Machine Learning (CSE 446): Unsupervised Learning

Sham M Kakade

University of Washington
cse446-staff@cs.washington.edu

Announcements

- HW2 posted. Due Feb 1.
  - It is long. Start this week!
- Today:
  Review: the perceptron algo New: Unsupervised learning

Review

Neuron-Inspired Classifier

$$f(\mathbf{x}) = \mathsf{sign}\left(\mathbf{w} \cdot \mathbf{x} + b\right)$$

$$\text{remembering that: } \mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{d} \mathbf{w}[j] \cdot \mathbf{x}[j]$$

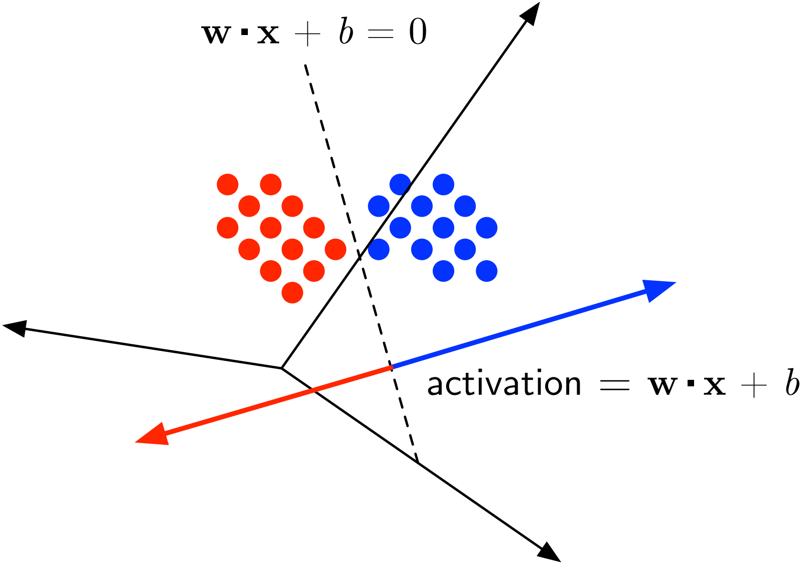
Learning requires us to set the weights $\mathbf{w}$ and the bias b.
Scalings: Note that assuming $\|x\| \leq 1$ doesn't change anything. Even with this scaling, the scale of $\|w\|$ is arbitrary.

Perceptron Learning Algorithm

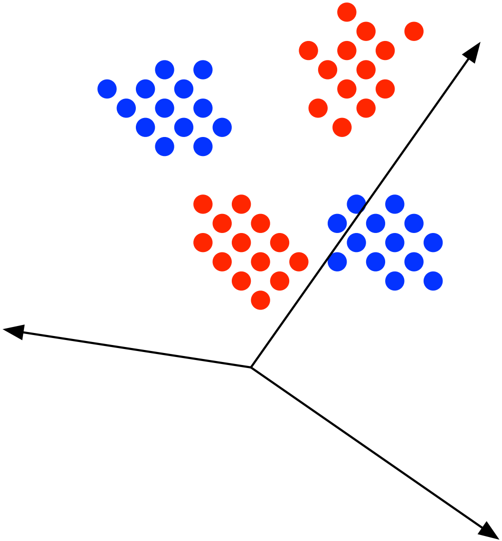
Data: $D = \langle(\mathbf{x}_n, y_n)\rangle_{n=1}^{N}$, number of epochs E
Result: weights $\mathbf{w}$ and bias b
initialize: $\mathbf{w} = \mathbf{0}$ and $b = 0$;
for $e \in \{1, \ldots, E\}$ **do**
  for $n \in \{1, \ldots, N\}$, *in random order* **do**
    # predict
    $\hat{y} = \text{sign}\left(\mathbf{w} \cdot \mathbf{x}_n + b\right)$;
    if $\hat{y} \neq y_n$ **then**
      # update
      $\mathbf{w} \leftarrow \mathbf{w} + y_n \cdot \mathbf{x}_n$;
      $b \leftarrow b + y_n$;
    end
  end
end
return $\mathbf{w}, b$

Algorithm 1: PerceptronTrain

Linear Decision Boundary



When does the perceptron not converge?



Linear Separability

A dataset $D = \langle(\mathbf{x}_n, y_n)\rangle_{n=1}^{N}$ is **linearly separable** if there exists some linear classifier (defined by $\mathbf{w}, b$) such that, for all n, $y_n = \text{sign}\left(\mathbf{w} \cdot \mathbf{x}_n + b\right)$.

If data are separable, (without loss of generality) can scale so that:

- "margin at 1", can assume for all (x, y)

$$y\left(\mathbf{w}_* \cdot \mathbf{x}\right) \geq 1$$

  (let w^* be smallest norm vector with margin 1).
- CIML: assumes $\|w^*\|$ is unit length and scales the "1" above.

Linear Separability and the Geometric Margin

Perceptron Convergence

Due to Rosenblatt (1958).

Theorem: Suppose data are scaled so that $\|\mathbf{x}_i\|_2 \leq 1$.
Assume D is linearly separable, and let be $\mathbf{w}_*$ be a separator with "margin 1".
Then the perceptron algorithm will converge in at most $\|\mathbf{w}_*\|^2$ epochs.

- Let $\mathbf{w}_t$ be the param at "iteration" t; $\mathbf{w}_0 = 0$
- "A Mistake Lemma": At iteration t

$$\begin{aligned} \text{If we do not make a mistake,} \quad \|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 &= \|\mathbf{w}_t - \mathbf{w}_*\|^2 \\ \text{If we do make a mistake,} \quad \|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 &\leq \|\mathbf{w}_t - \mathbf{w}_*\|^2 - 1 \end{aligned}$$

- The theorem directly follows from this lemma. Why?

Today

Unsupervised Learning

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

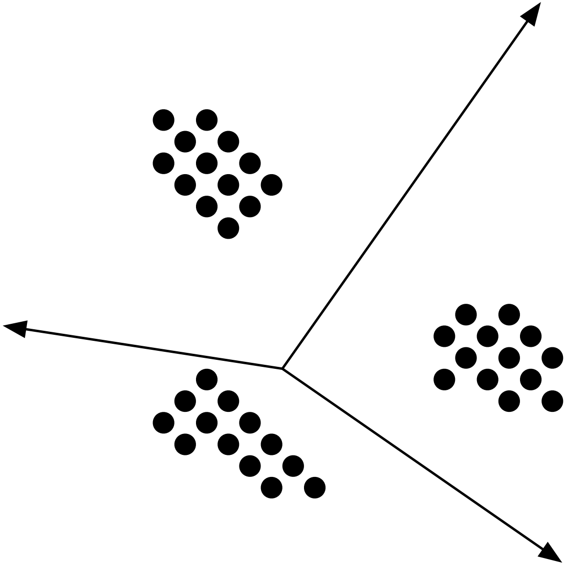
There might, or might not, be labels.

Two simple unsupervised learning methods:

- cluster into K groups.
- project your data into less dimensions
- Today: look at these methods as objective function minimization.

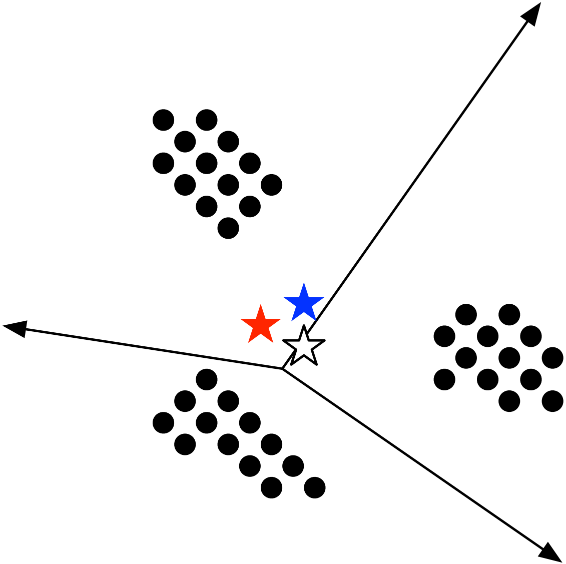
K-Means: An Iterative Clustering Algorithm

(Review from last week.)



K-Means: An Iterative Clustering Algorithm

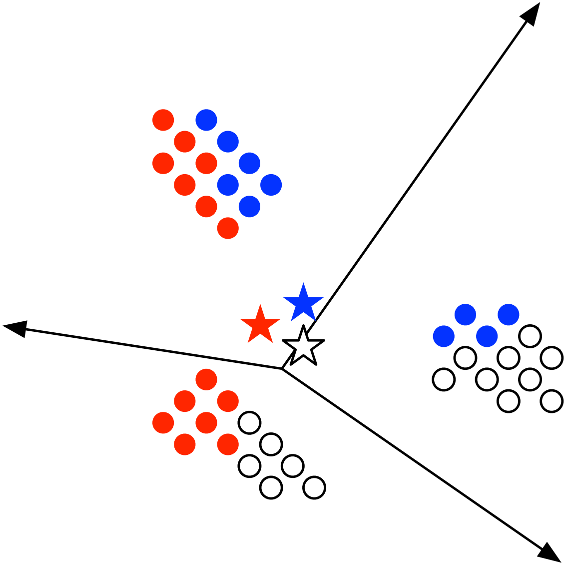
(Review from last week.)



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

K-Means: An Iterative Clustering Algorithm

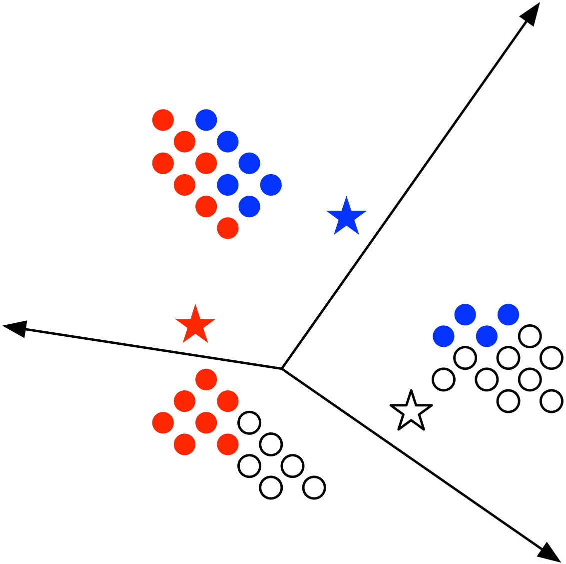
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

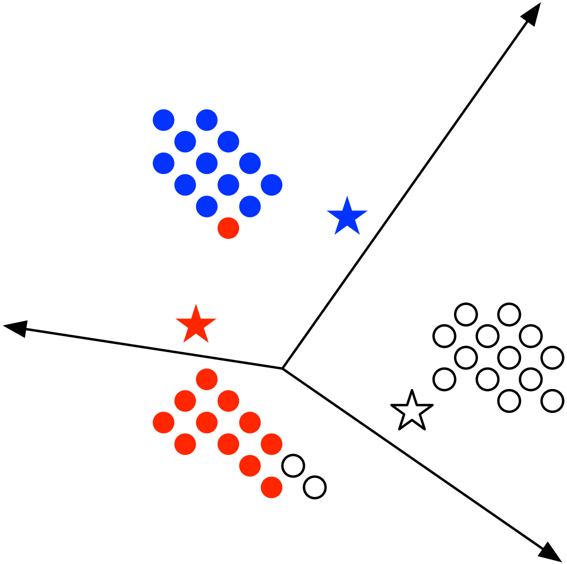
(Review from last week.)



Recalculate cluster centers to reflect their respective examples.

K-Means: An Iterative Clustering Algorithm

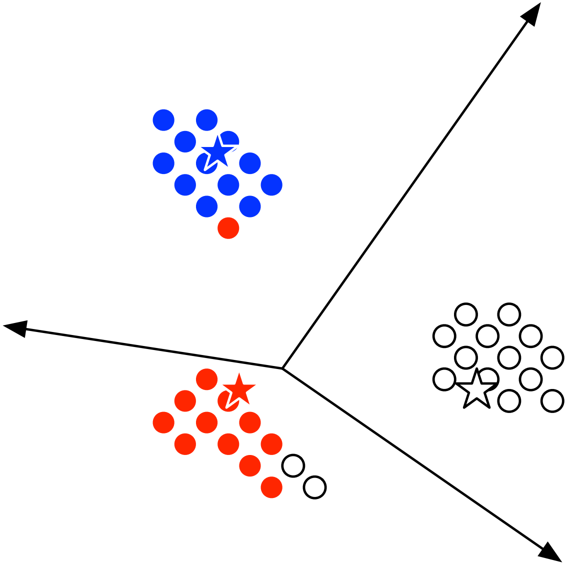
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

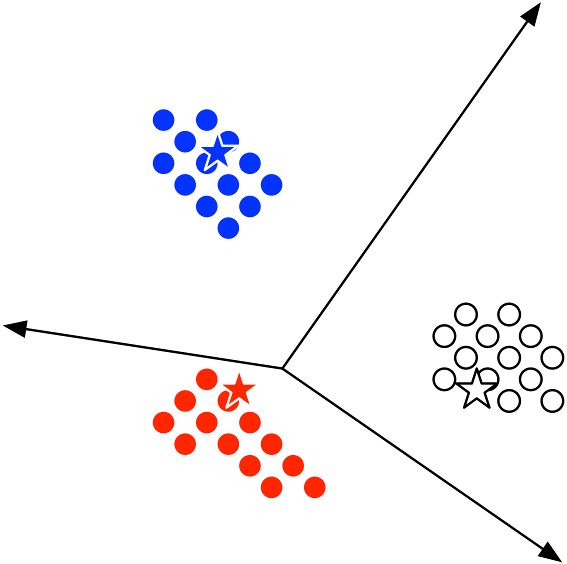
(Review from last week.)



Recalculate cluster centers to reflect their respective examples.

K-Means: An Iterative Clustering Algorithm

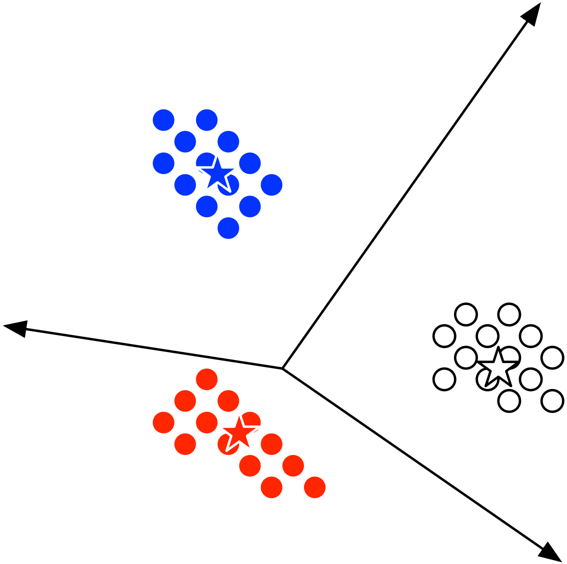
(Review from last week.)



Assign each example to its nearest cluster center.

K-Means: An Iterative Clustering Algorithm

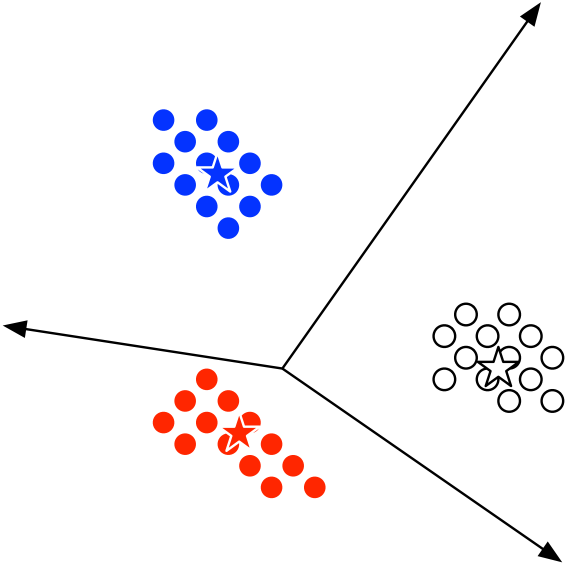
(Review from last week.)



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 2: K-MEANS

Questions about K-Means

1. Does it converge?
   Yes.
2. Does it converge to the right answer?

What would we like to do?

- **Objective function:** find k-means, $\mu_1, \ldots \mu_k$, which minimizes the following squared distance cost function:

$$\sum_{n=1}^{N} \left(\min_{k' \in \{1,\ldots,k-1\}} \|\mathbf{x}_n - \boldsymbol{\mu}_{k'}\|_2^2 \right)$$

- We can also write this objective function in terms of the assignments z_n's. How?

This is the general approach of loss function minimization: find parameters which make our objection function "small" (and which also "generalizes")

Convergence Proof Sketch

- The cluster assignments, the z_n's take only finitely many values. So the cluster centers, the $\boldsymbol{\mu}_k$'s, also must only take a finite number of values. 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 the **objective function** of K-Means clustering.
- Convergence must occur in a **finite number** of steps, due to:
  L decreases at every step; L can only take on finitely many values.
  See CIML, Chapter 15 for more details.
- Does the solution depend on the random initialization of the means $\boldsymbol{\mu}_*$?

Does K-means converge to the minimal cost solution?

- No! The objective is an NP-Hard problem, so we can't expect **any** algorithm to minimize the cost without essentially checking (near to) all assignments.
- Bad example for K-means:

Aside: Is NP-hardness a relevant concept for ML problems?

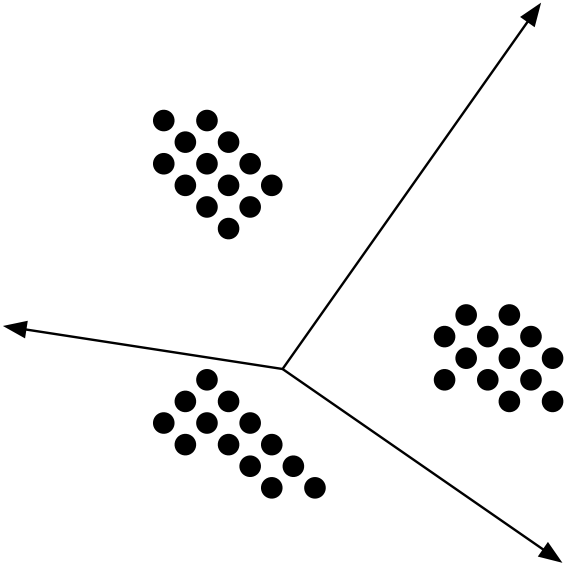
- Maybe the set of 'hard' problems may not be interesting.

A Heuristic for Initializing K-Means

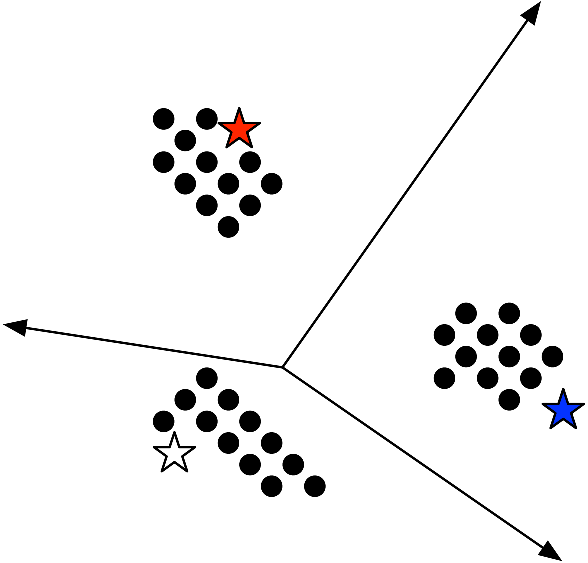
Data: unlabeled data $D = \langle \mathbf{x}_n \rangle_{n=1}^{N}$, number of clusters K
Result: initial points $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \rangle$
pick n uniformly at random from $\{1, \ldots, N\}$ and let $\boldsymbol{\mu}_1 = \mathbf{x}_n$;
for $k \in \{2, \ldots, K\}$ **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 3: FURTHESTFIRST (K-MEANS++)

FurthestFirst in action



FurthestFirst in action



Some Comments

- K-means usually converges very quickly in practice.
- K-means++ still not guaranteed to find the global optima,
  - in practice, we can get stuck.
  - often try multiple initializations (use a little randomness in K-means++ and run the algorithm multiple times).
  - it does have ("multiplicative") approximation guarantees.
- How to choose K?
  - Information theory criterion (see CIML).
  - Based on 'good' function value decrease on 'holdout' set.

See CIML.

Recap: Unsupervised Learning

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

There might, or might not, be labels.

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

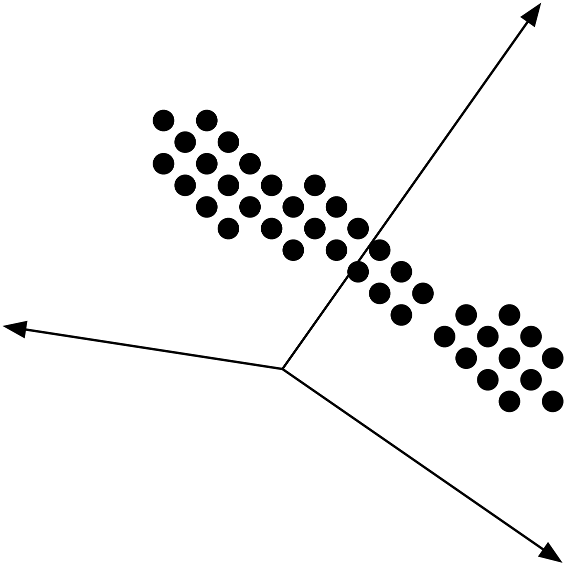
Linear Dimensionality Reduction

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

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

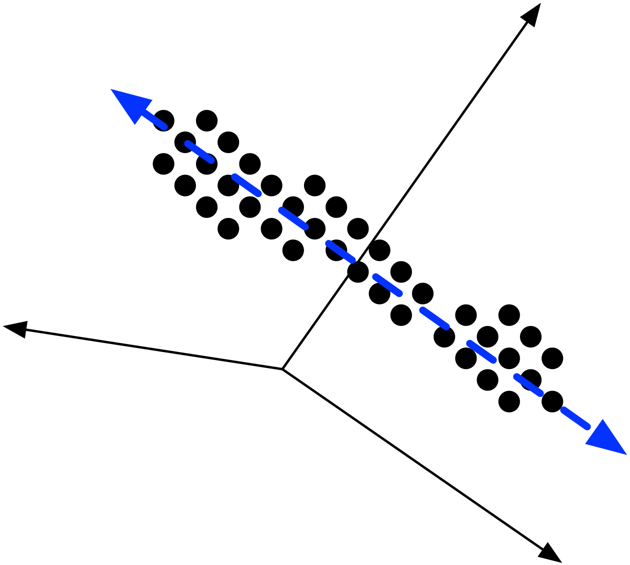
Why would we want to do this?

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 $\|\mathbf{u}\|_2^2 = 1$.

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

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 $\dfrac{1}{N}\displaystyle\sum_{n=1}^{N} p_n^2$.

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

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

Projecting x onto a vector u

Projecting x onto an 'orthonormal' basis u

References I

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.