutional neural network
 - Graph Neural Networks
 - Transformers
 - etc., etc.

Learning \approx Configuring the Learnable Function so it behaves as instructed.

- Speech Recognition

$$f(\text{audio waveform}) = \text{“你好”}$$

- Handwritten Recognition

$$f(\text{handwritten digit}) = \text{“2”}$$

- Weather forecast

$$f(\text{cloud and sun icon Thursday}) = \text{“cloud and rain icon Saturday”}$$

- Play video games

$$f(\text{gameplay screenshot}) = \text{“move left”}$$

Neural networks: Example: AlexNet (2012)

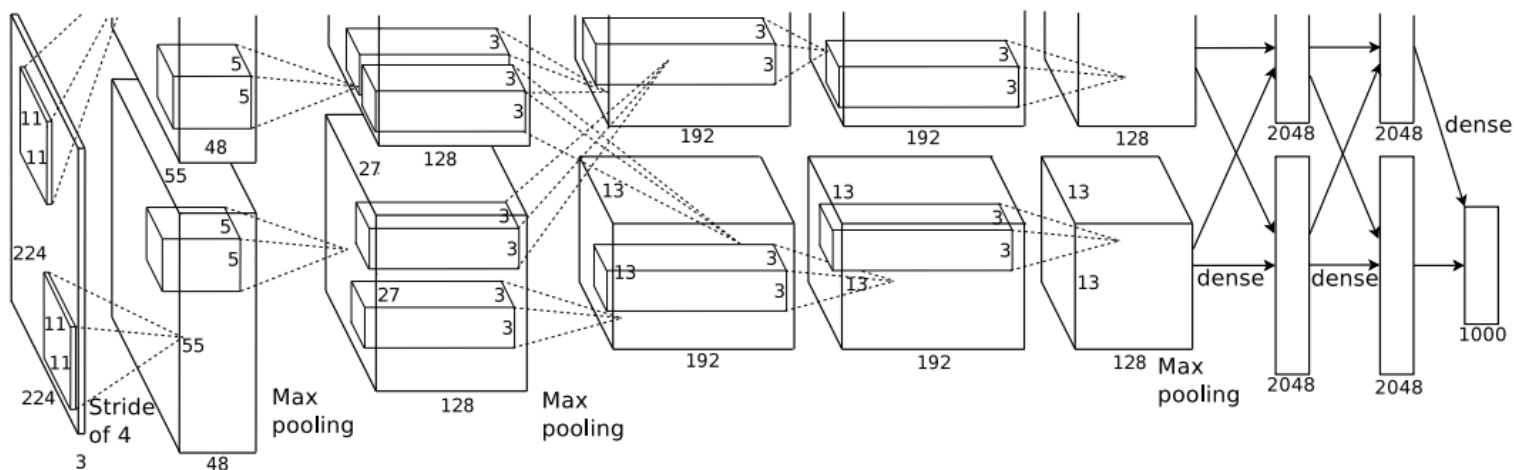


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

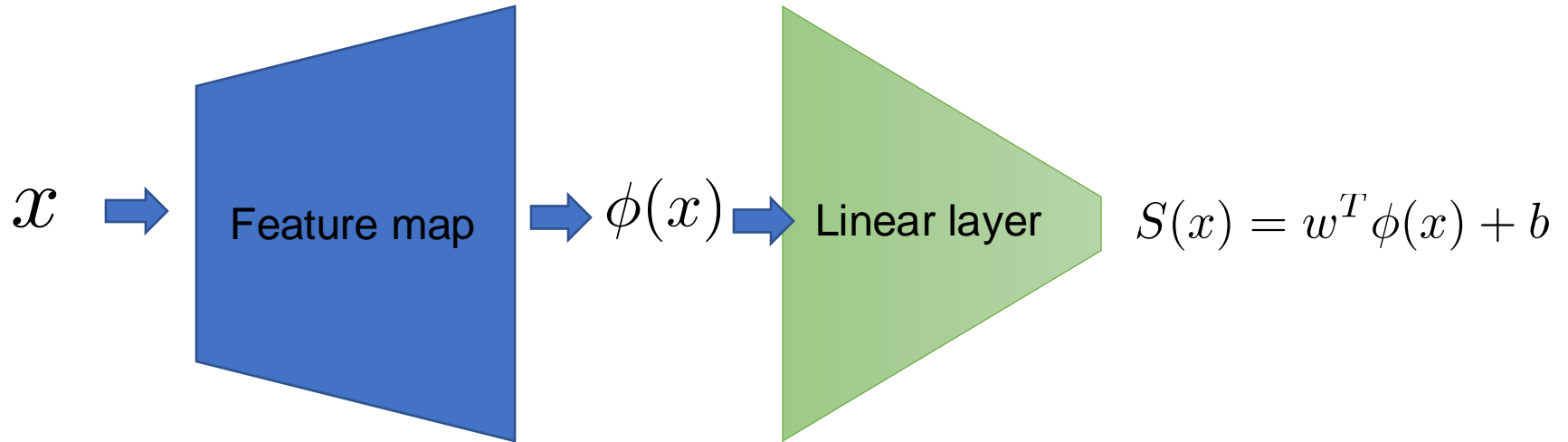
Imagenet classification with deep convolutional neural networks

[A Krizhevsky, I Sutskever...](#) - *Advances in neural ...*, 2012 - [proceedings.neurips.cc](#)

... a large, **deep convolutional neural network** to **classify** the 1.2 million high-resolution images in the **ImageNet** ... The **neural network**, which has 60 million parameters and 650,000 neurons, ...

☆ Save ↻ Cite Cited by 122248 Related articles All 111 versions Import into BibTeX ↻

Recap: From kernels to neural networks



Main reason neural networks took off --- computing + data + deep learning frameworks with autograd (from HW4)

Generally speaking, you need to make decisions about

- Which loss function to use
 - Regression, classification, clustering, dimension reduction, but also ranking, recommendation, and others...
- What type of neural network to use
 - Images
 - Text
 - Graphs (node and edges)
 - Time series
 - Decide on the hyperparameters: Depth, Width, number of hidden units, etc...
- How to train the neural network?
 - Initialization of weights: iid random? Recale or not?
 - Optimizer to use: SGD, ADAM, etc...
- How to collect, pre-process the data...

Modern Research in Machine Learning

- New model architectures
- Faster optimization
- Other dimensions of efficiency: Less for More
- Understand deep learning / AI
- How to handle distribution-shift. How to determine causal structures
- Robustness, Privacy, Explanability, fairness, security, etc...
- Applications to stateful, adversarial, strategic, environments.
- A lot of the above requires theory and algorithms. What concepts are learnable and what not? How many samples are needed? What is the optimal algorithm for each learning problem.
- And many more!

This course is a gentle introduction to machine learning... Where to go from here?

- **Various advanced ML courses from CSE and HDSI**

- CSE 252A/B computer vision. CSE 256 NLP, CSE 257 Discrete Optimization ...
- DSC 245 Causal inference. DSC 250 Advanced data mining. DSC 261 Responsible Data Science, etc.

- **DSC 291 Special topic courses**

- Convex Optimization (I may offer it for Fall 2025)
- Reinforcement Learning (Yian Ma taught it last quarter)
- Online Learning (Yoav Freund teaching it this quarter)
- Statistical Machine Learning
- Deep Learning (<https://zhiting.ucsd.edu/teaching/dsc291spring2024/>) offered for 2025 Spring.

- More advanced textbooks:

- [Probabilistic Machine Learning](#) by **Kevin Murphy**. (Two books: 2022 and 203)
- [ML Theory book](#) by **Tong Zhang**.
- [Patterns, Predictions, and Actions](#) by **Recht and Hardt (2023)**.
- [Learning Theory from First Principles](#) by Francis Bach (2023)

That's the end of all materials in this course. Time for a review!

	Tues and Thurs	Lectures	Reading materials
1	7-Jan	Intro and course overview	
2	9-Jan	Spam Filtering [Annotated]	Lecture note
3	14-Jan	ML Basics [Annotated]	Lecture note
4	16-Jan	Linear Algebra Review [Annotated]	
5	21-Jan	How to train a linear classifier? Perceptron [Annotated]	Bishop 4.1
6	23-Jan	Surrogate Loss and First-Order Optimization [Annotated]	D2L 12.3.1, 12.3.2, 12.4.1
7	28-Jan	Linear Regression [Annotated]	D2L 3.1
8	30-Jan	Regularization [Annotated]	Bishop 3.1.4
9	4-Feb	Generalization theory + Midterm Review [Annotated]	
10	6-Feb	Midterm Quiz	
11	11-Feb	Max-Margin Linear Separator and Probability Review [Annotated]	Bishop 2.1-2.3
12	13-Feb	Statistics review and Max-Likelihood Estimation [Annotated]	Bishop 4.2.2, Bishop 4.3
13	18-Feb	Generative Models and Naive Bayes Classifier [Annotated]	Bishop 4.2.1-4.2.3
14	20-Feb	Decision Tree and Boosting [Annotated]	Bishop 14.2, 14.3, 14.4
15	25-Feb	Feature Expansion and Kernel Methods [Annotated]	Bishop 1.1, 6.1, 6.2
16	27-Feb	Neural Networks [Annotated]	Bishop 5.1, D2L 2.5, D2L 5.1
17	4-Mar	Unsupervised Learning (Clustering and GMM) [Annotated]	Bishop 9.1, 9.2, (optional 9.3)
18	6-Mar	Unsupervised Learning (Dimension Reduction) [Annotated]	Bishop 12.1, (optional 12.2)
19	11-Mar	Advanced Topic (Reinforcement Learning) and Final Review	
20	13-Mar	Final Quiz (9:30 am CENTR 105)	

Problem setup for machine learning problems

- Loss function

$$\ell(h, (x, y))$$

- Empirical Risk function

$$\hat{R}(h, \text{Data}) = \frac{1}{n} \sum_{i=1}^n \ell(h, (x_i, y_i))$$

- (Population) Risk function

$$R(h, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(h, (x_i, y_i))]$$

Recap: Risk Decomposition

$$\begin{aligned} & \mathbb{E}[R(\hat{h})] - R(h_{\text{Bayes}}) \\ \leq & \underbrace{\mathbb{E}[\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})]}_{\text{Optimization Error}} + \underbrace{R(h^*) - R(h_{\text{Bayes}})}_{\text{Approximation Error}} + \underbrace{\mathbb{E}[R(\hat{h}) - \hat{R}(\hat{h})]}_{\text{Generalization Error}} \end{aligned}$$

How close am I from minimizing the empirical risk?

How much worse the best “representable” classifier is from the best classifier out there.

How different the empirical risk of my classifier is from its population risk?

Make sure you understand what each kind of error means!

Machine learning can be viewed as a collection of techniques in minimizing the three types of errors

	Optimization error	Generalization Error	Approximation Error
Definition	$\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})$	$R(\hat{h}) - \hat{R}(\hat{h})$	$R(h^*) - R(h_{\text{Bayes}})$
Challenges	<ul style="list-style-type: none"> Finding ERM for some loss functions is NP-Hard. Efficiency isn't enough. Need to be scalable. 	<ul style="list-style-type: none"> We do not observe Risk! Don't have infinite data. Large generalization error \Leftrightarrow Overfitting 	<ul style="list-style-type: none"> Don't know data distribution. No knowledge of Bayes optimal classifier. Large approx. error \Leftrightarrow Underfitting!
What we have learned to address these challenges?	<p>"Just-relax" Surrogate loss, Gradient Descent, SGD</p> <p>Other more specialized solutions to optimization problems</p>	<p>Holdout, Cross-Validation</p> <p>Regularization</p> <p>Statistical learning theory</p>	<p>Better features</p> <p>More flexible decision boundaries</p> <p>Better probabilistic models</p> <p>Ensemble learning: Boosting, Bagging</p> <p>Feature expansion/ Kernels</p> <p>Neural Networks / Representation Learning</p>

Key questions in modelling and tradeoffs

- How to come up with suitable hypothesis class?
 - We want the approximation error to be small
 - We want it to be (statistically) efficiently learnable
- How to come up with suitable loss functions?
 - We want the loss function to reflect that performance metric of interest.
 - We want the loss function to be efficiently optimizable

Two philosophy for answering the two questions

- Deterministic / Discriminative view:
 - Loss function is a surrogate the performance metric that we care about.
 - e.g. logistic loss, hinge-loss, etc. upper bounds the 0-1 loss
 - Geometric view: Hypothesis class specifies the shapes of the decision boundary.
- Probabilistic / Generative view:
 - Loss function can be derived from Max-Likelihood Principle.
 - Hypothesis class is specified by a probabilistic model of the data-generation process.

Maximum Likelihood Estimation

- MLE defines an optimization problem to solve for estimating the parameters given data.
 - Find the parameter that maximizes the likelihood
 - Find the **distribution within a set of distributions** that maximizes the probability (likelihood) of observing the data

Problem 5 Maximum Likelihood Estimation

Let the data space be $[0, 1]$. And the data we observed be $0.1, 0.01, 0.5$. Let the set of probability distributions be $\mathcal{P} = \{P_1, P_2, P_3\}$, where P_1 is a uniform distribution on $[0, 1]$; P_2 is a uniform distribution on $[0, 0.2]$; P_3 is a distribution with probability density $p(x) = 2(1 - x)$ defined on $[0, 1]$.

Work out the likelihood function and the maximum likelihood estimate from \mathcal{P} .

Examples of supervised learning problems

	Binary classification	Multi-class classification	Regression
Feature space	\mathbb{R}^d	\mathbb{R}^d	\mathbb{R}^d
Label space	$\{-1, 1\}$	$\{1, 2, 3, \dots, K\}$	\mathbb{R}
Popular Performance metric	Classification error (0-1 loss) for test data	Classification error (0-1 loss) for test data	Mean Square Error (MSE) vs ground truth
Popular surrogate loss (for training)	Logistic loss	Multiclass logistic loss aka. Cross-Entropy loss	Square loss
Probabilistic interpretation	MLE for a Bernoulli (Linear) Model	MLE for a Multinomial model	MLE for a Gaussian Observation model

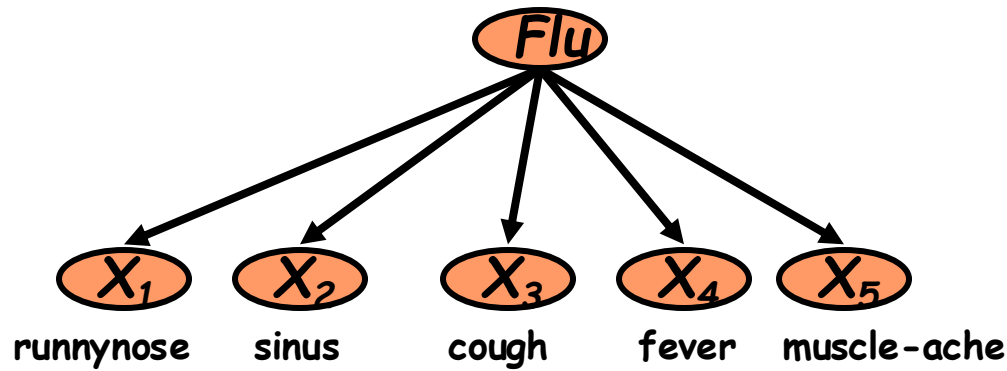
Classifiers that we have learned so far

- Linear classifier
- Decision Tree (Decision Stumps)
- Naïve Bayes classifier
- Voting classifiers \Leftarrow Bagging, Boosting
- Feature-expanded linear classifiers \Leftarrow Kernel methods
- Neural Networks \Leftarrow Learning representation

Important learning goals:

1. What are the parameters
2. Fix a parameter, how does the classifier make predictions
3. Sketching the decision boundary in a 2d plane

Naïve Bayes Model --- a simple example of a generative model



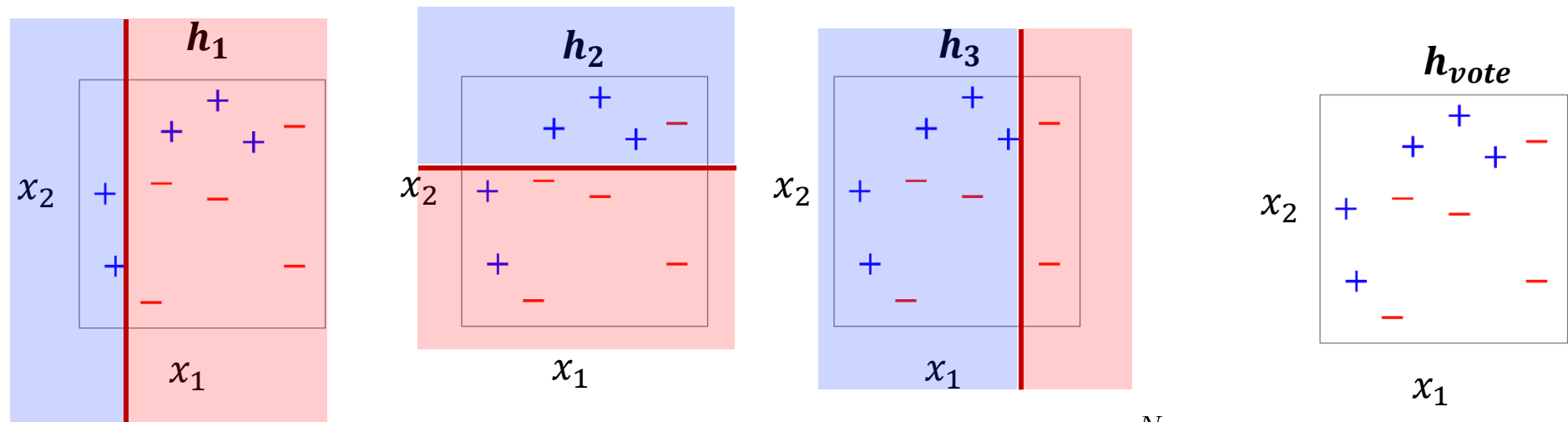
- **Conditional Independence Assumption:** features are independent of each other given the class:

$$P(X_1, \dots, X_5 | C) = P(X_1 | C) \cdot P(X_2 | C) \cdot \dots \cdot P(X_5 | C)$$

- **Interpretation:** given that you have Flu, the event that you experience each of the five symptoms are independent (is this a valid assumption?)
- **The task of classification:** Predict disease using symptoms

Decision Boundary of Gaussian Naïve Bayes (HW3)

Decision stumps and voting classifiers



$$\alpha_1 = \alpha_2 = \alpha_3 = 1.0$$

$$h_{vote}(x) = \arg \max_{y \in \mathcal{Y}} \frac{1}{N} \sum_{i=1}^N \alpha_i \mathbf{1}(h_i(x) = y)$$

Practice problem #1

Problem 4 Decision boundaries

Sketch the decision boundary and indicate the region that is classified into the positive class.

- The naive bayes classifier where $P(x|y) = \mathcal{N}([0, 1]^T, I_2)$ and $P(x|y) = \mathcal{N}([1, 2]^T, I_2)$ and $P(y = -1) = P(y = 1) = 0.5$.
- The classifier $h(x) = \text{sign}(x_1^2 - x_2 - 1)$.
- The voting classifier of two decision stumps h_1, h_2, h_3 . h_1 outputs 1 if and only if $x_1 < 0$. h_2 outputs 1 if and only if $x_1 > -1$. h_3 outputs 1 if and only if $x_2 \leq 1$.

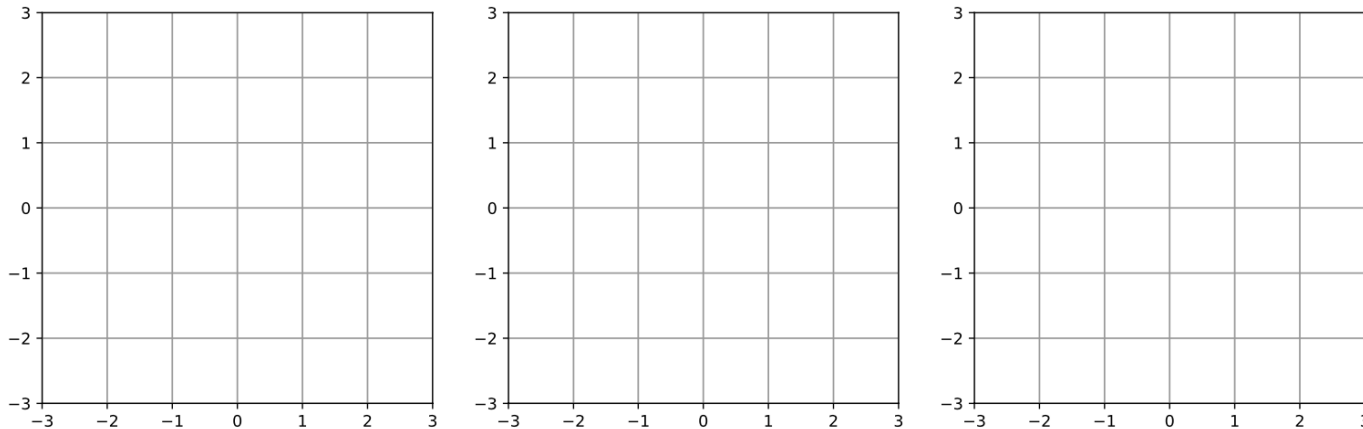


Figure 1: Decision boundaries for problem (a) (b) and (c) from left to right.

Two unsupervised learning problems we learned

- K-means clustering for clustering
 - The Lloyd's algorithm for solving k-means clustering
 - Fact: K-means is NP-hard. Lloyd's algorithm is a useful heuristic for finding a local solution for k-means in practice.
- Principal Component Analysis for dimension reduction
 - PCA can be solved in polynomial time using the magical SVD algorithm.

Practice Problem #2

Problem 6 Unsupervised Learning

K-means clustering Let the data points be $[1, 5, -2, 3]$. Let $K = 2$ and the two centroids be μ_1 and μ_2 . Fix $\mu_1 = 1$. Sketch the K-means objective function as you vary μ_2 .

$$\min_{\mu_1, \dots, \mu_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{j \in [k]} \|x_i - \mu_j\|^2$$

Problems, Learning Algorithm, and Inference Algorithm

Problem categories	Problems	Learning Algorithm	Inference
Unsupervised Learning	K-means clustering	Lloyd's algorithm / SGD	Assigning data points to clusters: $c \leftarrow x$
	Gaussian Mixture Model	The EM algorithm / SGD	Output density: $P(x)$ Inference: $P(c x)$
	Principal component analysis	The SVD algorithm	Reduce dimension. (Lossy) Reconstruction
Supervised Learning	Linear classification	Perceptron / SGD with logistic loss	Prediction $y \leftarrow \text{sign}(x^T w + b)$
	Naïve Bayes Classification	Solve MLE (with e.g., direct solver or SGD)	Prediction $p(y x)$
	Learning Kernel Machines	Solving a Quadratic Program or SGD	Prediction (e.g. for regression) $y \leftarrow \sum_{i=1} \alpha_K(x, x_i) y_i$
	Learning Voting Classifiers	Bagging, Boosting, AdaBoost	Prediction using voting classifiers
	Learning Neural Networks	SGD to learn ϕ , and w, b	Prediction by $y \leftarrow \text{sign}(\phi(x)^T w + b)$

Other types of problems in the final quiz

MCQs (choose only one answer)

1. Which of the following statements about Maximum Likelihood Estimation (MLE) is true?
 - a. MLE aims to maximize the probability of observing the data.
 - b. MLE is a randomized algorithm.
 - c. MLE can always be solved using SGD.
 - d. MLE is primarily used in unsupervised learning.
2. Which of the following is a characteristic of a decision tree?
 - a. Prone to overfitting with many features
 - b. Invariant to feature scaling
 - c. Not efficiently learnable even when the depth = 1 (e.g., decision stumps).
 - d. Unaffected by missing values

True or False:

1. Bagging and Boosting are techniques used to improve the performance of decision trees by reducing generalization error. (True/False)
2. A multinomial naive Bayes classifier is a linear classifier for the Bag-of-Word feature (True/False)

It's my pleasure teaching you all!
Good luck with the final!