Types of learning

- Supervised learning: labels known for each data sample
- Unsupervised learning: only features known; no labels!
- Weakly supervised learning: features paired with noisy labels
- Semi-supervised learning: features paired with partial (incomplete) labels
- …
Unsupervised learning

• Clustering
• Dimensionality reduction
What is clustering?

• Clustering is the process of grouping data points into “clusters”
• High intra-cluster similarity
• Low inter-cluster similarity
What is clustering?

- Clustering is the process of grouping data points into “clusters”
- High intra-cluster similarity
- Low inter-cluster similarity
Unsupervised learning

- Given: unlabeled data $S = \{x_i\}_{i=1}^N$
  - Only input features
  - No labels
- Goal: find hidden structure/patterns
  - e.g. hidden structure is a clustering of data
  - Generative model of data $P(x)$
    - Discussed further in guest lecture
- Low dimensional summary of the data
Why is clustering useful?

• Clustering is a “summary” of the data
  • Can just inspect cluster centers
  • Or inspect a few data points per cluster

• Compact preprocessing of data before supervised training
Centroid-based \((K\text{-means})\) clustering
Centroid-based \((K\text{-means})\) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
Centroid-based \((K\text{-means})\) clustering
Centroid-based \((K\text{-means})\) clustering
Centroid-based \((K\text{-means})\) clustering
Centroid-based ($K$-means) clustering
Centroid-based \((K\text{-means})\) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
Centroid-based ($K$-means) clustering
**K-means objective**

- Given \( S = \{ x_i \}_{i=1}^{N} \), find

\[
\text{argmin}_{S=C_1 \cup \cdots \cup C_k, \{c_1, \ldots, c_k\}} \sum_{k} \sum_{x \in C_k} \| x - c_k \|_2^2
\]

\[
\text{argmin}_{S=C_1 \cup \cdots \cup C_k} \sum_{k} | C_k | \text{var}(C_k)
\]
EM algorithm for $K$-means

- Expectation/maximization (EM) algorithm
- E step:
  - Estimate clusters $C_k$
  - Estimate cluster membership
- M step:
  - Estimate cluster centers $c_k$
  - Estimate model parameters

\[
\arg\min_{S=C_1 \cup \ldots \cup C_k, \{c_1, \ldots, c_k\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2
\]

\[
S = \{x_i\}_{i=1}^N
\]
E step

- For each $x$,

  - Assign to cluster $C_k$ with smallest distance to $c_k$

$$\text{argmin}_{S = C_1 \cup \ldots \cup C_k, \{c_1, \ldots, c_k\}} \sum_k \sum_{x \in C_k} \| x - c_k \|^2$$

$$S = \{x_i\}_{i=1}^{N}$$
M step

- For each $c_k$,
  - Compute $c_k = \text{mean}(C_k)$

\[
\text{argmin}_{S=C_1 \cup \ldots \cup C_k, \{c_1, \ldots, c_k\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2
\]

\[
S = \{x_i\}_{i=1}^N
\]
More advanced clustering algorithms

• DBSCAN: density-based clustering with some advantages over $K$-means
  - Does not require one to specify the number of clusters in the data a priori
  - Can find arbitrarily-shaped clusters
  - Has a notion of noise, and is robust to outliers
  - Just two parameters and mostly insensitive to the ordering of the data points

Recap: *K*-means

- Centroid-based clustering
  - Defines clusters using a notion of centrality
  - e.g. all items in the cluster must be close to each other
- Solve using EM algorithm
  - Also probabilistic variant (Gaussian mixture models)
- Useful when centrality assumption is good
- Other (density-based) variants include DBSCAN
Limitations of clustering
Principal component analysis
Principal component analysis

New Feature Representation!
Summarizing data

• Summarize data using smaller number of attributes
• Clustering: summarize data via clusters
  • $K$-means: summarize via cluster membership
• PCA: summarize via orthogonal projections
  • Define new feature representation
  • Rotation + projection
Orthogonal matrices

- A matrix $U$ is orthogonal if $U^T U = UU^T = I$
  - For any column $u$: $u^T u = 1$
  - For any two columns $u, u'$: $u^T u' = 0$
  - $U$ is a rotation matrix, $U^T$ is the inverse rotation
  - If $x' = U^T x$, then $x = U x'$

**Diagram:**

- $x$ and $x'$ in the original and transformed spaces, respectively.
- PCA finds a specific orthogonal $U$.
Orthogonal matrix properties

- Orthogonal transformation: $x' = U^T x$
- Norm preserving:
  
  $$(x')^T x' = x^T x$$
- Preserves total variance:

  $$\sum_{d=1}^{D} \sum_{i=1}^{N} (x_i^{(d)})^2 = \sum_{d=1}^{D} \sum_{i=1}^{N} (x_i'^{(d)})^2$$

  assuming zero mean

PCA finds a specific orthogonal $U$
Principal component analysis
Principal component analysis

Summarize Using 1 Feature?
Principal component analysis

Summarize Using 1 Feature?

Works with arbitrary subsets of columns of orthogonal $U$

E.g., $U' = [u_1, u_5, u_{20}]$
PCA definition

• Define $X$ as the matrix of all data

$$X = [x_1, \ldots, x_N] \in \mathbb{R}^{D \times N}$$

• Mean-centered matrix

$$\bar{X} = X - [\bar{x}_1, \ldots, \bar{x}_N]$$

• PCA

$$\bar{X}\bar{X}^\top = U \Lambda U^\top$$

- Symmetric
- Orthogonal
- Diagonal
PCA properties

• Assuming zero mean,

\[ XX^\top = U\Lambda U^\top \]

• Each column of \( U \) if an eigenvector of \( XX^\top \)

• Each \( \lambda_i \) is an eigenvalue

  • \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D \)

\[ \Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_D \end{bmatrix} \]

\[ (XX^T)u_d = \lambda_d u_d \]
Interpretation

- Feature covariance matrix

$$\Sigma = XX^T = U\Lambda U^T$$

- $\Sigma_{dd'}$ is the covariance of features $d$ and $d'$ in the training data

- The first column $u_1$ is the single direction of greatest variance
  - $\lambda_1$ is the total variance along $u_1$

$$\lambda_1 = \sum_{i=1}^{N} (u_1^T x_i)^2 = \sum_{i=1}^{N} (x_i^{(1)})^2$$
Interpretation (continued)

- The first column $u_1$ is the single direction that minimized the squared loss of reconstructing the original $x$ features.
- It minimizes the amount of residual variation.
- One can prove that:

\[
    u_1 = \arg\min_{u: u^T u = 1} \sum_{i=1}^{N} \|x_i - uu^T x_i\|^2
\]
Find the $u_1$ that minimizes the residual squared norm:

$$\min \| x - u_1 u_1^T x \|^2$$
Solving PCA

- Given $X = [x_1, \ldots, x_N] \in \mathbb{R}^{D \times N}$ (assuming zero mean)
- Initialize $X_1 = X$
- For $d = 1, \ldots, D$
  - Solve:
    $$u_d = \arg\min_{u: u^T u = 1} \|X_d - uu^T X_d\|^2$$
  - Update:
    $$X_{d+1} = X_d - uu^T X_d$$
Dimensionality reduction with PCA

• PCA: $XX^\top = U \Lambda U^\top$

• The first $K$ columns of $U$ are guaranteed to be the $K$-dimensional subspace that captures the most variability of $X$

• Use first $K$ columns of $U$ to create a $K$-dimensional representation:

$$x' = U_{1:K}^\top x$$

• This creates a compact summary of the original dataset

  • e.g. $K = 50$, $D = 1,000,000$
Backpropagation to implement PCA?

- Make the output the same as the input with a central bottleneck

- Activations of the hidden units form a “code”

- If the hidden and output layers are linear, hidden units are a linear function of the data and minimize the squared reconstruction error
  
  • This is exactly what PCA does!

- The $K$ hidden units will span the same space as the first $K$ components found by PCA
  
  • Their weight vectors may not be orthogonal
  
  • They will tend to have equal variances
Autoencoders

• Feedforward neural networks with same input and output shapes

• Goal: reconstruct original input by minimizing mean-squared error loss (or similar)
Autoencoders vs. PCA

- Autoencoders can learn a nonlinear lower-dimensional manifold, while PCA attempts to discover a lower-dimensional hyperplane.
Undercomplete vs. overcomplete

• Usually autoencoders are undercomplete (i.e. bottleneck layer that has some compression)

• Overcomplete ones are also possible typically with some regularization (such as L1 or sparsity conditions)
Example: fashion MNIST

- Autoencoder in Keras:

```python
m = Sequential()
m.add(Dense(1000, activation='relu',
            input_shape=(784,)))
m.add(Dense(500, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(32, activation='relu'))
m.add(Dense(2, activation='linear',
            name="bottleneck"))
m.add(Dense(32, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(500, activation='relu'))
m.add(Dense(1000, activation='relu'))
m.add(Dense(784, activation='sigmoid'))
```

Example: fashion MNIST

• Autoencoder in Keras:

```python
m = Sequential()
m.add(Dense(1000, activation='relu',
           input_shape=(784,)))
m.add(Dense(500, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(32, activation='relu'))
m.add(Dense(2, activation='linear',
           name="bottleneck"))
m.add(Dense(32, activation='relu'))
m.add(Dense(250, activation='relu'))
m.add(Dense(500, activation='relu'))
m.add(Dense(1000, activation='relu'))
m.add(Dense(784, activation='sigmoid'))
```

https://rittikghosh.com/autoencoder.html
Application: outlier (anomaly) detection

• Autoencoders compress data and then uncompress it

• Autoencoder is trained on background (and has good reconstruction performance)

• If \( x \) is far from \( \text{AE}(x) = \text{Decoder}(\text{Encoder}(x)) \), then \( x \) has low \( p_{\text{bkgd}}(x) \)

• Directly use “reconstruction loss” \( L(x, \text{AE}(x)) \) as an anomaly score
Next time

- Weakly supervised (QWoLa)
- Variational autoencoders