1 Introduction

*Machine learning* uses data examples to predict a label or value for a new example.

**Supervised vs. Unsupervised Learning**

- In *supervised learning*, the dataset is a collection of labeled examples \( \{(x_i, y_i)\}_{i=1}^{N} \), where \( x_i = [x_1^{(i)}, \ldots, x_D^{(i)}] \) is a \( D \)-dimensional *feature vector*.

  e.g. Here is a dataset for three children of \( \{(x_i = [height, weight], y_i = age)\} = \{(x_1 = [44, 70], y_1 = 7), (x_2 = [45, 75], y_2 = 9), (x_3 = [38, 40], y_3 = 4)\} \)

  We create a model to map new examples to suitable labels, e.g.:

  - A *support vector machine* (SVM) is a binary classifier that uses a line to separate points in a plane into two groups (or it separates \( D \)-dimensional points with a \((D - 1)\)-dimensional hyperplane). e.g.[2]

  ![SVM Example](https://scikit-learn.org/stable/_images/sphx_glr_plot_ols_001.png)

  - *Linear regression* predicts a real-valued label given an unlabeled example as \( y \leftarrow f_{w,b}(x) = wx + b \) for scalar \( y \), vector \( x \), and parameter vector \( w \). e.g.

    ![Linear Regression Example](https://scikit-learn.org/stable/_images/sphx_glr_plot_ols_001.png)

---


Logistic regression models a probability in $[0, 1]$. e.g.

A decision tree is a directed acyclic graph that we use like a flowchart to make a decision. At each node, if the value of some feature $j$ is less than a threshold, the left branch is followed; otherwise the right branch is followed. e.g.

---

3 from https://scikit-learn.org/stable/_images/sphx_glr_plot_logistic_001.png
4 from https://scikit-learn.org/stable/_images/iris.svg
- *k*-nearest neighbors (*k*-NN) classification assigns a new $\mathbf{x}$ the most frequent label among its $k$ nearest neighbors. *k*-NN regression assigns $\mathbf{x}$ the average value among its $k$ nearest neighbors. e.g.

- In *unsupervised learning*, the dataset is a collection of unlabeled examples $\{\mathbf{x}_i\}_{i=1}^N$ and we infer a function on $\mathbf{x}$ to solve a problem. e.g.:

  - *Density estimation* models the probability density function of the (unknown) distribution from which data were drawn. e.g.

---

5 from [https://scikit-learn.org/stable/_images/330px-KnnClassification.svg.png](https://scikit-learn.org/stable/_images/330px-KnnClassification.svg.png) and [https://scikit-learn.org/stable/_images/sphx_glr_plot_kde_1d_001.png](https://scikit-learn.org/stable/_images/sphx_glr_plot_kde_1d_001.png)
– *Clustering* maps each unlabeled example \( x \) to a cluster ID. e.g.

– *Dimensionality reduction* maps \( x \) into a vector with fewer features to remove highly-correlated features, reduce noise, visualize data (since we can only see up to 3D), and facilitate simple interpretable models.

– *Outlier detection* quantifies how far \( x \) is from typical examples. e.g.

---


7from https://scikit-learn.org/stable/_images/sphx_glr_plot_oneclass_001.png
Support Vector Machine (SVM): The Linear Model

- A hyperplane in a $D$-dimensional space is a $(D - 1)$-dimensional space.
  e.g. A hyperplane is a point in 1D, a line in 2D, and a plane in 3D.

- SVM using a linear model finds a hyperplane decision boundary specified by $\mathbf{w}x + b = 0$ that separates label +1 examples from label −1 examples.
  (Note: $\mathbf{w}x = w^{(1)}x^{(1)} + \ldots + w^{(D)}x^{(D)}$.)

- Training learns optimal values $\mathbf{w}^*$ and $b^*$.

- The SVM labels a new $\mathbf{x}$ with $y = f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \mathbf{x} + b^*) \in \{-1, 1\}$.

- In the easiest hard margin SVM case where the two labeled subsets are linearly separable,
  training consists of minimizing Euclidean norm $||\mathbf{w}|| = \sqrt{\sum_{i=1}^{D}(w^{(i)})^2}$ subject to constraints:
  \[
  \begin{array}{ll}
  w^{(i)}x_i + b \geq 1 & \text{if } y_i = +1 \\
  w^{(i)}x_i + b \leq -1 & \text{if } y_i = -1,
  \end{array}
  \]
  (We omit the details of this constrained optimization problem.)

- The parallel hyperplanes $\mathbf{w}x + b = 1$ and $\mathbf{w}x + b = -1$ have normal vector $\mathbf{w}$. Let $\mathbf{x}_1$ be any point in the first hyperplane. The normal line through $\mathbf{x}_1$ is $\mathbf{x}_1 + t\mathbf{w}$ for $t \in \mathbb{R}$. It intersects the second hyperplane when
  \[
  \begin{align*}
  \mathbf{w}(\mathbf{x}_1 + tw) + b = -1 & \implies t = \frac{-(\mathbf{w}\mathbf{x}_1 + b) - 1}{||\mathbf{w}||^2} \\
  \mathbf{x}_2 & = \mathbf{x}_1 + \mathbf{w}
  \left(\frac{-2}{||\mathbf{w}||^2}\right) = \mathbf{x}_1 - \frac{2\mathbf{w}}{||\mathbf{w}||^2}
  \end{align*}
  \]
  The distance from $\mathbf{x}_1$ to $\mathbf{x}_2$ is
  \[
  \left|\mathbf{x}_1 - \left(\mathbf{x}_1 - \frac{2\mathbf{w}}{||\mathbf{w}||^2}\right)\right| = \frac{2}{||\mathbf{w}||^2}.
  \]

- $||\mathbf{w}||$ is in the denominator of the distance, so minimizing $||\mathbf{w}||$ maximizes the margin between +1 and −1 support vectors.

- A sample on either of the constraint/margin boundaries is called a support vector.

Coming in §3:

- An SVM can have a hyperparameter (parameter controlling learning; not trained) to penalize misclassification of outliers (positives on the negative side of the boundary or negatives on the positive side).

- An SVM can include a kernel that allows a nonlinear decision boundary.

---

8 Burkov uses $\mathbf{w}\mathbf{x} - b = 0$. I use $\mathbf{w}\mathbf{x} + b = 0$ to match scikit-learn.
9 We return to SVMs in §3 to address some harder cases.
Python

- `from sklearn import svm` loads the svm module
- `clf = svm.SVC(kernel='linear', C=1)` gives a SVM support vector classification model. (A large C, like C=1000, gives ≈ the hard-margin version above; we will explore C more in §3.)
- `clf.fit(X, y)` fits the model to $X_{N \times D}$ and $y_{N \times 1}$\[10]\n- `clf.coef_` gives $w^*$ and `clf.intercept_` gives $b^*$
- `clf.predict(X)` does classification on examples in $X$
- `clf.score(X, y)` gives the average accuracy on $X$ with respect to $y$

To learn more:


\[10]\text{In the code, $X$ is 2D while $y$ is 1D.}