Unsupervised Learning

The training dataset consists only of \( \langle x_n \rangle_{n=1}^N \).

There might, or might not, be a test set with correct classes \( y \). The simplest kind of unsupervised learning: cluster into \( K \) groups.
Unsupervised Learning

The training dataset consists only of \( \langle x_n \rangle_{n=1}^N \).

There might, or might not, be a test set with correct classes \( y \).
Unsupervised Learning

The training dataset consists only of $\langle x_n \rangle_{n=1}^N$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.
$K$-Means: An Iterative Clustering Algorithm

(Review from last week.)
The stars are cluster centers, randomly assigned at first.
Assign each example to its nearest cluster center.
Recalculate cluster centers to reflect their respective examples.
Assign each example to its nearest cluster center.
Recalculate cluster centers to reflect their respective examples.
Assign each example to its nearest cluster center.
Recalculate cluster centers to reflect their respective examples.
At this point, nothing will change; we have converged.
**K-Means Clustering**

**Data:** unlabeled data $D = \langle x_n \rangle_{n=1}^{N}$, number of clusters $K$

**Result:** cluster assignment $z_n$ for each $x_n$

initialize each $\mu_k$ to a random location, for $k \in \{1, \ldots, K\}$;

do
  for $n \in \{1, \ldots, N\}$ do
    # assign each data point to its nearest cluster-center let
    $z_n = \text{argmin}_k \| \mu_k - x_n \|_2$;
  end
  for $k \in \{1, \ldots, K\}$ do
    # recenter each cluster
    let $X_k = \{x_n \mid z_n = k\}$;
    let $\mu_k = \text{mean}(X_k)$;
  end
  while any $z_n$ changes from previous iteration;

return $\{z_n\}_{n=1}^{N}$;

**Algorithm 1: K-MEANS**
Questions about \textit{K-Means}

1. Does it converge?
Questions about $K$-Means

1. Does it converge?
   Yes.
Questions about $K$-Means

1. Does it converge?
   Yes.
   
   **Proof sketch:** The $z_n$ (cluster assignments) and the $\mu_k$ (cluster centers) can only take finitely many values: $z_n \in \{1, \ldots, K\}$ and $\mu_k$ must be a mean of a subset of the data. Each time we update any of them, we will never increase this function:

   $$L(z_1, \ldots, z_N, \mu_1, \ldots, \mu_K) = \sum_{n=1}^{N} \|x_n - \mu_{z_n}\|_2^2 \geq 0$$

   $L$ is known as the **objective** of $K$-Means clustering.
   See ? section 15.1 for more details.
Questions about $K$-Means

1. Does it converge?
   Yes.
Questions about $K$-Means

1. Does it converge?
   Yes.

2. Does the solution depend on the random initialization of the means $\mu_*$?
Questions about $K$-Means

1. Does it converge?
   Yes.

2. Does the solution depend on the random initialization of the means $\mu_\ast$?
   Yes.
A Heuristic for Initializing $K$-Means

**Data:** unlabeled data $D = \langle x_n \rangle_{n=1}^N$, number of clusters $K$

**Result:** initial points $\langle \mu_1, \ldots, \mu_K \rangle$

pick $n$ uniformly at random from $\{1, \ldots, N\}$ and let $\mu_1 = x_n$;

for $k \in \{2, \ldots, K\}$ do

\# find the example that is furthest from all previously selected means

let $n = \arg\max_{n \in \{1, \ldots, N\}} \left( \min_{k' \in \{1, \ldots, k-1\}} \|x_n - \mu_{k'}\|^2 \right)$;

let $\mu_k = x_n$;

end

return $\langle \mu_1, \ldots, \mu_K \rangle$;

**Algorithm 2:** FurthestFirst
FurthestFirst in action
FurthestFirst in action – still a good idea?
Randomized Tweak on *FurthestFirst*

**Data:** unlabeled data $D = \langle x_n \rangle_{n=1}^N$, number of clusters $K$

**Result:** initial points $\langle \mu_1, \ldots, \mu_K \rangle$

pick $n$ uniformly at random from $\{1, \ldots, N\}$ and let $\mu_1 = x_n$;

for $k \in \{2, \ldots, K\}$ do

for all $n \in \{1, \ldots, N\}$, let $d[n] = \min_{k' \in \{1, \ldots, k-1\}} \|x_n - \mu_{k'}\|_2^2$ # compute distances ;

let $p = \frac{1}{\sum_{n=1}^N d[n]} d$ # normalize distances into a probability distribution;

let $n$ be a random sample from $p$;

let $\mu_k = x_n$;

end

return $\langle \mu_1, \ldots, \mu_K \rangle$;
**$K$-Means++**

Using the randomized version of **FurthestFirst** to initialize $K$-Means clustering is known as $K$-Means++. 

Approximation guarantee: let $L^*_K$ be the lowest value possible for $L(z_1, \ldots, z_N, \mu_1, \ldots, \mu_K)$, and let $\hat{L}_K$ be the value we obtain after running $K$-Means++ with $K$ clusters.

$$\mathbb{E}[\hat{L}_K] \leq 8(\log K + 2)L^*_K$$
Choosing $K$ (Hyperparameter Tuning)

Imagine testing values of $K$ from $K_{\text{min}}$ up to $K_{\text{max}}$.

In general, we expect $\hat{L}_{K+1} < \hat{L}_K$; that is, increasing $K$ should always lead to a lower $K$-means objective. Eventually, we'll see diminishing returns: as $K$ goes up, the reduction in $\hat{L}$ will be smaller and smaller.

Two ways to choose, both corresponding to "penalties" for having more clusters:

▶ Bayes information criterion (BIC):

$$K^* = \arg\min_K \hat{L}_K + K \log d$$

▶ Akaike information criterion (AIC):

$$K^* = \arg\min_K \hat{L}_K + 2Kd$$

where $x_n \in \mathbb{R}^d$. 
Choosing $K$ (Hyperparameter Tuning)

Imagine testing values of $K$ from $K_{\text{min}}$ up to $K_{\text{max}}$.

In general, we expect $\hat{L}_{K+1} < \hat{L}_K$; that is, increasing $K$ should always lead to a lower $K$-means objective.
Choosing $K$ (Hyperparameter Tuning)

Imagine testing values of $K$ from $K_{\text{min}}$ up to $K_{\text{max}}$.

In general, we expect $\hat{L}_{K+1} < \hat{L}_K$; that is, increasing $K$ should always lead to a lower $K$-means objective.

Eventually, we'll see diminishing returns: as $K$ goes up, the reduction in $\hat{L}$ will be smaller and smaller.
Choosing $K$ (Hyperparameter Tuning)

Imagine testing values of $K$ from $K_{\text{min}}$ up to $K_{\text{max}}$.

In general, we expect $\hat{L}_{K+1} < \hat{L}_K$; that is, increasing $K$ should always lead to a lower $K$-means objective.

Eventually, we'll see diminishing returns: as $K$ goes up, the reduction in $\hat{L}$ will be smaller and smaller.

Two ways to choose, both corresponding to “penalties” for having more clusters:

- Bayes information criterion (BIC): $K^* = \arg\min_K \hat{L}_K + K \log d$

- Akaike information criterion (AIC): $K^* = \arg\min_K \hat{L}_K + 2Kd$

where $x_n \in \mathbb{R}^d$. 

Recap: Unsupervised Learning

The training dataset consists only of $\langle x_n \rangle_{n=1}^N$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.

Second kind of unsupervised learning: dimensionality reduction.
▶ Useful for visualization.
▶ Also fight the curse of dimensionality.
Recap: Unsupervised Learning

The training dataset consists only of $\langle x_n \rangle_{n=1}^N$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.

Second kind of unsupervised learning: dimensionality reduction.
Recap: Unsupervised Learning

The training dataset consists only of $\langle x_n \rangle_{n=1}^{N}$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.

Second kind of unsupervised learning: dimensionality reduction.
- Useful for visualization.
Recap: Unsupervised Learning

The training dataset consists only of $\langle x_n \rangle_{n=1}^N$.

There might, or might not, be a test set with correct classes $y$.

Simplest kind of unsupervised learning: cluster into $K$ groups.

Second kind of unsupervised learning: dimensionality reduction.
  ▶ Useful for visualization.
  ▶ Also fight the curse of dimensionality.
Linear Dimensionality Reduction
Linear Dimensionality Reduction

As before, you only have a training dataset consisting of $\langle x_n \rangle_{n=1}^N$. Is there a way to represent each $x_n \in \mathbb{R}^d$ as a lower-dimensional vector? (Why would we want to do this?)
Linear Dimensionality Reduction

As before, you only have a training dataset consisting of \( \langle x_n \rangle_{n=1}^N \).

Is there a way to represent each \( x_n \in \mathbb{R}^d \) as a lower-dimensional vector?
Linear Dimensionality Reduction

As before, you only have a training dataset consisting of $\langle x_n \rangle_{n=1}^{N}$.

Is there a way to represent each $x_n \in \mathbb{R}^d$ as a lower-dimensional vector?

(Why would we want to do this?)
Assume that the data are \textit{centered},
i.e., that
\[
\text{mean } \left( \langle x_n \rangle_{n=1}^N \right) = 0.
\]
Dimension of Greatest Variance

Assume that the data are centered, i.e., that $\text{mean} \left( \left\langle x_n \right\rangle_{n=1}^N \right) = 0$. 
Projection into One Dimension

Let $\mathbf{u}$ be the dimension of greatest variance, and (without loss of generality) let $\|\mathbf{u}\|_2^2 = 1$.

$p_n = \mathbf{x}_n \cdot \mathbf{u}$ is the projection of the $n$th example onto $\mathbf{u}$.

(This should remind you a little bit of the perceptron's activation, $\mathbf{w} \cdot \mathbf{x}_n + b$.)

Since the mean of the data is 0, the mean of $\langle p_1, \ldots, p_N \rangle$ is also 0.

This implies that the variance of $\langle p_1, \ldots, p_N \rangle$ is $1/N \sum_{n=1}^{N} p_n^2$.

The $\mathbf{u}$ that gives the greatest variance, then, is:

$$\arg\max_{\mathbf{u}} N \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2$$
Projection into One Dimension

Let \( \mathbf{u} \) be the dimension of greatest variance, and (without loss of generality) let \( \| \mathbf{u} \|_2^2 = 1 \).

\[ p_n = \mathbf{x}_n \cdot \mathbf{u} \] is the projection of the \( n \)th example onto \( \mathbf{u} \).

(This should remind you a little bit of the perceptron’s activation, \( \mathbf{w} \cdot \mathbf{x}_n + b \).)
Projection into One Dimension

Let \( \mathbf{u} \) be the dimension of greatest variance, and (without loss of generality) let \( ||\mathbf{u}||^2_2 = 1 \).

\( p_n = \mathbf{x}_n \cdot \mathbf{u} \) is the projection of the \( n \)th example onto \( \mathbf{u} \).

(This should remind you a little bit of the perceptron’s activation, \( \mathbf{w} \cdot \mathbf{x}_n + b \).)

Since the mean of the data is 0, the mean of \( \langle p_1, \ldots, p_N \rangle \) is also 0.
Projection into One Dimension

Let $\mathbf{u}$ be the dimension of greatest variance, and (without loss of generality) let $\|\mathbf{u}\|_2^2 = 1$.

$p_n = \mathbf{x}_n \cdot \mathbf{u}$ is the projection of the $n$th example onto $\mathbf{u}$. (This should remind you a little bit of the perceptron’s activation, $\mathbf{w} \cdot \mathbf{x}_n + b$.)

Since the mean of the data is 0, the mean of $\langle p_1, \ldots, p_N \rangle$ is also 0.

This implies that the variance of $\langle p_1, \ldots, p_N \rangle$ is $\frac{1}{N} \sum_{n=1}^{N} p_n^2$. 
Projection into One Dimension

Let \( \mathbf{u} \) be the dimension of greatest variance, and (without loss of generality) let \( \| \mathbf{u} \|_2^2 = 1 \).

\( p_n = \mathbf{x}_n \cdot \mathbf{u} \) is the projection of the \( n \)th example onto \( \mathbf{u} \).
(This should remind you a little bit of the perceptron’s activation, \( \mathbf{w} \cdot \mathbf{x}_n + b \).)

Since the mean of the data is 0, the mean of \( \langle p_1, \ldots, p_N \rangle \) is also 0.

This implies that the variance of \( \langle p_1, \ldots, p_N \rangle \) is \( \frac{1}{N} \sum_{n=1}^{N} p_n^2 \).

The \( \mathbf{u} \) that gives the greatest variance, then, is:

\[
\arg\max_{\mathbf{u}} \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2
\]