CSC 311: Introduction to Machine Learning

Lecture 12 - Algorithmic Fairness and Final Exam Review

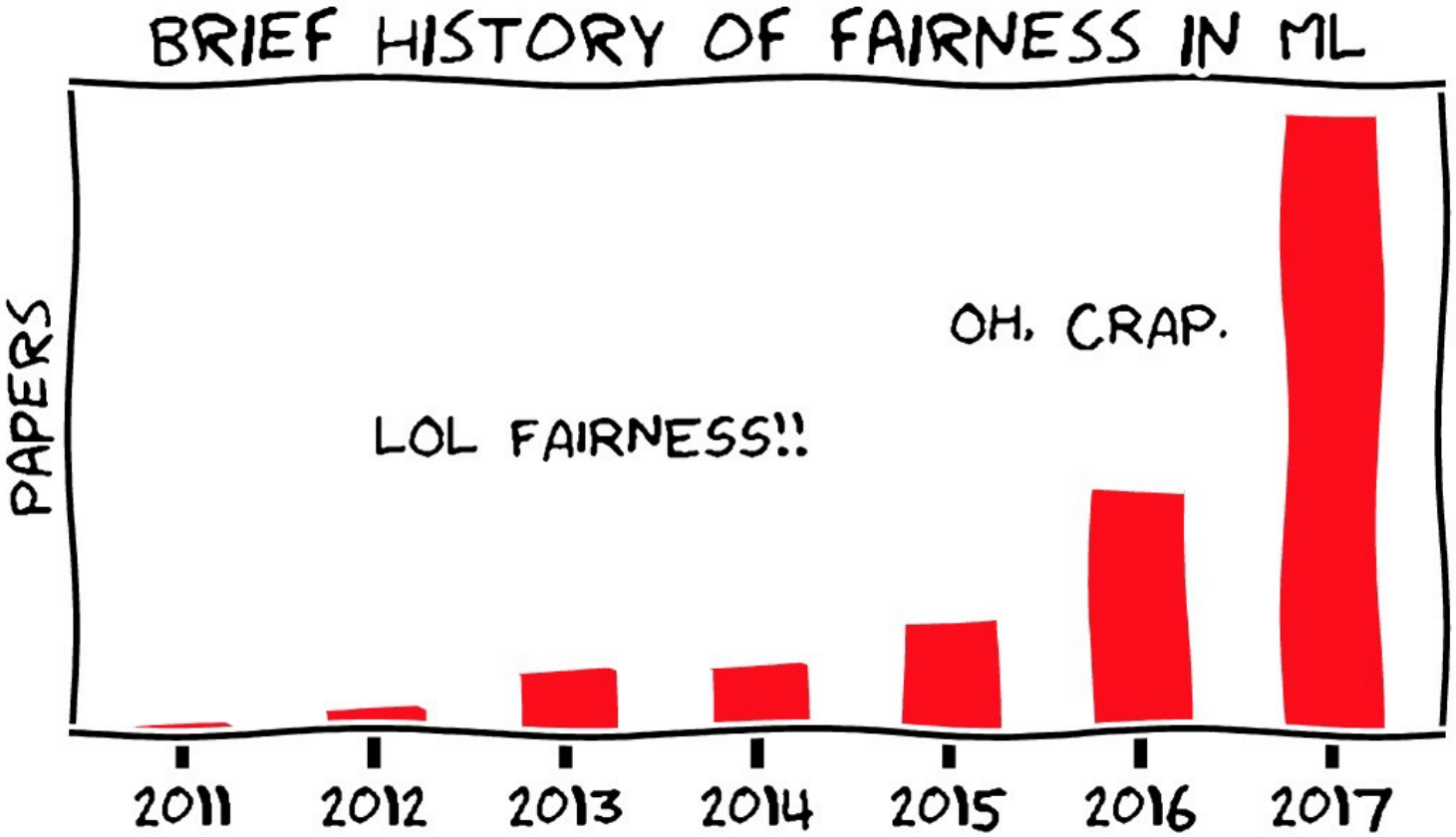
Richard Zemel & Murat A. Erdogdu

University of Toronto

Overview

- As ML starts to be applied to critical applications involving humans, the field is wrestling with the societal impacts
  - **Security:** what if an attacker tries to poison the training data, fool the system with malicious inputs, "steal" the model, etc.?
  - **Privacy:** avoid leaking (much) information about the data the system was trained on (e.g. medical diagnosis)
  - **Fairness:** ensure that the system doesn't somehow disadvantage particular individuals or groups
  - **Transparency:** be able to understand why one decision was made rather than another
  - **Accountability:** an outside auditor should be able to verify that the system is functioning as intended
- If some of these definitions sound vague, that's because formalizing them is half the challenge!

Overview: Fairness

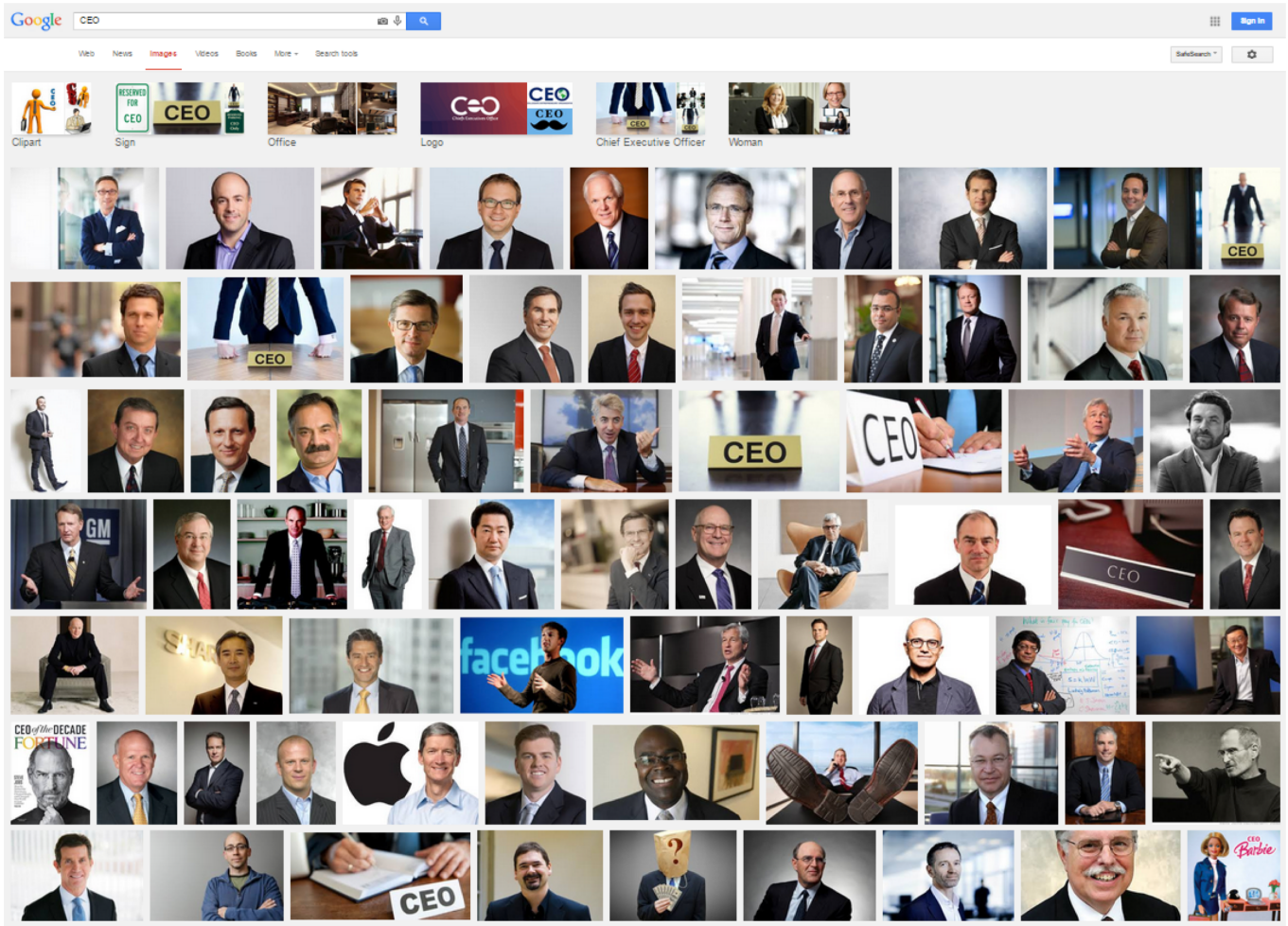


Credit: Moritz Hardt

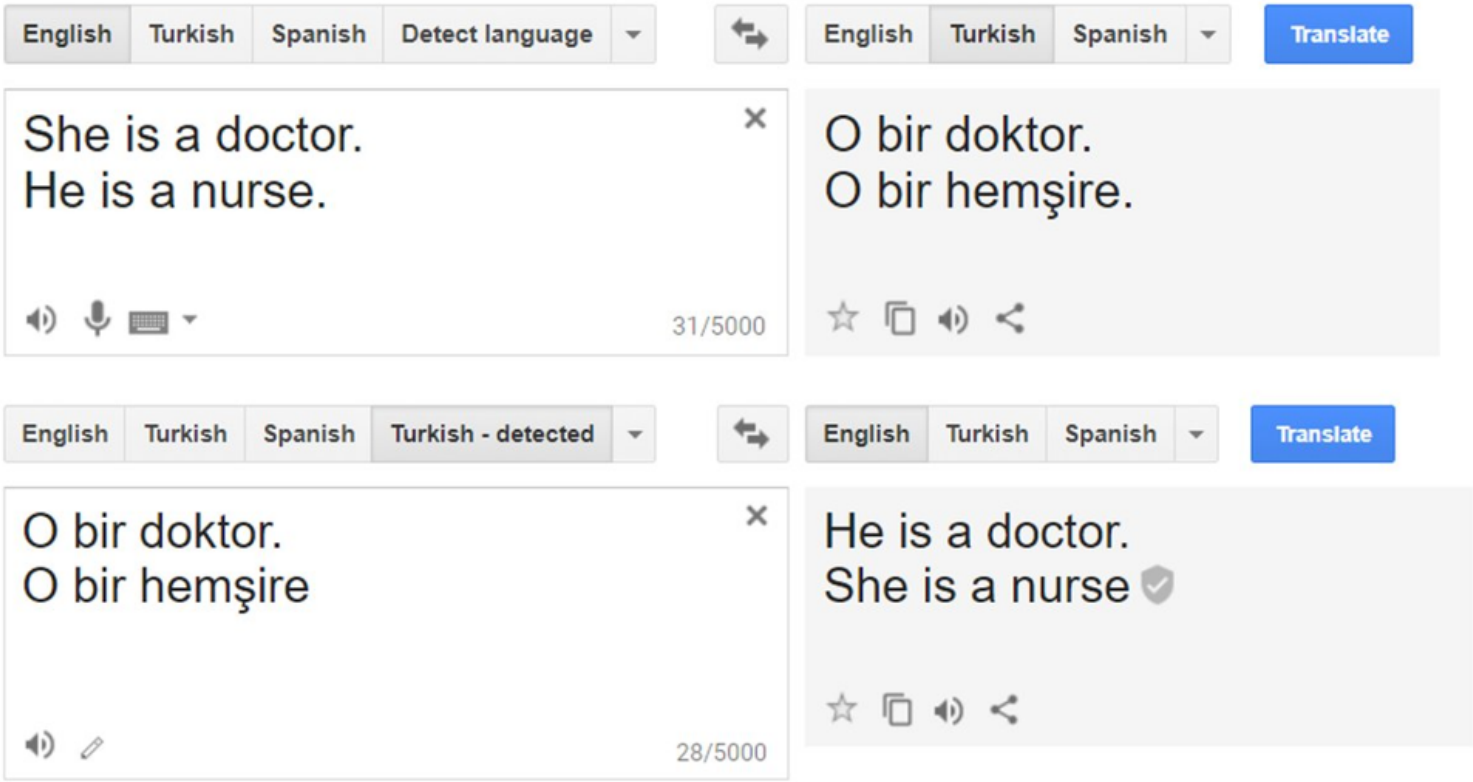
FAIRNESS IN AUTOMATED DECISIONS

Overview: Fairness

SUBTLER BIAS



Overview: Fairness



Turkish has gender neutral pronouns

Overview: Fairness

- This lecture: algorithmic fairness
- Goal: identify and mitigate **bias** in ML-based decision making, in all aspects of the pipeline
- Sources of bias/discrimination
  - Data
    - Imbalanced/impoverished data
    - Labeled data imbalance
    - Labeled data incorrect / noisy
  - Model
    - ML prediction error imbalanced
    - Compound injustices
- Important: Algorithmic fairness does not imply real fairness!

Learning Fair Representations

- A naïve attempt: simply don't use the sensitive feature.
  - Problem: the algorithm implicitly learns to predict the sensitive feature from other features (e.g. race from zip code)
- Another idea: limit the algorithm to a small set of features you're pretty sure are safe and task-relevant
  - This is the conservative approach, and commonly used for both human and machine decision making
  - But removing features hurts the classification accuracy. Maybe we can make more accurate decisions if we include more features and somehow enforce fairness algorithmically?
- Can we learn fair representations, which can make accurate classifications without implicitly using the sensitive attribute?

Overview: Fairness

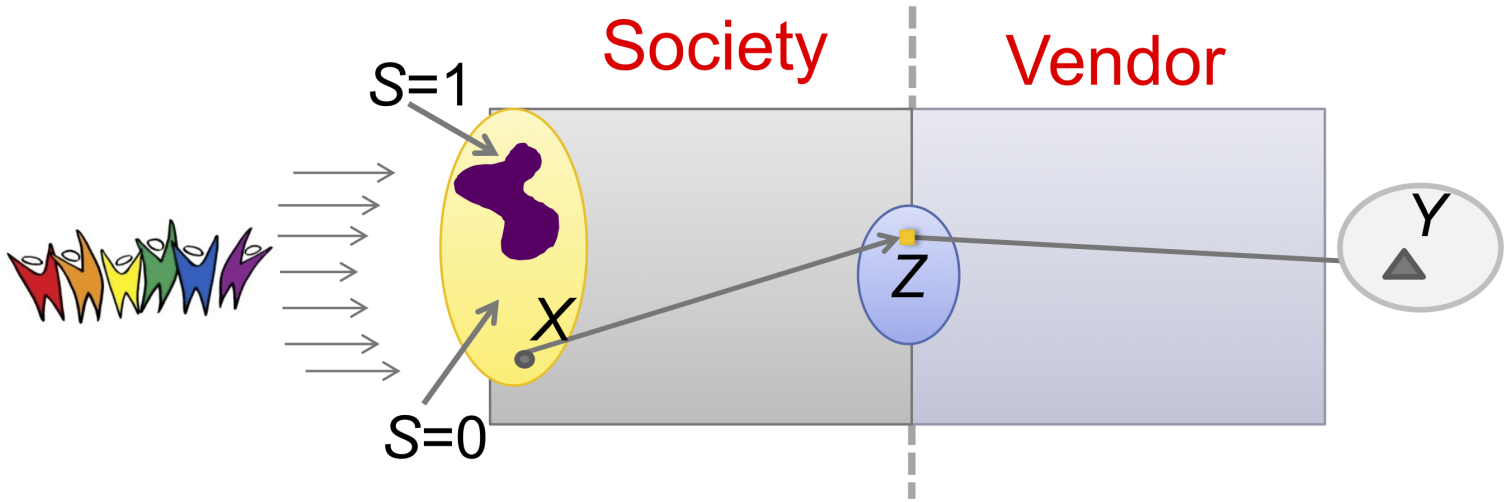
- Notation
  - $X \in \mathbb{R}^D$: input to classifier
  - $S \in \{0, 1\}$: belongs to protected group (age, gender, race, etc.)
  - $Z \in \{1, 2, ..., K\}$: latent representation (code)
  - $T \in \{0, 1\}$: true label
  - $Y \in [0, 1]$: prediction ($p(T = 1 \mid X)$)
- We use capital letters to emphasize that these are random variables.

Fairness Criteria

- $X \perp\!\!\!\perp Y$ means X and Y are independent
- Most common way to define fair classification is to require some invariance with respect to the sensitive attribute
  - Demographic parity: $Y \perp\!\!\!\perp S$
  - Equalized odds: $Y \perp\!\!\!\perp S \,|\, T$
  - Equal opportunity: $Y \perp\!\!\!\perp S \,|\, T = t$, for a fixed t
  - Equal (weak) calibration: $T \perp\!\!\!\perp S \,|\, Y$
  - Equal (strong) calibration: $T \perp\!\!\!\perp S \,|\, Y$ and $Y = \Pr(T = 1)$
  - Fair subgroup accuracy: $\mathbb{1}[T = Y] \perp\!\!\!\perp S$
- Many of these definitions are incompatible!

Learning Fair Representations

- Idea: separate the responsibilities of the (trusted) society and (untrusted) vendor



- Goal: find a representation Z that removes any information about the sensitive attribute
- Then the vendor can do whatever they want!

Learning Fair Representations

Desiderata for the representation:

- Retain information about $X \Rightarrow$ high mutual information between X and Z
- Obfuscate $S \Rightarrow$ low mutual information between S and Z
- Allow high classification accuracy $\Rightarrow$ high mutual information between T and Z

Learning Fair Representations

First approach: Zemel et al., 2013, "Learning fair representations"

- Let Z be a discrete code or representation (like K-means, PCA)
- Determine Z based on distance to (the cluster center in K-means)

$$r_k^{(i)} = p(Z = k \,|\, \mathbf{x}^{(i)}) \propto \exp(-\beta \|\mathbf{x}^{(i)} - \mathbf{v}_k\|^2),$$

  where $\beta > 0$ is a constant, and $\mathbf{v}_k$ is a prototype for the cluster.
- Need to fit the prototypes $\mathbf{v}_k$. They are unknown.
- Similar to EM update, we let the reconstruction be

$$\tilde{\mathbf{x}}^{(i)} = \sum_{k=1}^{K} r_k^{(i)} \mathbf{v}_k$$

  and enforce that $\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)}$ by minimizing

$$\mathcal{L}_{\text{reconst}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2.$$

Learning Fair Representations

- Remember, we want to train a fair **classifier**.
- We predict using a linear function of $\mathbf{r}^{(i)} = [r_1^{(i)}, r_2^{(i)}, ..., r_K^{(i)}]^\top$.

$$y^{(i)} = \sigma(\mathbf{w}^\top \mathbf{r}^{(i)}) = p(t^{(i)}|\mathbf{x}^{(i)})$$

- Need to fit weights $\mathbf{w}$. They are unknown.
- Loss: we can use cross-entropy

$$L_{\text{CE}}(y^{(i)}, t^{(i)}) = -t^{(i)} \log y^{(i)} - (1 - t^{(i)}) \log(1 - y^{(i)})$$

Learning Fair Representations

- Next, enforce a fairness constraint:

$$\mathcal{L}_{\text{discrim}} = \frac{1}{K} \sum_{k=1}^{K} \left| \frac{1}{N_0} \sum_{i:s^{(i)}=0} p(Z = k \,|\, \mathbf{x}^{(i)}) - \frac{1}{N_1} \sum_{i:s^{(i)}=1} p(Z = k \,|\, \mathbf{x}^{(i)}) \right|.$$

- $N_0 = \#\{i : s^{(i)} = 0\}$, $N_1 = \#\{i : s^{(i)} = 1\}$ and $N_0 + N_1 = N$.

- Next, we show this enforces **demographic parity**.

- Note that $(Z \,|\, X) \perp\!\!\!\perp S$.

Learning Fair Representations

- Enforce **demographic parity** by obfuscating S:

$$\mathcal{L}_{\text{discrim}} = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{1}{N_0}\sum_{i:s^{(i)}=0} p(Z=k\,|\,\mathbf{x}^{(i)},s^{(i)}) - \frac{1}{N_1}\sum_{i:s^{(i)}=1} p(Z=k\,|\,\mathbf{x}^{(i)},s^{(i)})\right|,$$

- $N_0 = \#\{i : s^{(i)} = 0\}$, $N_1 = \#\{i : s^{(i)} = 1\}$ and $N_0 + N_1 = N$.
- If the above discrimination loss is $\mathcal{L}_{\text{discrim}} = 0$, we have LHS=RHS for all $k = 1, 2, ..., K$. Therefore,

$$\begin{aligned}
p(Y=1\,|\,S=1) &= \sum_k p(Y=1\,|\,Z=k)p(Z=k\,|\,X,S=1) \\
&\approx \sum_k p(Y=1\,|\,Z=k)\frac{1}{N_1}\sum_{i:s^{(i)}=1} p(Z=k\,|\,\mathbf{x}^{(i)},s^{(i)}=1) \\
&= \sum_k p(Y=1\,|\,Z=k)\frac{1}{N_0}\sum_{i:s^{(i)}=0} p(Z=k\,|\,\mathbf{x}^{(i)},s^{(i)}=0) \\
&\approx \sum_k p(Y=1\,|\,Z=k)p(Z=k\,|\,X,S=0) \\
&= p(Y=1\,|\,S=0) \quad \textbf{demographic parity}
\end{aligned}$$

Learning Fair Representations

- We want to retain information about X: $\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)}$ penalize reconstruction error

$$\mathcal{L}_{\text{reconst}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2$$

- Predict accurately: cross-entropy loss

$$\mathcal{L}_{\text{pred}} = \frac{1}{N} \sum_{i=1}^{N} -t^{(i)} \log y^{(i)} - (1 - t^{(i)}) \log(1 - y^{(i)})$$

- Obfuscate S:

$$\mathcal{L}_{\text{discrim}} = \frac{1}{K} \sum_{k=1}^{K} \left| \frac{1}{N_0} \sum_{i:s^{(i)}=0} p(Z = k \,|\, \mathbf{x}^{(i)}) - \frac{1}{N_1} \sum_{i:s^{(i)}=1} p(Z = k \,|\, \mathbf{x}^{(i)}) \right|.$$

Learning Fair Representations

- We can solve the following problem

$$\mathcal{L}_{\text{total}}(\{\mathbf{v}_k\}_{k=1}^K, \mathbf{w}) = \lambda_r \mathcal{L}_{\text{reconst}} + \lambda_p \mathcal{L}_{\text{pred}} + \lambda_d \mathcal{L}_{\text{discrim}}$$

  where λ_r, λ_p, and λ_d are hyperparameters governing the trade-off between losses.
- We can find the optimal parameter $\{\mathbf{v}_k\}_{k=1}^K, \mathbf{w}$ using an optimization method such as gradient descent.

Learning Fair Representations

Datasets

1. **German Credit**

   Task: classify individual as good or bad credit risk

   Sensitive feature: Age

2. **Adult Income**

   Size: 45,222 instances, 14 attributes

   Task: predict whether or not annual income > 50K

   Sensitive feature: Gender

3. **Heritage Health**

   Size: 147,473 instances, 139 attributes

   Task: predict whether patient spends any nights in hospital

   Sensitive feature: Age

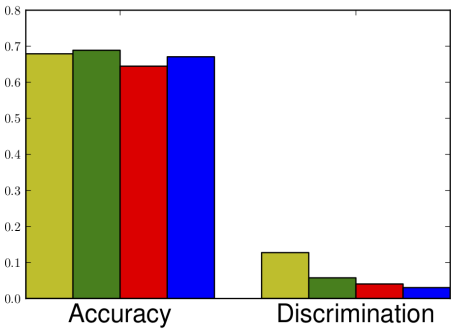
Learning Fair Representations

Metrics

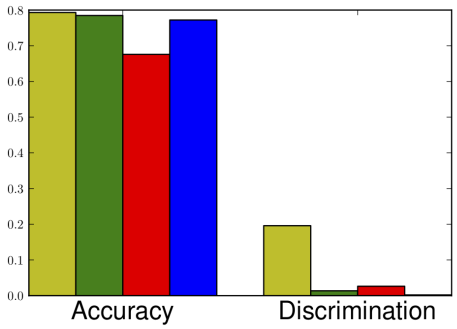
- Classification accuracy
- Discrimination: measuring the difference in proportion of positive classification of individuals in the protected or unprotected groups.

$$\left| \frac{\sum_{i:s^{(i)}=1}^{N} y^{(i)}}{N_1} - \frac{\sum_{i:s^{(i)}=0}^{N} y^{(i)}}{N_0} \right|$$

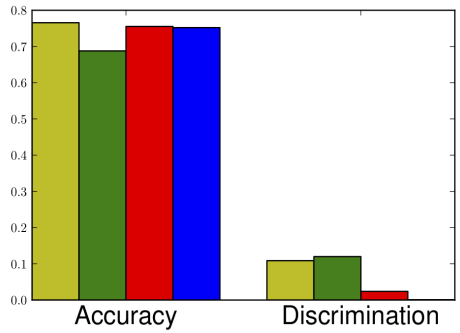
German



Adult



Health



Blue = theirs, others: logistic reg (LR), naive Bayes, regularized LR

Individual Fairness

- The work on fair representations was geared towards group fairness
- Another notion of fairness is individual level: ensuring that similar individuals are treated similarly by the algorithm
  - This depends heavily on the notion of "similar".
- One way to define similarity is in terms of the "true label" T (e.g. whether this individual is in fact likely to repay their loan)
  - Can you think of a problem with this definition?
  - The label may itself be biased
    - if based on human judgments
    - if, e.g., societal biases make it harder for one group to pay off their loans
  - Keep in mind that you'd need to carefully consider the assumptions when applying one of these methods!

Equal Opportunity

- Example: predict whether an individual is likely to repay their loan
- Two notions of individual fairness:
  - **Equalized odds**: equal true positive and false positive rates

$$p(Y = 1 \,|\, S = 0, T = t) = p(Y = 1 \,|\, S = 1, T = t) \quad \text{for } t \in \{0, 1\}$$

  - **Equal opportunity**: equal true positive rates

$$p(Y = 1 \,|\, S = 0, T = 1) = p(Y = 1 \,|\, S = 1, T = 1)$$

Fairness Summary

- Fairness is a challenging issue to address
  - Not something you can just measure on a validation set
  - Philosophers and lawyers have been trying to define it for thousands of years
  - Different notions are incompatible. Need to carefully consider the particular problem.
    - individual vs. group
- Explosion of interest in ML over the last few years
- Conference on Fairness, Accountability, and Transparency (FAT*)
- New textbook: `https://fairmlbook.org/`

Final Exam

- Dec 13, at 9am–12pm EX200.
- Covers all lectures except this one.
- It doesn't cover material only covered in tutorials.
- Similar difficulty to midterm.
- Practice final exam will be posted.

Basic ML Terminology

The final exam will be on the entire course; however, it will be more weighted towards post-midterm material. For pre-midterm material, refer to the midterm review slides (second half of tutorial 6) on the course website.

- Generalization, overfitting, underfitting
- Curse of dimensionality
- Bias-Variance decomp
- Training, test, validation
- Neural Network, backprop
- Cross-entropy, softmax
- MLE, MAP, prior, posterior
- PCA, Dimension Reduction, projection on subspace
- Matrix factorization, SVD, EigenVD
- Generative vs. Discriminative Methods, Bayes rule
- Naive Bayes, Gaussian Discriminant Analysis
- SVMs, Ensembles: Bagging, boosting
- (soft) k-Means, EM algorithm
- Value function, Q-learning, Bellman equations

Lec 6: Support Vector Machines

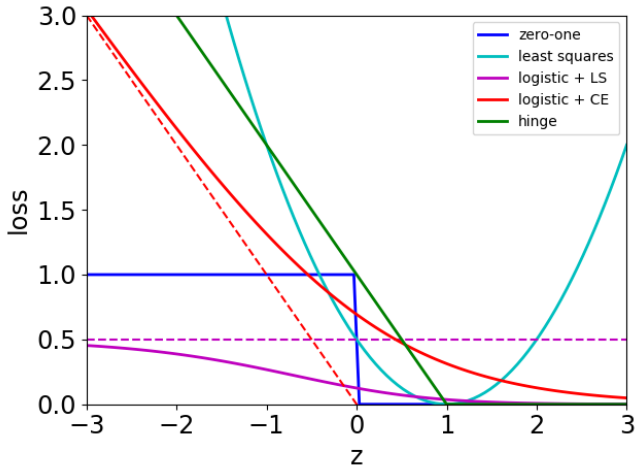
Previously, we saw loss functions as relaxations to 0-1 loss. For SVMs, we consider a different relaxation: **Hinge loss**

$$\text{0-1 loss:} \quad \mathcal{L}_{0-1}(z,t) = \mathbb{I}\{\text{sign}(z) \neq t\}$$

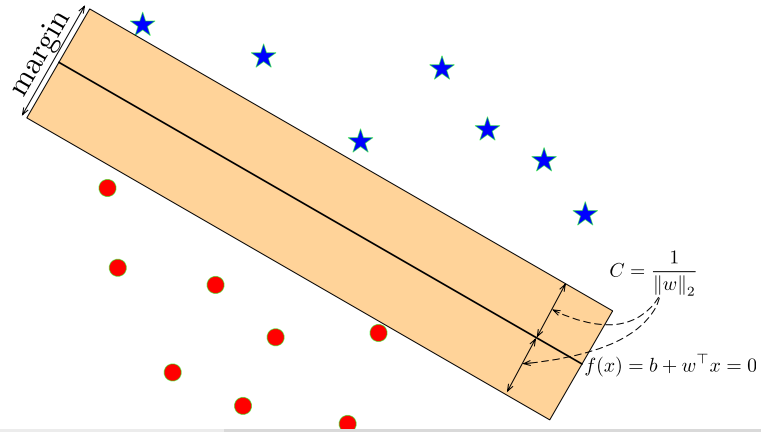
$$\text{Hinge loss:} \quad \mathcal{L}_{\text{H}}(z,t) = \max\{0, 1-zt\}$$

$$\text{SVM loss:} \quad \min_{\mathbf{w},b} \sum_{i=1}^{N} \max\{0, 1 - t^{(i)} z^{(i)}(\mathbf{w}, b)\} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Classification w/ Different losses



What does SVM do?: Optimal Separating Hyperplane



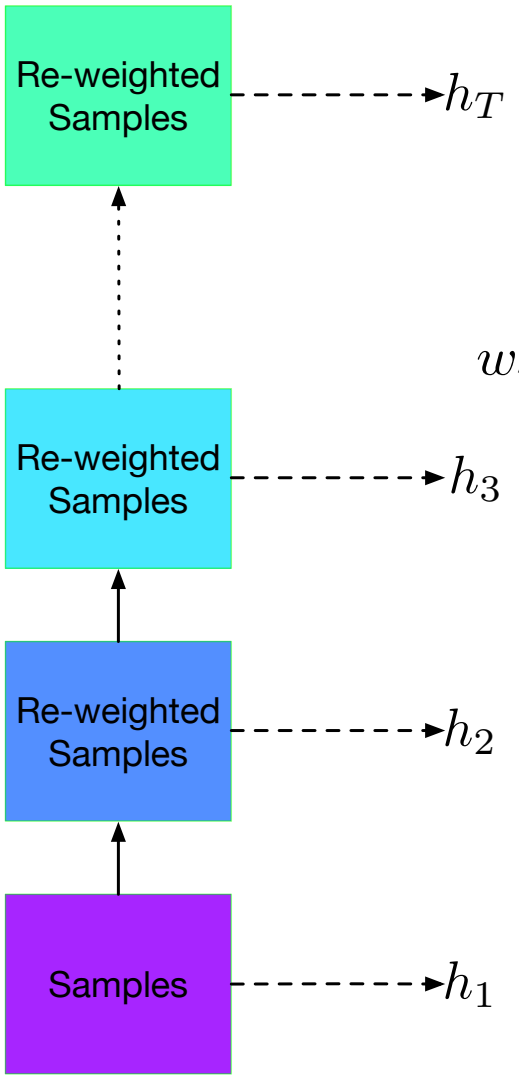
Lec 6: Ensembles

- Recall bias-variance trade-off from first half.
- Ensembles combine classifiers to improve performance
- Bagging
  - Reduces variance (large ensemble can't cause overfitting)
  - Bias is not changed
  - Parallel
  - Want to minimize correlation between ensemble elements.
- Boosting
  - Reduces bias
  - Increases variance (large ensemble can cause overfitting)
  - Sequential
  - High dependency between ensemble elements

Lec 6: AdaBoost

- **Boosting**: Key steps of AdaBoost:
  1. At each iteration we re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
  2. We train a new weak classifier based on the re-weighted samples.
  3. We add this weak classifier to the ensemble of weak classifiers. This ensemble is our new classifier.
  4. We repeat the process many times.
- The weak learner needs to minimize weighted error.
- **AdaBoost reduces bias by making each classifier focus on previous mistakes.**
- AdaBoost's training error (loss) converges to zero.
- If one runs AdaBoost long enough, it can in fact overfit. Generally, it doesn't.

Lec 6: AdaBoost Algorithm



$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$$w_i \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})}) \neq \mathbf{t}^{(\mathbf{i})}\}\right)$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\text{err}_t}{\text{err}_t}\right)$$

$$\text{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})}) \neq \mathbf{t}^{(\mathbf{i})}\}}{\sum_{i=1}^{N} w_i}$$

Lec 7: Probabilistic models

Two approaches to classification:

- **Discriminative approach:** estimate parameters of decision boundary/class separator directly from labeled examples.
  - Tries to solve: How do I separate the classes?
  - learn $p(t|\mathbf{x})$ directly (logistic regression models)
  - learn mappings from inputs to classes (linear/logistic regression, decision trees etc)
- **Generative approach:** model the distribution of inputs characteristic of the class (Bayes classifier).
  - Tries to solve: What does each class "look" like?
  - Build a model of $p(\mathbf{x}|t)$
  - Apply Bayes Rule
- Key difference: is there a distribution over inputs $\mathbf{x}^{(i)}$?

Lec 7: Probabilistic models

- **MLE:** Write the log-likelihood, take derivative and set it to zero:

$$\theta^{\text{MLE}} = \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} p(\mathbf{x}_i|\theta) = \operatorname*{argmax}_{\theta} \log \left(\prod_{i=1}^{N} p(\mathbf{x}_i|\theta) \right),$$

$$= \operatorname*{argmax}_{\theta} \sum_{i=1}^{N} \log p(\mathbf{x}_i|\theta).$$

- **Bayesian methods:** rely on prior distribution.
- Apply **Bayes' Rule**

$$p(c \,|\, \mathbf{x}) = \frac{p(c)p(\mathbf{x} \,|\, c)}{\sum_{c'} p(c')p(\mathbf{x} \,|\, c')} \propto p(c)p(\mathbf{x} \,|\, c)$$

- **MAP:** uses Bayes rule to maximize posterior distribution.

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \,|\, \mathcal{D}) = \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta})\, p(\mathcal{D} \,|\, \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \; \log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \,|\, \boldsymbol{\theta})$$

Lec 7: Probabilistic models

In Bayesian models, we generally need simplifying assumptions such as

- **Naïve Bayes** assumes that the word features x_i are **conditionally independent** given the class c.
  - This means x_i and x_j are independent under the conditional distribution $p(\mathbf{x}|c)$.
  - Note: this doesn't mean they're independent.
  - Bayes rule gives (under Naive Bayes assumption):

$$p(c \,|\, \mathbf{x}) = \frac{p(c) \prod_{j=1}^{D} p(x_j \,|\, c)}{\sum_{c'} p(c') \prod_{j=1}^{D} p(x_j \,|\, c')} \propto p(c) \prod_{j=1}^{D} p(x_j \,|\, c)$$

  - Compact representation of the joint distribution: $2D + 1$ parameters total (before $2^{D+1} - 1$)

- Alternatively, $p(\mathbf{x} \,|\, c = k) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ which leads to Gaussian Discreminant Analysis.

Lec 8: Principal Component Analysis

- Dimensionality reduction: map data to a lower dimensional space
  - Save computation/memory
  - Reduce overfitting, achieve better generalization
  - Visualize in 2 dimensions
- We can approach this problem by:
  - Minimize the **reconstruction error**: Find vectors in a subspace that are closest to data points.

$$\min_{\mathbf{U}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2$$

  - Maximize the **variance of reconstructions**: Find a subspace where data has the most variability.

$$\max_{\mathbf{U}} \frac{1}{N} \sum_{i} \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2$$

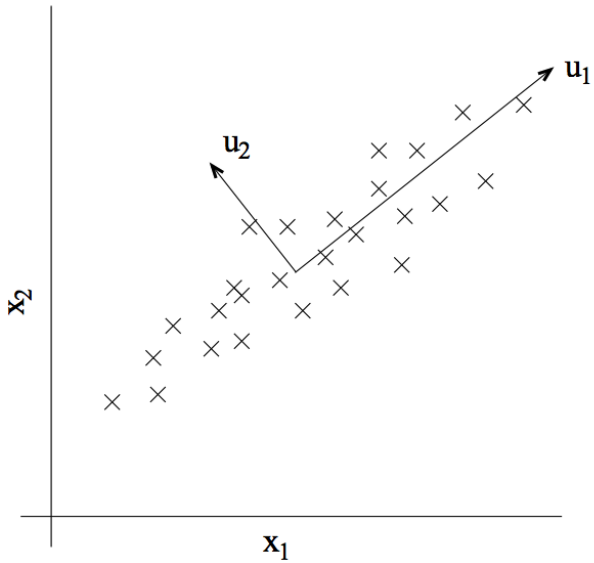
  - Both of them lead to PCA.

Lec 8: Principal Component Analysis

- Consider the **empirical covariance matrix**:

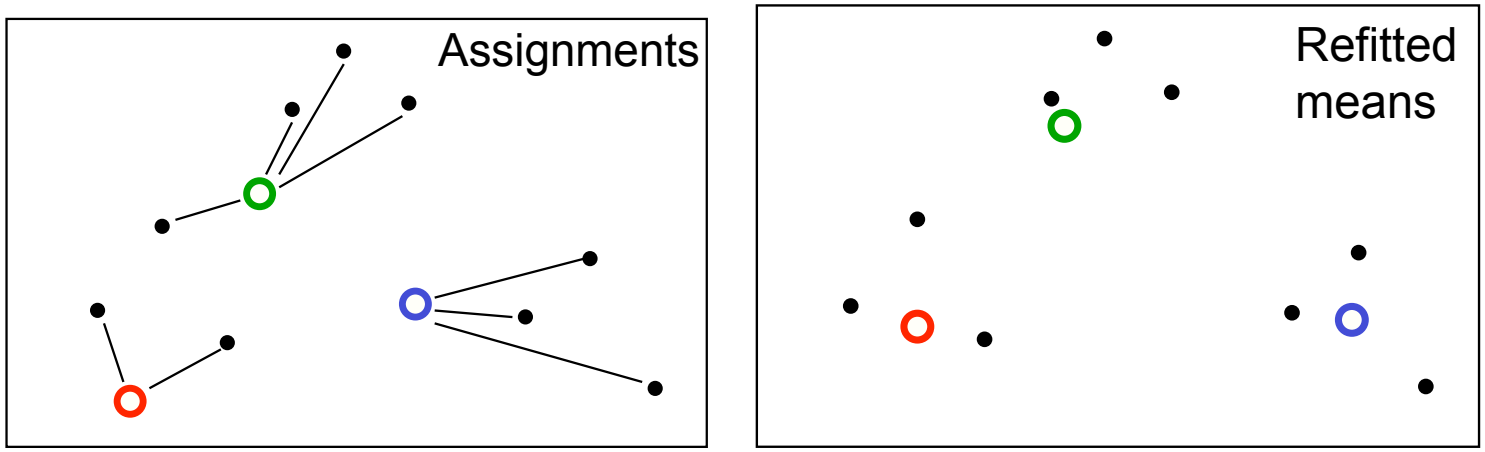
$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top$$

- Recall: Covariance matrices are symmetric and positive semidefinite.
- The optimal PCA subspace is spanned by the top K eigenvectors of $\hat{\boldsymbol{\Sigma}}$.
  - More precisely, choose the first K of any orthonormal eigenbasis for $\hat{\boldsymbol{\Sigma}}$.
  - The general case was tricky, but we derived this for $K = 1$.
- These eigenvectors are called **principal components**, analogous to the principal axes of an ellipse.



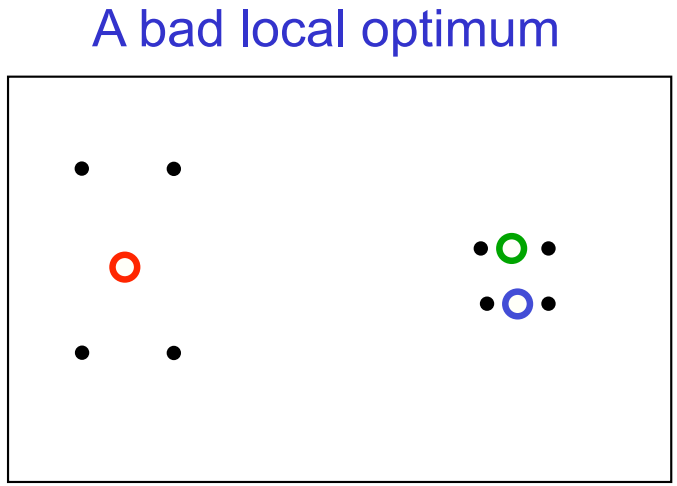
High level overview of algorithm:

- **Initialization**: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - **Assignment step**: Assign each data point to the closest cluster
  - **Refitting step**: Move each cluster center to the mean of the data assigned to it



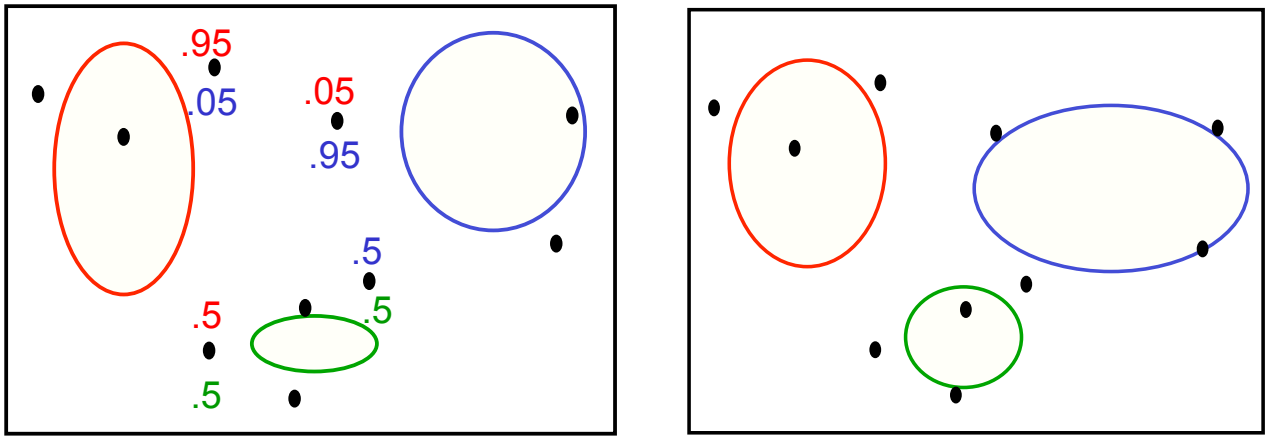
Lec 9: k-Means Algorithm

- The objective is non-convex (so coordinate descent is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points



Lec 9: EM Algorithm

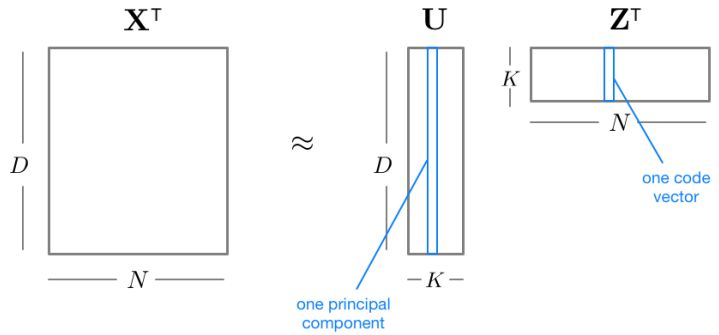
- **Expectation-Maximization alg** alternates between 2 steps:
  1. **E-step**: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ given our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.
  2. **M-step**: Use the equations derived in lec 9 to update the parameters, assuming $r_k^{(n)}$ are held fixed- change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.



- When used with general covariance, it can capture the shape of data. K-means with Euclidean distance cannot.

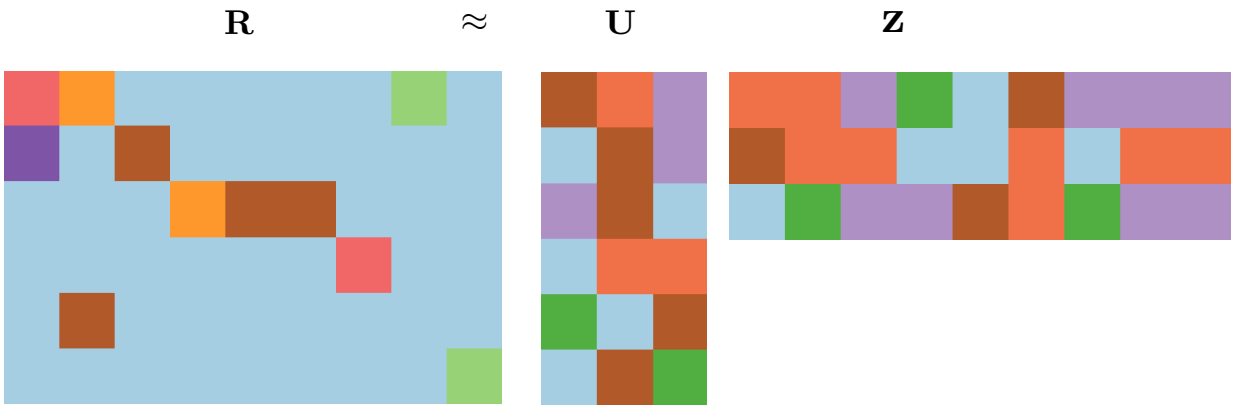
Lec 10: PCA as Matrix Factorization

- PCA is approximating $\mathbf{X} \approx \mathbf{Z}\mathbf{U}^\top$, or equivalently $\mathbf{X}^\top \approx \mathbf{U}\mathbf{Z}^\top$.



- Based on the sizes of the matrices, this is a rank-K approximation.
- Write SVD of data matrix: $\mathbf{X} = \mathbf{Q}\mathbf{S}\mathbf{U}^\top$ and let $\mathbf{Z} = \mathbf{Q}\mathbf{S}$
  - First K principal components corresponds first K columns of $\mathbf{U}$, i.e., $\mathbf{U}[:, :K]$ in python notation.
  - PCA reduces the dimension of data D to K. Low-dimensional representation is given by the first K columns of $\mathbf{Z}[:, :K]$. Rows of this matrix are the K-dimensional code vectors.
- Since $\mathbf{U}$ was chosen to minimize reconstruction error, this is the *optimal* rank-K approximation, in terms of error $\|\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top\|_F^2$.

Lec 10: Matrix Factorization & Recommender Systems

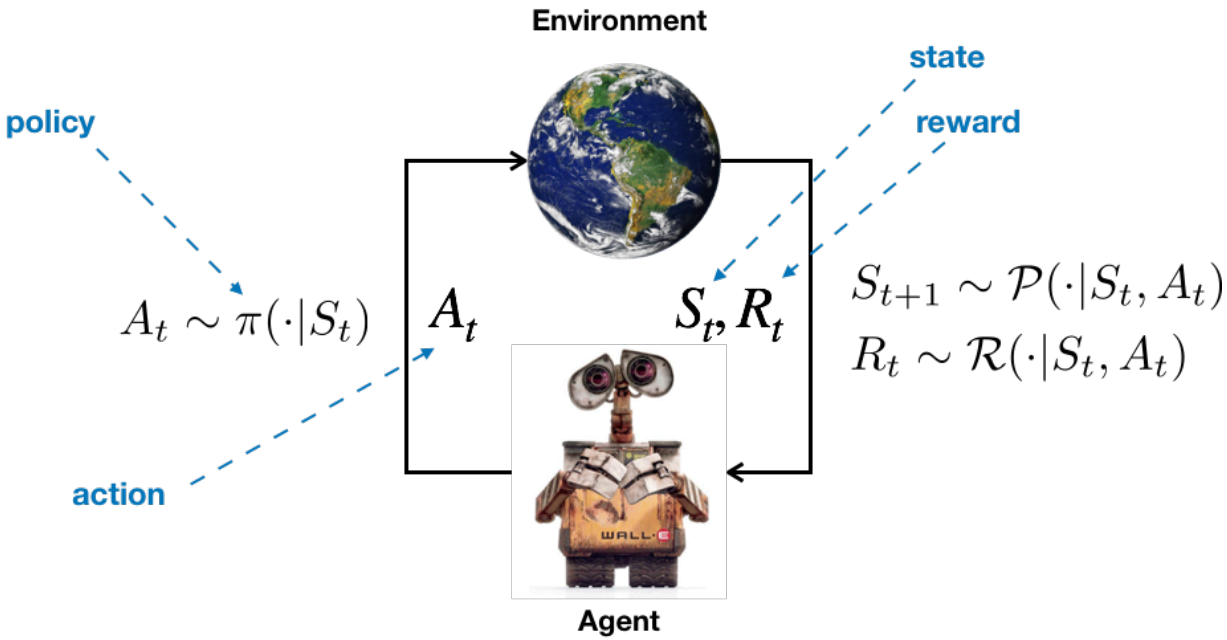


- How do we enforce that rating matrix satisfies $\mathbf{R} \approx \mathbf{U}\mathbf{Z}^\top$
  - Try
$$\min_{\mathbf{U},\mathbf{Z}} \sum_{i,j} (R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j)^2$$
  - Most entries of $\mathbf{R}$ are missing! (unseen movies)
- Let $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } \mathbf{R} \text{ is observed}\}$
- Using the squared error loss, a matrix factorization corresponds to solving
$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(n,m)\in O} \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m\right)^2$$
- Use alternating least squares, stochastic gradient descent, etc to minimize it.

Lec 11: Reinforcement Learning

A discounted MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.

- $\mathcal{S}$: State space. Discrete or continuous
- $\mathcal{A}$: Action space. Here we consider finite action space, i.e., $\mathcal{A} = \{a_1, \ldots, a_M\}$.
- $\mathcal{P}$: Transition probability
- $\mathcal{R}$: Immediate reward distribution
- γ: Discount factor $(0 \leq \gamma \leq 1)$



Lec 11: Reinforcement Learning

The value function is the expected discounted reward if the agent starts from state s, takes action a, and afterwards follows policy π, and satisfies the following recursive relationship:

$$\text{Bellman equations:} \quad Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a \right]$$

$$\text{Bellman operator:} \quad = \underbrace{r(s,a) + \gamma \int_{\mathcal{S}} Q^\pi(s', \pi(s')) \mathcal{P}(s'|s,a) \mathrm{d}s'}_{\triangleq (T^\pi Q^\pi)(s,a)}$$

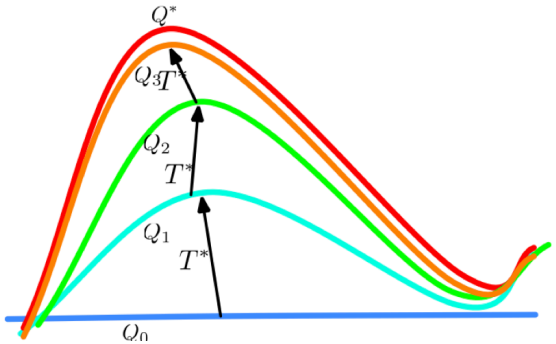
$$\text{Bellman Opt Operator:} \quad (T^* Q)(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} \max_{a' \in \mathcal{A}} Q(s', a') \mathcal{P}(s'|s,a) \mathrm{d}s'$$

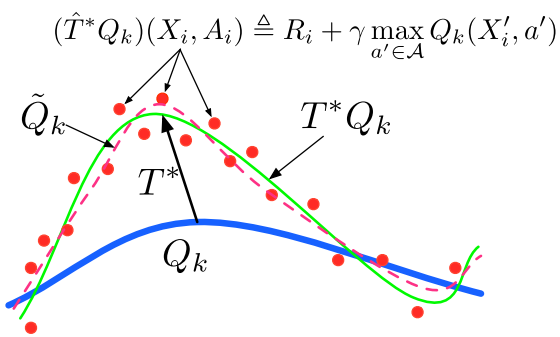
- Key observation for $Q^*(s,a) = \sup_\pi Q^\pi(s,a) = \max_\pi Q^\pi(s,a)$:

$$Q^* = T^* Q^*$$

- **If we find a Q such that $T^* Q = Q$, then $Q = Q^*$.** We just need to find a fixed point of the operator T^*.

Three algorithms:





1. Value iteration: $Q_{k+1} \leftarrow T^* Q_k$
   It assumes the **model is known** (distribution $\mathcal{P}(\cdot|s,a)$)
2. Batch RL: **Given the dataset** $\mathcal{D}_N = \{(S_i, A_i, R_i, S_i')\}_{i=1}^N$ and an action-value function estimate Q_k, we solve a regression problem. We minimize the squared error:

$$Q_{k+1} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \left| Q(S_i, A_i) - \left(R_i + \gamma \max_{a' \in \mathcal{A}} Q_k(S_i', a) \right) \right|^2$$

3. Q-learning: **Online** update for the action-value function at state S_t:

$$A_t \leftarrow \pi_\varepsilon(S; Q) = \begin{cases} \operatorname{argmax}_{a \in \mathcal{A}} Q(S, a) & \text{with probability } 1 - \varepsilon \\ \text{Uniformly random action in } \mathcal{A} & \text{with probability } \varepsilon \end{cases}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') - Q(S_t, A_t) \right]$$

CSC 311 Closing Remarks

Continuing with machine learning

- Courses
  - CSC 412/2506, "Probabilistic Graphical Models"
  - CSC 421/2516, "Neural Networks and Deep Learning"
  - CSC 2515, "Machine Learning"
  - CSC 2532, "Statistical Learning Theory"
  - Topics courses (varies from year to year): Reinforcement Learning, Algorithmic Fairness, Computer Vision w/ ML, NLP w/ ML, Health w/ ML etc.
- Videos from top ML conferences (NeurIPS, ICML, ICLR, UAI)
- Try to reproduce results from papers
  - If they've released code, you can use that as a guide if you get stuck
- Lots of excellent free resources avaiable online!

Thank you!