Caught up in Neural Nets?
When and How to use Classical Machine Learning in Your Research

PICSciE / Research Computing
Christina Peters - 10 Nov 2022
What is machine learning?

A computer observes some data, builds a model based on the data, and uses the model as both a hypothesis about the world and a piece of software that can solve problems.
Why use machine learning?

Why not just program the model?

- you may not know the model
- you want the model to be flexible and adapt as you have new data
Why use **classical** machine learning?

- You want a physically interpretable model
- Your data isn’t images
- You have a simple-ish problem
Supervised Machine Learning

Computer observes input-output pairs and learns a function that maps input to output.

• Labeled data
Supervised Machine Learning

Training set of N input-output pairs:

\[(X_1, y_1), (X_2, y_2), \ldots (X_N, y_N)\]

where each pair was generated by an unknown function \( y = f(X) \).

Goal: learn a function \( h \) that approximates the true function \( f \).
Unsupervised Machine Learning

Computer learns patterns/structure in input data.

- Without labels.
Unsupervised Machine Learning

Training set of N inputs:

$X_1, X_2, \ldots, X_N$

Goal: learn structure/patterns in the data
Supervised Machine Learning

- You have simulated data where you know the truth
- You have data labeled by a human

Unsupervised Machine Learning

- You have experimental data
- You want to explore / visualize the data
## Common algorithms

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Clustering</td>
</tr>
<tr>
<td>Classification</td>
<td>Dimensionality Reduction</td>
</tr>
</tbody>
</table>
# Common algorithms

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Clustering</td>
</tr>
<tr>
<td>Classification</td>
<td>Dimensionality Reduction</td>
</tr>
</tbody>
</table>
Regression

Training set of $N$ input-output pairs:

$$(X_1,y_1), (X_2,y_2), \ldots (X_N,y_N)$$

where desired output value is a continuous value.

Goal: learn a function that approximates the true function $f$. 
Linear Regression

input-output pairs:
- $X_i$, BMI
- $y_i$, diabetes progression

Diabetes dataset: `sklearn.datasets.load_diabetes`

Goal: learn a function $h$ that approximates the true function $f$:

**Predicted diabetes progression:** $\hat{y} = h(x)$
Linear Regression

\[ y = b + mx \]

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_N
\end{bmatrix} =
\begin{bmatrix}
  1 & x_1 \\
  1 & x_2 \\
  \vdots & \vdots \\
  1 & x_N
\end{bmatrix}
\begin{bmatrix}
  b \\
  m
\end{bmatrix}
\]
Linear Regression

\[ y = X\beta \]

Ordinary Least Squares

Compute the vector \( \beta \) that minimizes:

\[ \sum_{i}^{N} (y_i - X_i \beta)^2 \]
Linear Regression

Evaluate the generalizability of the model with a test set of M input-output pairs:

$$(X_1,y_1), (X_2,y_2), \ldots (X_M,y_M)$$

Test set data is not included in the training set.
Gaussian Process Regression

input-output pairs:
- \( x_i \)
- \( y_i = \sin(x_i) + \text{Gaussian noise} \)

Goal: learn a function \( h \) that approximates the true function \( f \):

\[ \hat{y} = h(x) \]
Gaussian Process Regression

A regression method where the prediction is probabilistic

- Gaussian
- compute empirical confidence intervals

sklearn.gaussian_process.GaussianProcessRegressor

Examples of problems in your area of research where *regression* could be used?
# Common algorithms

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Clustering</td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td>Dimensionality Reduction</td>
</tr>
</tbody>
</table>
Classification

Training set of $N$ input-output pairs:

$$(X_1, y_1), (X_2, y_2), \ldots (X_N, y_N)$$

where desired output value is a discrete value.

Goal: learn a function $h$ that approximates the true function $f$. 
Classification

input-output pairs:
- \( X_i = \{ x_{i,1}, x_{i,2} \} \)
- \( y_i \), class label, either 1 or -1

Goal: learn a function \( h \) that approximates the true function \( f \):

predicted class label, \( \hat{y} = h(x_1, x_2) \)
Support Vector Classification

Calculate the line that separates the classes with the maximal margin.

Margin: area between two parallel lines that separate the two classes of data.

- 3-D : plane
- > 3-D : hyperplane

```
sklearn.svm.SVC
```
Support Vector Classification

What if the data are not linearly separable?

Minimize a function that is a sum over values for all training data:

- 0 - if on correct side of margin
- value proportional to the distance from the margin

```
sklearn.svm.SVC
```
Support Vector Classification

Evaluate the generalizability of the model with a test set of $M$ input-output pairs:

$$(X_1, y_1), (X_2, y_2), \ldots, (X_M, y_M)$$

Test set data is not included in the training set.

Accuracy = \frac{1}{M} \sum_{i=1}^{M} (\hat{y}_i = y_i)$

`sklearn.svm.SVC`
K-Nearest Neighbors

Classification is computed from a simple majority vote of the nearest neighbors of each point.

Does not construct a general model; instead stores the training data.

sklearn.neighbors.KNeighborsClassifier
K-Nearest Neighbors

Classification is computed from a simple majority vote of the nearest neighbors of each point.

Does not construct a general model; instead stores the training data.

At left, the case where $k = 5$.

$\texttt{sklearn.neighbors.KNeighborsClassifier}$
Examples of problems in your area of research where *classification* could be used?
## Common algorithms

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Clustering</td>
</tr>
<tr>
<td>Classification</td>
<td>Dimensionality Reduction</td>
</tr>
</tbody>
</table>
Clustering

Training set of N inputs:

$X_1, X_2, \ldots X_N$.

Goal: identify M clusters in the data.
Clustering

input:
• \( x_1, x_2 \)

Goal: identify \( M \) clusters in the data:

\[
predicted \text{ class label} = h(x_1, x_2)
\]
K-means Clustering

- requires the number of clusters to be specified
- separate samples in M clusters of equal variance
- choose centroids that minimize the inertia or within-cluster sum-of-squares

\[ \sum_{i=0}^{N} \min_{\mu_j \in C} (x_i - \mu_j)^2 \]
K-means Clustering

- requires the number of clusters to be specified
- separate samples in M clusters of equal variance
- choose centroids that minimize the inertia or within-cluster sum-of-squares

\[ \sum_{i=0}^{N} \min_{\mu_j \in C} (x_i - \mu_j)^2 \]
Examples of problems in your area of research where *clustering* could be used?
## Common algorithms

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Clustering</td>
</tr>
<tr>
<td>Classification</td>
<td><strong>Dimensionality Reduction</strong></td>
</tr>
</tbody>
</table>
Dimensionality Reduction

Training set of $N$ inputs pairs:

$X_1, X_2, \ldots, X_N$.

Goal: data exploration / visualization
Principal Component Analysis

- First principal component: line that minimizes the average squared perpendicular distance from the points to the line.
- Second principal component: line orthogonal to first principal component, that does the same.

- Use the principal components to perform a change of coordinate system.

sklearn.decomposition.PCA

Principal Component Analysis

`sklearn.decomposition.PCA`
Dimensionality Reduction

input:
• $x_1, x_2, \ldots, x_{64}$

Goal: visualize the data in two dimensions
\[ \hat{x}_1, \hat{x}_2 = h(x_1, x_2, \ldots, x_{64}) \]
Principal Component Analysis

The data set projected down onto the first two principal components

```
sklearn.decomposition.PCA
```
Principal Component Analysis

The data set projected down onto the first two principal components, with labels.

sklearn.decomposition.PCA
Examples of problems in your area of research where *dimensionality reduction* could be used?