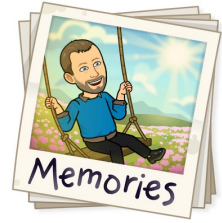


Machine Learning

Textbook: Machine Learning Refined, by Watt, Borhani, and Katsaggelos, 2nd edition:
MLRefined.com

Previous Lecture

- ♦ If $X_c = L\Sigma V^T$, then the PCA directions are the right singular vectors in V .
- ♦ Covariance eigenvalues and singular values satisfy $\lambda_j = \sigma_j^2/P$.
- ♦ The rank- d PCA reconstruction equals the truncated SVD $L_d\Sigma_d V_d^T$.
- ♦ The squared optimal reconstruction error is the sum of the squares of the omitted singular values.
- ♦ PCA can be written as a linear encoder $\mathbf{z} = V_d^T \mathbf{q}$ and decoder $\hat{\mathbf{q}} = V_d \mathbf{z}$.
- ♦ A general optimal linear autoencoder recovers the PCA principal subspace, although its internal coordinates need not equal the canonical PCA coordinates.



§8.5 - Clustering and k -Means

PCA asked:

which low-dimensional subspace
best represents the data?

Now ask a different unsupervised question:

can the observations be organized
into groups of similar points?

No labels are supplied.

The algorithm must infer the groups
from the feature vectors themselves.

This is **clustering**.

The Clustering Problem

Suppose

$$\mathbf{x}_1, \dots, \mathbf{x}_P \in \mathbb{R}^N.$$

Choose a number of clusters:

$$K \geq 2.$$

We want to assign each observation
to one of the labels

$$c_p \in \{1, \dots, K\}.$$

Unlike classification:

these labels are not known in advance.

They are created by the clustering algorithm.

The numerical names
 $1, \dots, K$
have no intrinsic meaning.

Represent Each Cluster by a Prototype

For each cluster k ,
introduce a prototype
 $\mu_k \in \mathbb{R}^N$.

In k -means,
the prototype is the cluster's
centroid.

An observation assigned to cluster k
is represented approximately by
 μ_k .

Thus:
continuous data vectors
are represented by
one of K prototype vectors.

This is a form of vector quantization.

The k -Means Objective

For assignments

$c_1, \dots, c_P$
and centroids
 $\mu_1, \dots, \mu_K$,

define
$$J = \sum_{p=1}^P \left\| \mathbf{x}_p - \mu_{c_p} \right\|^2.$$

This is the
within-cluster sum of squares.

Also called:

- within-cluster squared error;
- distortion;
- inertia in some software.

Dividing by P
does not change the minimizers.

What Is Being Optimized?

Two different kinds of variables appear.

Continuous variables:

$$\mu_1, \dots, \mu_K.$$

Discrete variables:

$$c_1, \dots, c_P.$$

So the problem is

$$\underset{c_1, \dots, c_P, \mu_1, \dots, \mu_K}{\text{minimize}} \quad J.$$

Optimizing all variables simultaneously
is difficult.

But:
if one group of variables is fixed,
the other group becomes easy to optimize.

This motivates alternating minimization.

Step 1: Fix the Centroids

Suppose

$$\mu_1, \dots, \mu_K$$

are fixed.

Observation p contributes

$$\left\| \mathbf{x}_p - \mu_{c_p} \right\|^2.$$

To minimize this term:
assign the point to its nearest centroid.

Thus

$$c_p = \underset{k=1, \dots, K}{\operatorname{argmin}} \quad \left\| \mathbf{x}_p - \mu_k \right\|^2.$$

This is the
assignment step.

Each observation can be assigned independently.

Step 2: Fix the Assignments

Now suppose

$$c_1, \dots, c_P$$

are fixed.

Let

$$S_k = \{p : c_p = k\}$$

be the observations assigned to cluster k .

Then the terms involving

μ_k
are

$$J_k(\mu_k) = \sum_{p \in S_k} \|\mathbf{x}_p - \mu_k\|^2.$$

What centroid minimizes this expression?

Derive the Centroid Update

Let

$$n_k = |S_k|.$$

Expand:

$$\begin{aligned} J_k &= \sum_{p \in S_k} [\mathbf{x}_p^T \mathbf{x}_p - 2\mu_k^T \mathbf{x}_p + \mu_k^T \mu_k]. \end{aligned}$$

Differentiate:

$$\begin{aligned} \nabla_{\mu_k} J_k &= -2 \sum_{p \in S_k} \mathbf{x}_p + 2n_k \mu_k. \end{aligned}$$

Set equal to zero:

$$\begin{aligned} n_k \mu_k &= \sum_{p \in S_k} \mathbf{x}_p. \end{aligned}$$

Therefore

$$\mu_k = \frac{1}{n_k} \sum_{p \in S_k} \mathbf{x}_p.$$

The optimal prototype under squared Euclidean distance is the arithmetic mean.

Why the Name k -Means?

There are

K
clusters.

Each cluster is represented by
the mean of its assigned observations.

Thus:

$$K \text{ clusters} + \text{cluster means} \Rightarrow k\text{-means}.$$

The mean is not an arbitrary choice.

It follows mathematically
from minimizing squared Euclidean distances.

Change the loss,

and the optimal prototype may change.

The k -Means Algorithm

Choose initial centroids.

Then repeat:

Assignment step

$$c_p \leftarrow \arg \min_k \|\mathbf{x}_p - \boldsymbol{\mu}_k\|^2.$$

Centroid step

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{n_k} \sum_{p:c_p=k} \mathbf{x}_p.$$

Stop when:

assignments no longer change;

or centroids move negligibly;

or the objective changes negligibly;

or a maximum iteration count is reached.

This alternating procedure is often called

Lloyd's algorithm.

Why the Objective Cannot Increase

During the assignment step:

centroids are fixed.

Each point is assigned to the centroid

giving the smallest possible squared distance.

Therefore

$$J_{\text{after assignment}} \leq J_{\text{before assignment}}.$$

During the centroid step:

assignments are fixed.

Each centroid is replaced by the value

minimizing its cluster's squared error.

Therefore

$$J_{\text{after centroid}} \leq J_{\text{before centroid}}.$$

So every full iteration

is nonincreasing in J .

But This Does Not Guarantee the Global Minimum

The objective is not jointly convex
in assignments and centroids.

Different initial centroids
can lead to different final clusterings.

Thus, with a fixed rule for handling exact distance ties,
k-means generally guarantees:

- monotone decrease of the objective;
- eventual arrival at a fixed assignment / centroid configuration;

but not:

- the globally best clustering.

Initialization matters.

A One-Dimensional Example

Take
 $x = (1, 2, 3, 8, 9, 10)$
and
 $K = 2$.

Initialize:
 $\mu_1 = 2, \quad \mu_2 = 8$.

Assign each observation
to its nearest centroid.

The first three points
go to cluster 1.

The last three
go to cluster 2.

So:
 $S_1 = \{1, 2, 3\},$
 $S_2 = \{8, 9, 10\}.$

Objective Before Updating the Centroids

Cluster 1:
 $(1 - 2)^2 + (2 - 2)^2 + (3 - 2)^2$
 $= 1 + 0 + 1$
 $= 2.$

Cluster 2:
 $(8 - 8)^2 + (9 - 8)^2 + (10 - 8)^2$
 $= 0 + 1 + 4$

$$= 5.$$

Therefore

$$J = 7.$$

Now update the centroids.

Update the Centroids

Cluster 1 mean:

$$\begin{aligned}\mu_1 &= \frac{1+2+3}{3} \\ &= 2.\end{aligned}$$

Cluster 2 mean:

$$\begin{aligned}\mu_2 &= \frac{8+9+10}{3} \\ &= 9.\end{aligned}$$

New objective:

cluster 1 error:

$$2.$$

Cluster 2 error:

$$\begin{aligned}(8-9)^2 + (9-9)^2 + (10-9)^2 \\ = 2.\end{aligned}$$

Thus

$$J = 4.$$

The objective decreased from

$$7$$

to

$$4.$$

Next Assignment Step

With centroids

$$\mu_1 = 2$$

and

$$\mu_2 = 9,$$

the assignment boundary lies halfway between them:

$$\begin{aligned}\frac{2+9}{2} \\ = 5.5.\end{aligned}$$

Points

$$1, 2, 3$$

remain in cluster 1.

Points
8, 9, 10
remain in cluster 2.

Assignments do not change.

The centroids remain
2
and
9.

The algorithm has converged.

Geometry: Nearest-Centroid Regions

Fix centroids

$$\mu_1, \dots, \mu_K.$$

Cluster k receives all points satisfying

$$\begin{aligned} &\|\mathbf{x} - \mu_k\|^2 \\ &\leq \|\mathbf{x} - \mu_j\|^2 \end{aligned}$$

for every

$$j \neq k.$$

These nearest-centroid regions are
Voronoi cells.

Each centroid owns the points
closer to it than to any competing centroid.

Boundary Between Two Centroids

The boundary between centroids

$$\mu_j$$

and

$$\mu_k$$

satisfies

$$\begin{aligned} &\|\mathbf{x} - \mu_j\|^2 \\ &= \|\mathbf{x} - \mu_k\|^2. \end{aligned}$$

Expand both sides:

$$\begin{aligned} &\mathbf{x}^T \mathbf{x} - 2\mu_j^T \mathbf{x} + \|\mu_j\|^2 \\ &= \mathbf{x}^T \mathbf{x} - 2\mu_k^T \mathbf{x} + \|\mu_k\|^2. \end{aligned}$$

The quadratic terms cancel.

Therefore

$$2(\mu_k - \mu_j)^T \mathbf{x} = \|\mu_k\|^2 - \|\mu_j\|^2.$$

This is a hyperplane.

The Boundary Is Perpendicular to the Line Joining the Centroids

The boundary's normal vector is

$$\mu_k - \mu_j.$$

That vector points directly
from one centroid to the other.

Therefore the boundary is perpendicular
to the segment joining the centroids.

It also passes through their midpoint.

So in two dimensions:
the boundary is the perpendicular bisector.

In higher dimensions:
the same geometry becomes a perpendicular-bisector hyperplane.

k-Means Produces Piecewise-Linear Regions

Each pair of centroids produces
a linear boundary.

A cluster's Voronoi cell
is the intersection of the corresponding half-spaces.

Therefore each *k*-means cluster region
is convex.

Important consequence:
ordinary *k*-means is naturally suited to
compact Euclidean clusters.

It is not designed to represent
arbitrary curved or interlocking cluster shapes.

Connection to Classification

Multiclass linear classification:
choose the largest class score.

k-means:
choose the nearest centroid.

But squared distance can be rewritten:

$$\begin{aligned} & -\|\mathbf{x} - \mu_k\|^2 \\ &= -\|\mathbf{x}\|^2 + 2\mu_k^T \mathbf{x} - \|\mu_k\|^2. \end{aligned}$$

The term

$$-\|\mathbf{x}\|^2$$

is common to every cluster.

Thus nearest-centroid assignment is equivalent to maximizing

$$f_k(\mathbf{x}) = 2\boldsymbol{\mu}_k^T \mathbf{x} - \|\boldsymbol{\mu}_k\|^2.$$

So fixed-centroid k -means assignment
can be written as a linear-score classifier.

A Crucial Difference From Classification

In supervised classification:
class labels are supplied.

The model learns a rule
to reproduce them.

In clustering:
labels are not supplied.

The algorithm defines groups
according to its own objective.

Therefore cluster labels should not automatically be interpreted
as real-world categories.

A cluster is:
a geometric pattern under the chosen representation and distance.

Its substantive interpretation requires additional evidence.

Initialization Matters

Because k -means can converge
to different local solutions,

we should not usually rely on
one arbitrary initialization.

Common approaches:

- choose random observations as initial centroids;

- use several random restarts;

- keep the solution with smallest final objective;

- use a spread-out initialization such as k -means++.

Multiple restarts are inexpensive insurance
against a poor initial placement.

***k*-Means++ Idea**

A common initialization strategy:

1. choose one centroid from the observations;
2. measure each point's squared distance to its nearest chosen centroid;
3. prefer points with larger squared distance when choosing the next centroid;
4. continue until K initial centroids are chosen.

The intent:
spread initial centroids across the data rather than placing several in the same dense region.

The subsequent iterations are still ordinary k -means.

What if a Cluster Becomes Empty?

The centroid formula

$$\mu_k = \frac{1}{n_k} \sum_{p \in S_k} \mathbf{x}_p$$

requires

$$n_k > 0.$$

If no observation is assigned to cluster k :

$$n_k = 0.$$

Then the mean is undefined.

Practical implementations must choose a policy:

- reinitialize that centroid;
- move it to a poorly represented point;
- or use another explicit rule.

This is an algorithmic detail that the mathematical objective alone does not resolve.

Feature Scale Matters

k -means uses Euclidean squared distance.

Suppose feature 1

is measured on a scale of thousands,

while feature 2
is measured on a scale of tenths.

Then feature 1
can dominate
 $\|\mathbf{x} - \boldsymbol{\mu}\|^2$.

As with PCA:
centering does not equalize scales.

Possible preprocessing:
standardize features.

But:
standardization changes the geometry.

It should reflect the meaning of the variables,
not be applied mechanically.

***k*-Means Is Sensitive to Outliers**

The objective squares distances.

Therefore a very distant observation
can contribute a very large amount to J .

It can pull a centroid toward itself.

This is analogous to
least-squares regression:
squared error strongly emphasizes large residuals.

Thus k -means can be sensitive to outliers.

Alternative clustering methods
may use different distances or prototypes.

Choosing the Number of Clusters

The algorithm requires
 K
in advance.

But:
increasing K
can only decrease
or leave unchanged
the optimal within-cluster sum of squares.

Why?

With an extra centroid,
we can always reproduce the old solution
and simply not improve it.

Therefore:
 $J_{K+1}^* \leq J_K^*$.

So the raw objective alone
cannot tell us the “correct” K .

The Elbow Idea

One heuristic:
fit several values of
 K .

Plot the final within-cluster sum of squares
versus
 K .

Often:
large early improvements
are followed by smaller improvements.

A visible bend or “elbow”
may suggest a useful value of
 K .

But:
not every dataset has a clear elbow.

And an elbow is not a proof
that the data contain a unique natural number of clusters.

What k -Means Tends to Prefer

Because it minimizes Euclidean squared distances to centroids,
ordinary k -means works best when clusters are roughly:

- compact;
- reasonably separated;
- comparable in scale;
- well represented by their means.

Potential difficulties:

- elongated clusters;

very unequal cluster sizes;

strongly unequal densities;

curved or nonconvex shapes;

substantial outliers.

The algorithm's geometry determines what it can represent well.

A Matrix View of Cluster Membership

Define indicator variables

$$r_{pk}$$

by

$$r_{pk} = \begin{cases} 1, & c_p = k, \\ 0, & c_p \neq k. \end{cases}.$$

Each observation belongs to exactly one cluster:

$$\sum_{k=1}^K r_{pk} = 1.$$

Then the objective can be written

$$J = \sum_{p=1}^P \sum_{k=1}^K r_{pk} \|\mathbf{x}_p - \boldsymbol{\mu}_k\|^2.$$

This form makes the alternating optimization explicit:

optimize R with centroids fixed;

optimize centroids with R fixed.

Centroid Update in Indicator Form

For cluster k :

$$n_k = \sum_{p=1}^P r_{pk}.$$

Then

$$\boldsymbol{\mu}_k = \frac{\sum_{p=1}^P r_{pk} \mathbf{x}_p}{\sum_{p=1}^P r_{pk}}.$$

This is simply the mean

of observations whose indicator equals 1.

The indicator representation

will be useful later

when we compare hard assignments

with probabilistic or soft assignments.

k -Means in Python: Data and Initial Centroids

Start with the one-dimensional example

written as two-dimensional array data:

```
import numpy as np

X = np.array([[1.],
              [2.],
              [3.],
              [8.],
              [9.],
              [10.]])

centroids = np.array([[2.],
                      [8.]])
```

Rows of

x

are observations.

Rows of

centroids

are prototypes.

Compute Distances in Python

Create one column of squared distances

for each centroid:

```
K = len(centroids)

distances = np.zeros((len(X), K))

for k in range(K):
    diff = X - centroids[k]

    distances[:, k] = np.sum(
        diff**2,
        axis=1
    )
```

New Python idea:

np

creates an array filled with zeros.

After the loop:

distances[**p**, **k**]

contains

$$\|\mathbf{x}_p - \boldsymbol{\mu}_k\|^2.$$

Assignment Step in Python

For each row,

choose the smallest distance:

```
labels = np.argmin(
    distances,
    axis=1
)
```

New Python idea:

np

returns the index of the smallest entry.

For our example:

```
print(labels)
```

gives

```
[0 0 0 1 1 1]
```

Python cluster labels begin at 0.

Mathematically:

these correspond to clusters

1

and

2.

Centroid Step in Python

Update each centroid:

```
new_centroids = centroids.copy()

for k in range(K):
    cluster_points = X[labels == k]

    if len(cluster_points) > 0:
        new_centroids[k] = np.mean(
            cluster_points,
            axis=0
        )
```

New Python idea:

```
X[labels == k]
```

uses a Boolean mask

to select only observations in cluster k .

The empty-cluster check prevents

taking the mean of an empty set.

For the example:

```
new_centroids = (2,9)T
```

in one-dimensional coordinate form.

Compute the Objective in Python

After assignments have been determined:

```
J = 0.0

for p in range(len(X)):
    k = labels[p]

    diff = X[p] - centroids[k]

    J = J + np.sum(diff**2)
```

With initial centroids

```
2
and
8:
J = 7.
```

After replacing the centroids by

```
2
and
9:
J = 4.
```

The code reproduces the hand calculation.

A Complete Small *k*-Means Loop

```
for iteration in range(100):

    distances = np.zeros((len(X), K))

    for k in range(K):
        diff = X - centroids[k]

        distances[:, k] = np.sum(
            diff**2,
            axis=1
        )

    labels = np.argmin(
        distances,
        axis=1
    )

    new_centroids = centroids.copy()

    for k in range(K):
        cluster_points = X[labels == k]

        if len(cluster_points) > 0:
            new_centroids[k] = np.mean(
                cluster_points,
                axis=0
            )

    if np.allclose(
        new_centroids,
        centroids
    ):
        break

    centroids = new_centroids
```

The code alternates
exactly the two mathematical minimization steps.

Track the Objective During Training

To verify the descent behavior:

```
history = []  
  
for iteration in range(100):  
    ...  
    history.append(J)  
    ...
```

The resulting objective sequence should be
nonincreasing,
apart from implementation or floating-point subtleties.

This is a useful diagnostic:
if an ordinary k -means implementation
produces a large objective increase
after a correct alternating step,

something is likely wrong in the code.

What Has Changed From PCA?

PCA:
represent each observation by
continuous coordinates
in a low-dimensional subspace.

k -means:
represent each observation by
a discrete cluster identity
and its centroid.

PCA minimizes:
distance to a subspace.

k -means minimizes:
distance to one of K prototype points.

Both are unsupervised.

Both are reconstruction problems.

But the representation is fundamentally different.

What did we learn?

- ◆ k -means represents each observation by one of K centroid vectors and minimizes within-cluster squared distance.
- ◆ With centroids fixed, the optimal assignment sends each observation to its nearest centroid.
- ◆ With assignments fixed, the optimal centroid is the arithmetic mean of the observations assigned to that cluster.
- ◆ Alternating the assignment and centroid steps cannot increase the k -means objective, but it need not find the global optimum.
- ◆ Fixed centroids partition space into convex Voronoi cells separated by perpendicular-bisector hyperplanes.
- ◆ Initialization, feature scaling, outliers, and the choice of K can materially change the clustering.

Next

- The behavior of regression, classification, PCA, and clustering all depends on how the input features are represented.
- Scale, units, nonlinear transformations, and redundant variables can change the geometry seen by a learning algorithm.
- Next we begin treating the feature representation itself as part of model design.
- **1.26 - Preparing Features**