The standard approach to machine learning uses a training set of labeled examples to learn a prediction rule that will predict the labels of new examples. Collecting such training sets can be expensive and time-consuming. This course will explore methods that leverage already-collected data to guide future measurements, in a closed loop, to best serve the task at hand. We focus on two paradigms: i) in pure-exploration we desire algorithms that identify or learn a good model using as few measurements as possible (e.g., classification, drug discovery, science), and ii) in regret minimization we desire algorithms that balance taking measurements to learn a model with taking measurements to exploit the model to obtain high reward outcomes (e.g., medical treatment design, ad-serving). The course will assume introductory machine learning (e.g., CSE 546) and maturity in topics like linear algebra, statistics, and calculus. The course will be analysis heavy, with a focus on methods that work well in practice.
• Fill in the missing plots:

\[ \Sigma = X^T J J X = Z^T J J Z \]
\[ V S V^T = \text{eig}(\Sigma) \]
\[ J = I - 11^T / n \]
\[ \mu_X = X^T 1/n \quad \mu_Z = Z^T 1/n \]

\[ X \quad Z \quad \mu_X - \mu_Z \quad V S^{-1/2} V^T (\mu_X - \mu_Z) \]
Linear projections

Given $x_i \in \mathbb{R}^d$ and some $q < d$ consider

$$\min_{V_q} \sum_{i=1}^{N} \| (x_i - \bar{x}) - V_q V_q^T (x_i - \bar{x}) \|^2.$$ 

where $V_q = [v_1, v_2, \ldots, v_q]$ is orthonormal:

$$V_q^T V_q = I_q$$

$V_q$ are the first $q$ eigenvectors of $\Sigma$

$V_q$ are the first $q$ principal components

Principal Component Analysis (PCA) projects $(X - 1\bar{x}^T)$ down onto $V_q$

$$(X - 1\bar{x}^T)V_q = U_q \text{diag}(d_1, \ldots, d_q)$$

$U_q^T U_q = I_q$

Singular Value Decomposition defined as

$$X - 1\bar{x}^T = U S V^T$$
Dimensionality reduction

\( V_q \) are the first \( q \) eigenvectors of \( \Sigma \) and SVD \( X - 1\bar{x}^T = USV^T \)
Power method - one at a time

\[ \Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \quad v_* = \arg \max_v v^T \Sigma v \]

\[ v_{k+1} = \frac{\Sigma v_k}{\|\Sigma v_k\|} \quad v_0 \sim \mathcal{N}(0, I) \]
Matrix completion

Given historical data on how users rated movies in past:

17,700 movies, 480,189 users, 99,072,112 ratings (Sparsity: 1.2%)

Predict how the same users will rate movies in the future (for $1 million prize)
Matrix completion

n movies, m users, |S| ratings

\[
\text{arg min}_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j,s) \in S} \| (UV^T)_{i,j} - s_{i,j} \|_2^2
\]

How do we solve it? With full information?
n movies, m users, |S| ratings

\[
\arg \min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j,s) \in S} \left\| (UV^T)_{i,j} - s_{i,j} \right\|_2^2
\]
Random projections

PCA finds a low-dimensional representation that reduces population variance

$$\min_{V_q} \sum_{i=1}^{N} \| (x_i - \bar{x}) - V_q V_q^T(x_i - \bar{x}) \|^2.$$  

$V_q V_q^T$ is a projection matrix that minimizes error in basis of size $q$

$V_q$ are the first $q$ eigenvectors of $\Sigma$

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

But what if I care about the reconstruction of the individual points?

$$\min_{W_q} \max_{i=1,\ldots,n} \| (x_i - \bar{x}) - W_q W_q^T(x_i - \bar{x}) \|^2$$
Random projections

$$\min_{W_q} \max_{i=1,\ldots,n} \|(x_i - \bar{x}) - W_q W_q^T (x_i - \bar{x})\|^2$$

Johnson-Lindenstrauss (1983)

**Theorem 1.1.** (Johnson-Lindenstrauss) Let $\epsilon \in (0, 1/2)$. Let $Q \subset \mathbb{R}^d$ be a set of $n$ points and $k = \frac{20 \log n}{\epsilon^2}$. There exists a Lipschitz mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $u, v \in Q$:

$$(1 - \epsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2$$

**Theorem 1.2.** (Norm preservation) Let $x \in \mathbb{R}^d$. Assume that the entries in $A \subset \mathbb{R}^{k \times d}$ are sampled independently from $N(0, 1)$. Then,

$$\Pr((1 - \epsilon)\|x\|^2 \leq \|\frac{1}{\sqrt{k}} A x\|^2 \leq (1 + \epsilon)\|x\|^2) \geq 1 - 2e^{-(\epsilon^2 - \epsilon^3)k/4}$$
Nonlinear dimensionality reduction

Find a low dimensional representation that respects “local distances” in the higher dimensional space

Many methods:
- Kernel PCA
- ISOMAP
- Local linear embedding
- Maximum volume unfolding
- Non-metric multidimensional scaling
- Laplacian
- Neural network auto encoder
- …

Due to lack of agreed upon metrics, it is very hard to judge which is best. Also, results from 3 to 2 dims is probably not representative of 1000 to 2 dimensions.

Zhang et al 2010
Other matrix factorizations

\[ X = U S V^T \]

Singular value decomposition

\[ U^T U = I, \quad V^T V = I, \quad S = \text{drag}(s) \]
\[ U \in \mathbb{R}^{n \times q}, \quad V \in \mathbb{R}^{m \times q}, \quad s \in \mathbb{R}_+^q \]
\[ X \approx U_q S_q V_q^T \]

Nonnegative matrix factorization (NMF)

\[ W \in \mathbb{R}_+^{n \times q} \text{ with } W1 = 1 \]
\[ B \in \mathbb{R}_+^{q \times n} \text{ with } B1 = 1 \]
\[ X \approx WBX \]
Clustering

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 16, 2016
Clustering images

Set of Images

[Goldberger et al.]
Clustering web search results
Hierarchical Clustering

Pick one:
- Bottom up: start with every point as a cluster and merge
- Top down: start with a single cluster containing all points and split

Different rules for splitting/merging, no "right answer"

Gives apparently interpretable tree representation. However, warning: even random data with no structure will produce a tree that "appears" to be structured.
Some Data
K-means

1. Ask user how many clusters they’d like. 
   *(e.g. \( k=5 \))
K-means

1. Ask user how many clusters they’d like. (e.g. $k=5$)

2. Randomly guess $k$ cluster Center locations
K-means

1. Ask user how many clusters they’d like.  
   *(e.g. $k=5$)*

2. Randomly guess $k$ cluster Center locations

3. Each datapoint finds out which Center it’s closest to. (Thus each Center “owns” a set of datapoints)
K-means

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to.
4. Each Center finds the centroid of the points it owns
K-means

1. Ask user how many clusters they’d like. *(e.g. $k=5$)*
2. Randomly guess $k$ cluster Center locations
3. Each datapoint finds out which Center it’s closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!
K-means

- Randomly initialize \( k \) centers
  - \( \mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)} \)

- **Classify**: Assign each point \( j \in \{1, \ldots, N\} \) to nearest center:
  - \( C^{(t)}(j) \leftarrow \arg \min_i \|\mu_i - x_j\|^2 \)

- **Recenter**: \( \mu_i \) becomes centroid of its point:
  - \( \mu_i^{(t+1)} \leftarrow \arg \min_\mu \sum_{j:C(j) = i} \|\mu - x_j\|^2 \)
  - Equivalent to \( \mu_i \leftarrow \text{average of its points!} \)
What is K-means optimizing?

- Potential function $F(\mu, C)$ of centers $\mu$ and point allocations $C$:
  \[ F(\mu, C) = \sum_{j=1}^{N} ||\mu_{C(j)} - x_j||^2 \]

- Optimal K-means:
  \[ \min_{\mu} \min_{C} F(\mu, C) \]
Does K-means converge??? Part 1

- Optimize potential function:

\[
\min_{\mu} \min_C F(\mu, C) = \min_{\mu} \min_C \sum_{i=1}^k \sum_{j : C(j) = i} ||\mu_i - x_j||^2
\]

- Fix \( \mu \), optimize \( C \)
Does K-means converge??? Part 2

- Optimize potential function:

\[
\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j: C(j) = i} \|\mu_i - x_j\|^2
\]

- Fix C, optimize \(\mu\)
Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2×2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel.
Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2×2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel.
Vector Quantization, Fisher Vectors

**Vector Quantization** (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

Similar reduced representation can be used as a feature vector

*Coates, Ng, Learning Feature Representations with K-means, 2012*
Spectral Clustering

Adjacency matrix: $W$

$W_{i,j} = \text{weight of edge (}i, j\text{)}$

$D_{i,i} = \sum_{j=1}^{n} W_{i,j}$ \quad $L = D - W$

Given feature vectors, could construct:
- $k$-nearest neighbor graph with weights in $\{0,1\}$
- Weighted graph with arbitrary *similarities* $W_{i,j} = e^{-\gamma \|x_i - x_j\|^2}$

Let $f \in \mathbb{R}^n$ be a function over the nodes

$$f^T L f = \sum_{i=1}^{N} g_i f_i^2 - \sum_{i=1}^{N} \sum_{i'=1}^{N} f_i f_{i'} w_{ii'}$$

$$= \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} w_{ii'} (f_i - f_{i'})^2.$$
Spectral Clustering

Adjacency matrix: \( W \)

\[ W_{i,j} = \text{weight of edge } (i, j) \]

\[ D_{i,i} = \sum_{j=1}^{n} W_{i,j} \]

\[ L = D - W \]

Given feature vectors, could construct:
- \((k=10)\)-nearest neighbor graph with weights in \( \{0, 1\} \)

Popular to use the Laplacian \( L \) or its normalized form \( \tilde{L} = I - D^{-1}W \) as a regularizer for learning over graphs.