Machine Learning (CSE 446): Unsupervised Learning

Swabha Swayamdipta & Noah Smith

University of Washington
nasmith@cs.washington.edu

October 13, 2017

Unsupervised Learning

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

Unsupervised Learning

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

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

Unsupervised Learning

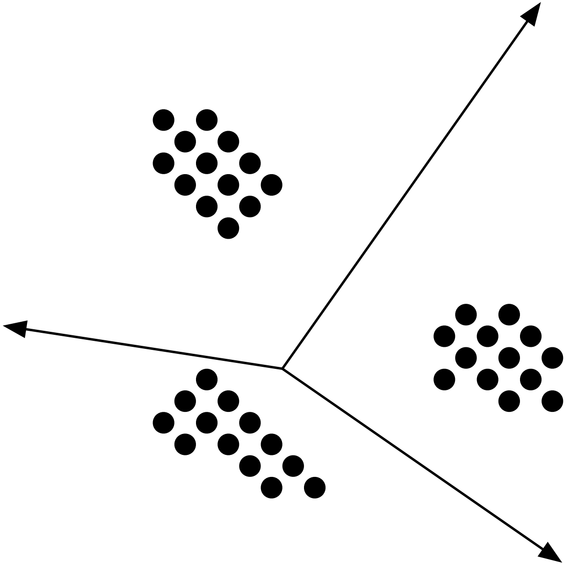
The training dataset consists only of $\langle \mathbf{x}_n \rangle_{n=1}^{N}$.

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

Simplest kind of unsupervised learning: cluster into K groups.

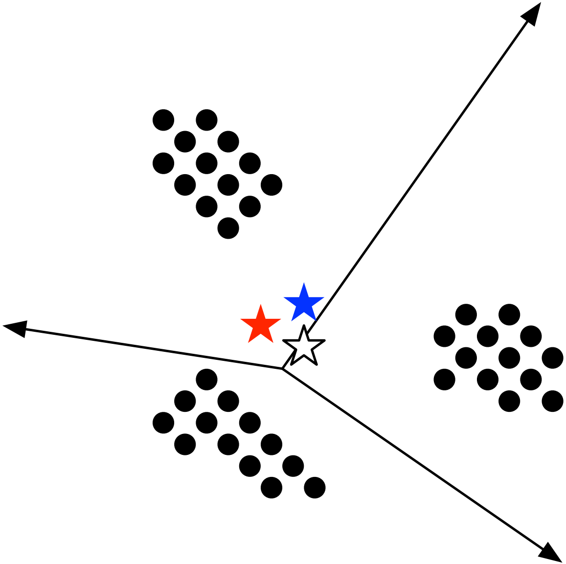
K-Means: An Iterative Clustering Algorithm

(Review from last week.)



K-Means: An Iterative Clustering Algorithm

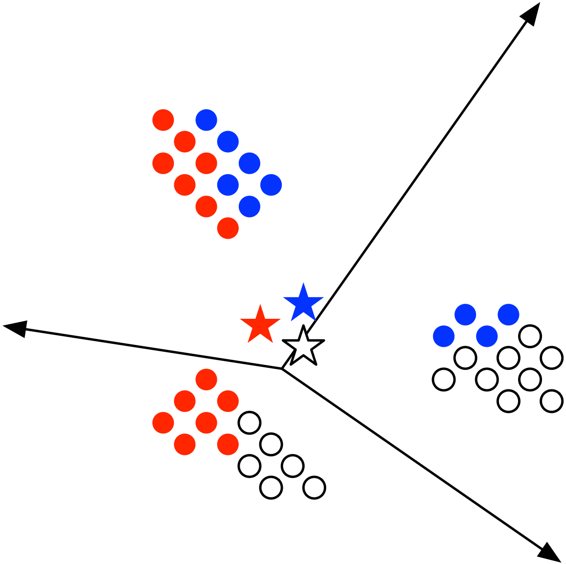
(Review from last week.)



The stars are **cluster centers**, randomly assigned at first.

K-Means: An Iterative Clustering Algorithm

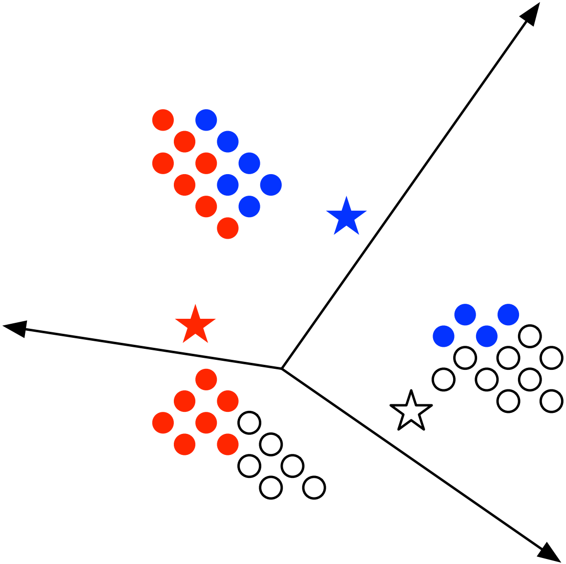
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

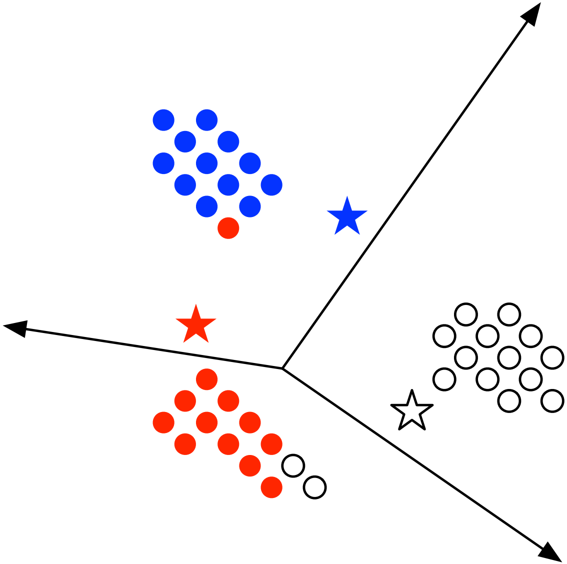
(Review from last week.)



Recalculate cluster centers to reflect their respective examples.

K-Means: An Iterative Clustering Algorithm

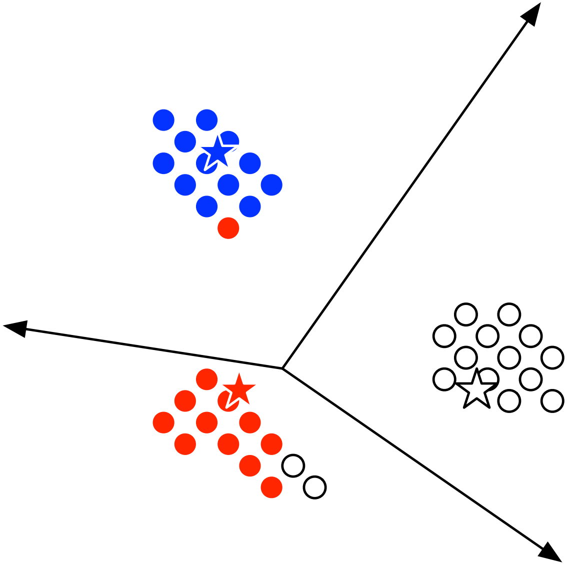
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

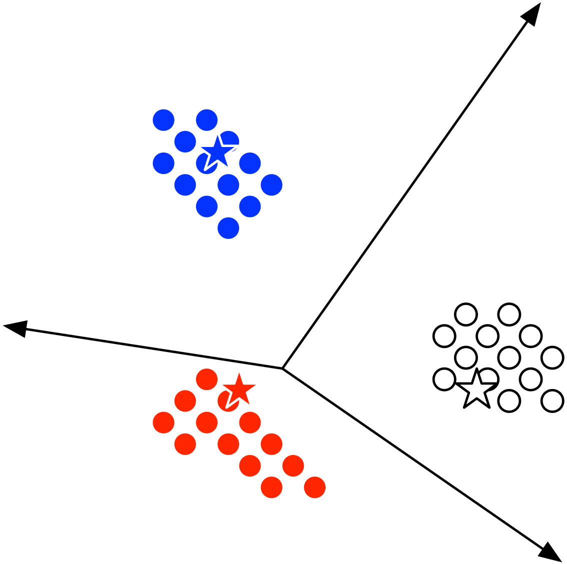
(Review from last week.)



Recalculate cluster centers to reflect their respective examples.

K-Means: An Iterative Clustering Algorithm

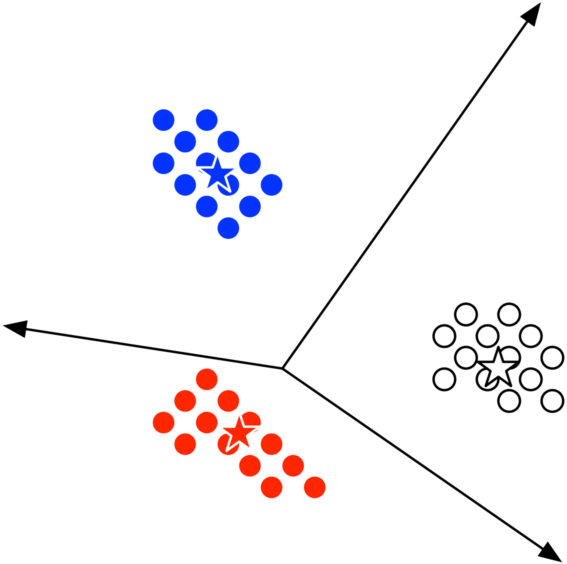
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

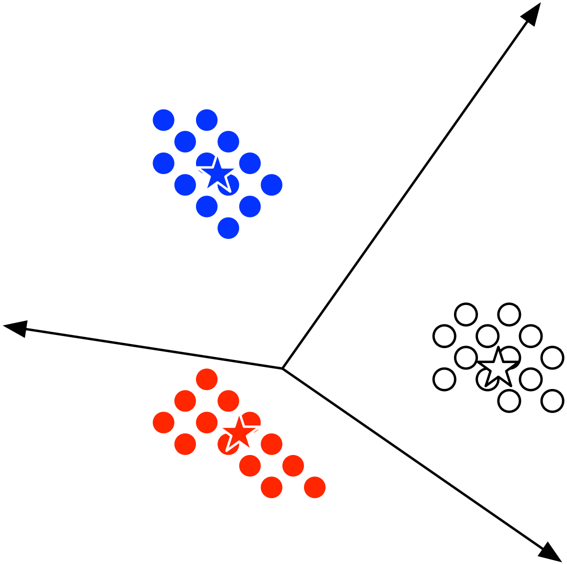
(Review from last week.)



Recalculate cluster centers to reflect their respective examples.

K-Means: An Iterative Clustering Algorithm

(Review from last week.)



At this point, nothing will change; we have converged.

K-Means Clustering

Data: unlabeled data $D = \langle \mathbf{x}_n \rangle_{n=1}^{N}$, number of clusters K
Result: cluster assignment z_n for each $\mathbf{x}_n$
initialize each $\boldsymbol{\mu}_k$ 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
        $z_n = \operatorname{argmin}_k \|\boldsymbol{\mu}_k - \mathbf{x}_n\|_2$;
    end
    for $k \in \{1, \ldots, K\}$ **do**
        # recenter each cluster
        let $\boldsymbol{X}_k = \{\mathbf{x}_n \mid z_n = k\}$;
        let $\boldsymbol{\mu}_k = \text{mean}(\boldsymbol{X}_k)$;
    end
while *any* z_n *changes from previous iteration*;
return $\{z_n\}_{n=1}^{N}$;

Algorithm 1: K-Means

Questions about 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 $\boldsymbol{\mu}_k$ (cluster centers) can only take finitely many values: $z_n \in \{1, \ldots, K\}$ and $\boldsymbol{\mu}_k$ must be a mean of a subset of the data. Each time we update any of them, we will never increase this function:

$$L(z_1, \ldots, z_N, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K) = \sum_{n=1}^{N} \left\| \mathbf{x}_n - \boldsymbol{\mu}_{z_n} \right\|_2^2 \geq 0$$

   L is known as the **objective** of K-Means clustering.
   See Daume (2017) 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 $\boldsymbol{\mu}_*$?

Questions about K-Means

1. Does it converge?
   Yes.

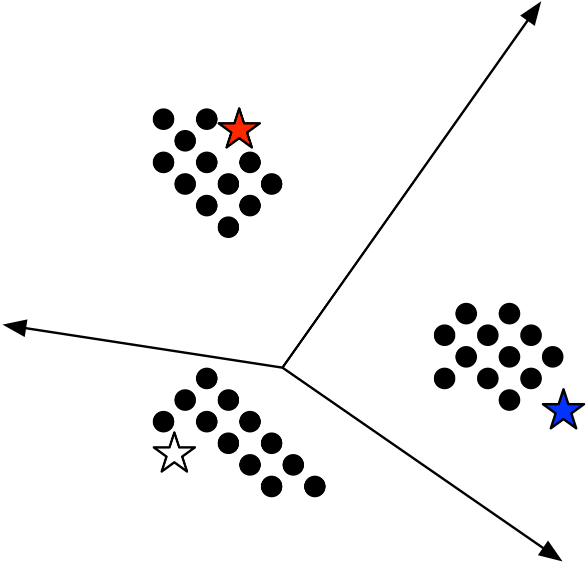
2. Does the solution depend on the random initialization of the means $\boldsymbol{\mu}_*$?
   Yes.

A Heuristic for Initializing K-Means

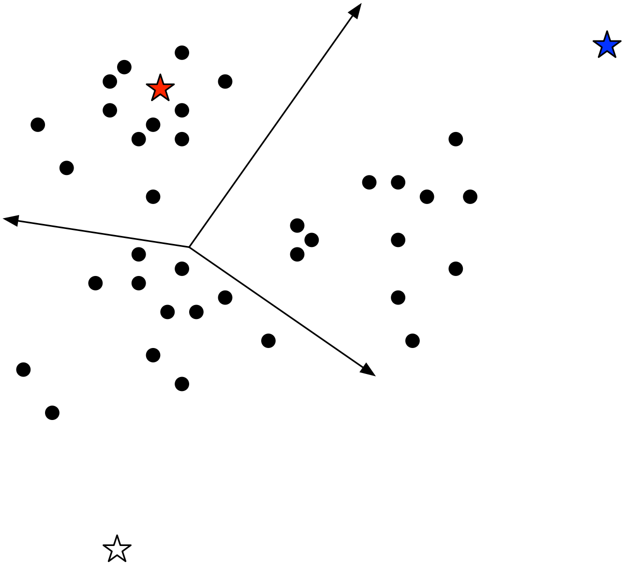
Data: unlabeled data $D = \langle \mathbf{x}_n \rangle_{n=1}^{N}$, number of clusters K
Result: initial points $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \rangle$
pick n uniformly at random from $\{1, \ldots, N\}$ and let $\boldsymbol{\mu}_1 = \mathbf{x}_n$;
for $k \in \{2, \ldots, K\}$ **do**
  # find the example that is furthest from all previously selected means
  let $n = \operatorname*{argmax}_{n \in \{1,\ldots,N\}} \left(\min_{k' \in \{1,\ldots,k-1\}} \|\mathbf{x}_n - \boldsymbol{\mu}_{k'}\|_2^2 \right)$;
  let $\boldsymbol{\mu}_k = \mathbf{x}_n$;
end
return $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \rangle$;

Algorithm 2: FurthestFirst

FurthestFirst in action



FurthestFirst in action – still a good idea?



Randomized Tweak on FurthestFirst

Data: unlabeled data $D = \langle \mathbf{x}_n \rangle_{n=1}^{N}$, number of clusters K
Result: initial points $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \rangle$
pick n uniformly at random from $\{1, \ldots, N\}$ and let $\boldsymbol{\mu}_1 = \mathbf{x}_n$;
for $k \in \{2, \ldots, K\}$ **do**
    for all $n \in \{1, \ldots, N\}$, let $\mathbf{d}[n] = \min_{k' \in \{1, \ldots, k-1\}} \|\mathbf{x}_n - \boldsymbol{\mu}_{k'}\|_2^2$ # compute distances ;
    let $\mathbf{p} = \frac{1}{\sum_{n=1}^{N} \mathbf{d}[n]} \mathbf{d}$ # normalize distances into a probability distribution;
    let n be a random sample from $\mathbf{p}$;
    let $\boldsymbol{\mu}_k = \mathbf{x}_n$;
end
return $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \rangle$;

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 $L(z_1, \ldots, z_N, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$, 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_{min} up to K_{max}.

Choosing K (Hyperparameter Tuning)

Imagine testing values of K from K_{min} up to K_{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_{min} up to K_{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_{min} up to K_{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): $K^* = \operatorname*{argmin}_K \hat{L}_K + K \log d$
- Akaike information criterion (AIC): $K^* = \operatorname*{argmin}_K \hat{L}_K + 2Kd$

where $\mathbf{x}_n \in \mathbb{R}^d$.

Recap: Unsupervised Learning

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

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

Simplest kind of unsupervised learning: cluster into K groups.

Recap: Unsupervised Learning

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

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 $\langle \mathbf{x}_n \rangle_{n=1}^{N}$.

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 $\langle \mathbf{x}_n \rangle_{n=1}^{N}$.

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 $\langle \mathbf{x}_n \rangle_{n=1}^{N}$.

Linear Dimensionality Reduction

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

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

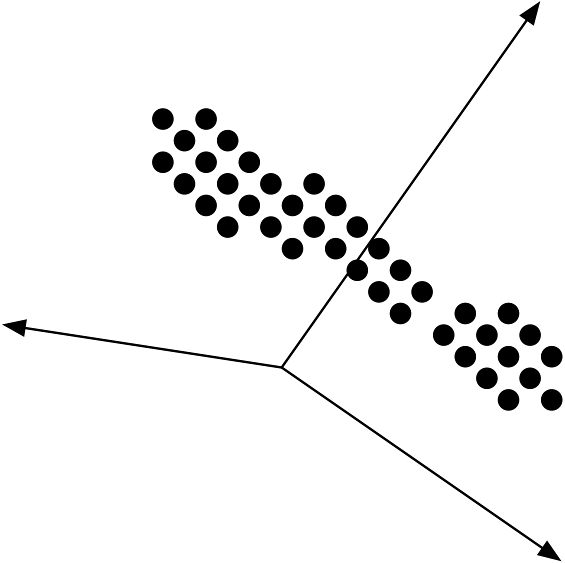
Linear Dimensionality Reduction

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

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

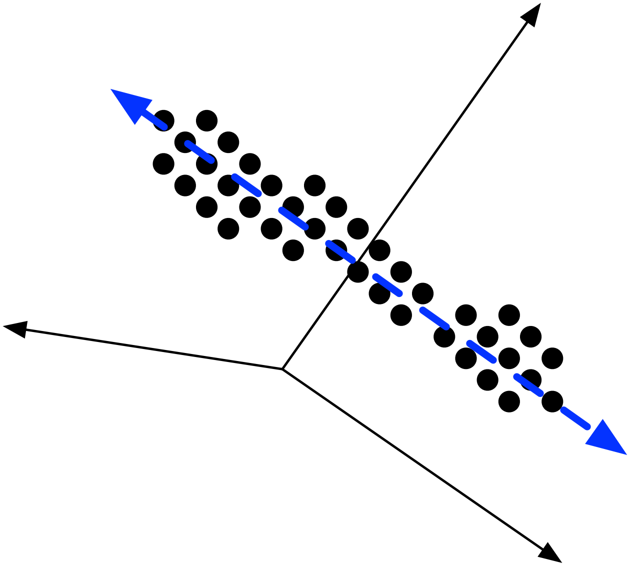
(Why would we want to do this?)

Dimension of Greatest Variance



Assume that the data are *centered*,
i.e., that
mean $\left(\langle \mathbf{x}_n \rangle_{n=1}^{N}\right) = \mathbf{0}$.

Dimension of Greatest Variance



Assume that the data are *centered*,
i.e., that
mean $\left(\langle \mathbf{x}_n \rangle_{n=1}^{N}\right) = \mathbf{0}$.

Projection into One Dimension

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

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 nth 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 nth 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 $\mathbf{0}$, 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 nth 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 $\mathbf{0}$, 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 nth 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 $\mathbf{0}$, 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:

$$\underset{\mathbf{u}}{\operatorname{argmax}} \sum_{n=1}^{N} (\mathbf{x}_n \cdot \mathbf{u})^2$$

References I

Hal Daume. *A Course in Machine Learning (v0.9)*. Self-published at http://ciml.info/, 2017.