9 Unsupervised Learning

Unsupervised learning works with __________ data (no ____), making evaluation more difficult.

Density Estimation

Density estimation models the probability __________ function of the (unknown) distribution from which data were drawn.

• Recall: We used a __________ model for density estimation in §7 to help with one-class classification: \( gm = \text{mixture.GaussianMixture(n_components=1)} \) estimates the parameters \( \mu \) and \( \Sigma \) of the multivariate normal distribution \( N_D(\mu_D, \Sigma_{D \times D}) \), which has density function

\[
     f_{\mu,\Sigma}(x) = \frac{\exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)}{\sqrt{(2\pi)^D |\Sigma|}}.
\]

For density estimation, use the same code to estimate \( f_{\mu,\Sigma}(x) \) given \( \{x_i\} \). (Just omit the __________ used to decide whether a new \( x \) is in the one class.)

• Recall: We used a nonparametric model in __________ regression, a supervised learning method in §7. We used \( N \) Gaussians centered at \( \{x_i\} \) to make weighted averages of \( y \)'s.

In kernel density estimation (KDE), use the __________ of the same \( N \) Gaussians to estimate the probability density function \( f(x) \) which generated the unsupervised (no \( y \)) examples \( \{x_i\} \).

Consider the 1D case. Our kernel model is

\[
     \hat{f}_b(x) = \frac{1}{Nb} \sum_{i=1}^{N} k\left(\frac{x - x_i}{b}\right),
\]

where \( b \) is a hyperparameter controlling the underfit-overfit tradeoff and \( k(x) \geq 0 \) is a kernel with \( \int_{-\infty}^{\infty} k(x)dx = 1 \).

As in §7, we use a Gaussian kernel, \( k(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) \).

We can rewrite the model as

\[
     \hat{f}_b(x) = \__________
\]

which is the __________ of the \( N \) Gaussians \( \{N(\mu = ____, \sigma = ____)|i = 1, \ldots, N\} \).²

¹ These methods improve upon just using a (density) __________.

² The notation of Burkov and Wikipedia conceals the essential point that the model is an __________ of \( N \) Gaussians centered at \( \{x_i\} \). Possibly they emphasize we are passing \( \frac{x-x_i}{b} \) to a parameterless kernel.
Python

- To estimate $N_D(\mu_D, \Sigma_{D \times D})$, use from sklearn import mixture and
  gm = mixture.GaussianMixture(n_components=1) as in §7. Then gm.fit(X), gm.means_,
gm.covariances_, and np.exp(gm.score_samples(X)) work as before. To learn more:
  
  - Reference manual:

- For KDE:
  
  - from sklearn.neighbors import KernelDensity
  - kde = KernelDensity(bandwidth=1.0, kernel='gaussian')
    
    * $b =$ bandwidth is the bandwidth
    * kernel is one of 'gaussian' (the default), 'tophat', 'epanechnikov', 'exponential',
      'linear', 'cosine'; see Examples link below for their shapes and effects
  
  - kde.fit(X) fits the model to the data.
  - kde.score_samples(X) gives log-likelihood of each $x$ in $X$, so np.exp(kde.score_samples(X))
gives $f_b(x)$.

To learn more:

  - Reference manual:
  - Example:
    https://scikit-learn.org/stable/auto_examples/neighbors/plot_kde_1d.html

---

3 Click on “launch binder” to run it online. Change “N = 100” to “N = 10” to see kernels.
Clustering

*Clustering* finds groups of unlabeled examples and assigns a cluster ID to each example. It is used in, e.g., exploratory data analysis, market segmentation, social network analysis, recommender systems, and stock sector analysis.

- **k-means clustering** maps each unlabeled example $x$ to a cluster ID.
  - Choose the number of clusters $k$.
  - Randomly choose one example to start each cluster as its centroid $c$.
  - Label each example $x$ with the centroid to which it is closest.
  - Recompute each centroid as the mean of the examples labeled with it.
  - Repeat the last two steps until centroids stabilize.

$k$ is a hyperparameter typically decided by an educated guess.

$k$-means seeks to minimize the sum of squared distances of examples to their respective centroids.

E.g. Run $k$-means with $k = 3$, starting with the three unfilled points as cluster centers.

---

4 Clustering labeled examples is not classification, as it predicts $\{y_i\}$. 
Python

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=8, n_init=10, random_state=0):
  * `n_clusters` is the number of clusters to be found
  * `n_init` is the number of times k-means is run, each with different centroid seeds
  * `random_state=0` determines centroid initialization

kmeans.fit(X) computes the clusters
kmeans.labels_ gives the labels (cluster IDs) of each x in the training X
kmeans.cluster_centers_ gives coordinates of the cluster centers
kmeans.predict(X) gives the closest cluster for each x in X
```

To learn more:

- Examples:
  - 3D: [https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html)

• **DBSCAN**[^1] is a **based clustering** algorithm that puts x in a cluster if it is to many points in that cluster.

Hyperparameters:

- \( \epsilon \) is a distance threshold (__________).
- \( n \) is the __________ number of examples in a cluster.

Definitions:

- \( x' \) is a __________ of \( x \) if its distance to \( x \) is \( \leq \epsilon \).
- \( x \) is a __________ example if its neighborhood size is at least \( n \).
- An __________ (or noisy example) has no neighbors.

Algorithm:

- For each unexamined core example:
  * Make its neighborhood a __________.
  * __________ add core-example neighbors of this cluster’s examples.
  * Add non-core example neighbors (__________ recursively) of cluster examples.
- Call remaining examples __________.

[^1]“DBSCAN” refers to “Density-based spatial clustering of applications with noise.”
DBSCAN builds clusters with an _________ shape. \((k\text{-means builds hyper-}_____\text{clusters.})\) Choosing \(\epsilon\) and \(n\) is not easy. Increasing \(n\) or decreasing \(\epsilon\) requires higher density to make a cluster. DBSCAN cannot effectively handle clusters of _________ density.

e.g. Run DBSCAN with \(\epsilon = 1\) and \(n = 2\):

Python

– from sklearn.cluster import DBSCAN
– db = DBSCAN(eps=0.5, min_samples=5, metric='euclidean')
  eps is \(\epsilon\), min_samples is \(n\), and metric options include those we used in \(k\text{-NN}\)
– db.fit(X) computes the clusters
– db.labels_ gives labels (cluster IDs) of each \(x\) in the training \(X\); noisy examples get -1
– db.core_sample_indices_ gives indices of core samples
– For each \(k \neq -1\) in db.labels_, we can find neighbors in cluster \(k\):
  is_in_cluster_k = (db.labels_ == k)
  is_core_sample = np.zeros(shape=db.labels_.shape)
  is_core_sample[db.core_sample_indices_] = True
  is_neighbor_of_cluster_k = (is_in_cluster_k & ~is_core_sample)

To learn more:


• **HDBSCAN** improves upon DBSCAN and can handle clusters of _________ density. (It drops \(\epsilon\). Details are omitted.) Try it _________.

To learn more:


There is a _________ comparison of many clustering methods at
Dimensionality Reduction

*Dimensionality reduction* maps $\mathbf{x}$ into a vector with ________ features to reduce correlation among features, reduce noise, visualize data (we see only 2D or 3D), and facilitate interpretable models.

- **Principal component analysis** (PCA) fits a new coordinate system to $\{ \mathbf{x}_i \}$ where each new ________ is called a *principal component* (PC):

  - Each PC is a ________ vector (length 1).
  - The first PC is the direction of the ________ of the data $\{ \mathbf{x}_i \}$. (It is the ________ axis of a “minimal” ellipsoid enclosing the data.)
  - For $i > 1$, the $i$th PC is orthogonal to the first $i - 1$ PCs and in the direction of the ________ greatest variance in the data.

A helpful picture is Figure 7 on p. 15 of https://www.dropbox.com/s/y9a7b0hzmksqar/Chapter9.pdf?dl=0.

To do dimensionality reduction, we choose some number $p$ of dimensions ($0 < p < D$) and ________ each $\mathbf{x}_i$ onto the first $p$ PCs, transforming the $D$-dimensional $\mathbf{x}_i$ into a smaller $p$-dimensional example. Burkov omits details.

Benefits of PCA:

- PCA does ________, while retaining most of the information, saving memory, disk space, and computation time.
- PCA can mitigate the ________ of dimensionality: as $D$ increases, the “_______” of the feature space increases faster than the available data, which become ________. Many elementary models/algorithms/insights are not designed for sparse data.
- PCA does feature ________. (It creates several important new features as linear combinations of original features. This is not feature ________.)
- The first ________ PCs often account for most of the data variability, so even high-dimensional data can be visualized in 2D or 3D.

---

6 Two vectors $\mathbf{a}$ and $\mathbf{b}$ are orthogonal if $\mathbf{a} \cdot \mathbf{b} = 0$. In 2D, orthogonal means ________.
Python

```python
from sklearn.decomposition import PCA

- pca = PCA(n_components=None, random_state=0) keeps n_components PCs; using None keeps all D components
- pca.fit(X) learns n_components PCs from \{x_i\} in X
- pca.components_ gives PCs (axes/directions of maximum variance in the data)
- pca.explained_variance_ratio_ gives % of variance explained by each PC
- pca.transform(X) applies dimensionality reduction to each x in X

To learn more:

```

Perspective

“Burkov has undertaken a very useful but impossibly hard task in reducing all of machine learning to 100 pages. He succeeds well in choosing the topics—both theory and practice—that will be useful to practitioners, and for the reader who understands that this is the ________ 100 (or actually 150) pages you will read, not the ________ , provides a solid introduction to the field.”

—Peter Norvig, Research Director at Google and author of *Artificial Intelligence: A Modern Approach*