Intro to Machine Learning

What, Why, and How?

What is Machine Learning?

Machine Learning is the study of making machines learn a concept without explicitly programming it. It involves building algorithms that can learn from input data to make predictions or find patterns in the data.

Why Do We Need Machine Learning?

There are tasks that humans can distinguish but can't easily hard-code into a program.

For example, writing a set of rules to code this cat:

Sometimes, we don't even know the exact task we want to solve and just want to discover patterns in the data.

Real-World Applications of Machine Learning

- **Recommendation Systems:** Netflix, Amazon, Overstock
- **Stock Prediction:** Goldman Sachs, Morgan Stanley
- **Risk Analysis:** Credit card companies, Insurance firms
- **Face and Object Recognition:** Cameras, Facebook, Microsoft
- **Speech Recognition:** Siri, Cortana, Alexa, Dragon
- **Search Engines and Content Filtering:** Google, Yahoo, Bing

How Do We Approach Machine Learning?

Study a prediction problem in an **abstract manner** and come up with a solution which is applicable to many problems simultaneously.

Different types of **paradigms and algorithms** that have been successful in prediction tasks.

How to **systematically analyze** how good an algorithm is for a prediction task.

Types of Paradigms:

- **Supervised Learning:** e.g., decision trees, neural networks

- **Unsupervised Learning:** e.g., k-means clustering, principal component analysis
- **Reinforcement Learning:** e.g., Q-learning, policy gradients

Evaluating Algorithms:

- Use metrics like accuracy, precision, recall, F1 score, AUC-ROC.
- Techniques like cross-validation.
- Consider computational complexity, interpretability, and scalability.

Supervised Learning

- **Data:**

$$(x_1, y_1), (x_2, y_2), \ldots \in \mathrm{X} \times \mathrm{Y}\text{‘}$$

- **Assumption:** There is a (relatively simple) function:

$$f^* : \mathrm{X} \rightarrow \mathrm{Y}$$
$$f^*(x_i) = y_i$$

- **Learning Task:** Given n examples, find an approximation:

$$\hat{f} \approx f^*$$

- **Goal: It gives mostly correct predictions on unseen examples.**

Statistical Modeling Approach:

- **Labeled Training Data:** (x_i, y_i) drawn independently from a fixed underlying distribution (i.i.d assumption).
- **Learning Algorithm:** Select $\hat{f}$ from a pool of models F that maximize label agreement of the training data.
- **Selection Methods:** Maximum likelihood, maximum a posteriori, optimization of 'loss' criterion.

Classification

In classification tasks, the task is to build a function that takes in a vector of **features** $\mathbf{X}$ (also called "inputs") and predicts a **label** Y (also called the "class" or "output"). Features are things you know, and the label is what your algorithm is trying to figure out; for example, the label might be a binary variable indicating whether an animal is a cat or a dog, and the features might be the length of the animal's whiskers, the animal's weight in pounds, and a binary variable indicating whether the animal's ears stick up or are droopy. Your algorithm needs to tell dogs and cats apart (Y) using only this information about weight, whiskers, and ears ($\mathbf{X}$).

A natural way to define this function is to predict the label with the highest conditional probability: choose $g(\mathbf{X}) = \arg\max_y P(Y = y|\mathbf{X})$. To help with computing the probabilities $P(Y = y|\mathbf{X})$, you are able to use training data consisting of examples of feature–label pairs ($\mathbf{X}$, Y). You are given N of these pairs: ($\mathbf{x}^{(1)}$, $y^{(1)}$), ($\mathbf{x}^{(2)}$, $y^{(2)}$), . . ., ($\mathbf{x}^{(N)}$, $y^{(N)}$), where $\mathbf{x}^{(i)}$ is a vector of m discrete features for the ith training example and $y^{(i)}$ is the discrete label for the ith training example.

In classification, Y takes on discrete values. (The alternative is **regression**, in which your algorithm is trying to predict a continuous value.) For this class we are going to assume that all values in our training dataset are binary. While this is not a necessary assumption (both Naïve Bayes and next lecture's logistic regression algorithm can work for non-binary data), it makes it much easier to learn the core concepts. Specifically, we assume that all labels are binary ($\forall i . y^{(i)} \in \{0, 1\}$) and all features are binary ($\forall i, j . x_j^{(i)} \in \{0, 1\}$).

The classifier

- **Joint Input/Output Space:** $\mathrm{X} \times \mathrm{Y}$
- **Data Distribution:** $D(\mathrm{X} \times \mathrm{Y})$
- **Classifier:** $f : \mathrm{X} \rightarrow \mathrm{Y}$
- **Goal:** Maximize the accuracy of the classifier: $\mathrm{acc}(f) := P_{(x,y)}[f(x) = y] = E_{(x,y)}[1[f(x) = y]]$

We want a classifier f that maximizes acc:

Bayes classifier:

$$f(\vec{x}) = \arg\max_{y \in Y} P[Y = y|X = \vec{x}]$$

Other Classifiers

- $g(x) = \mathrm{X} \rightarrow \mathrm{Y}$ where Y represents the set of possible class labels, typically $\{0, 1, 2, \ldots, k - 1\}$ for k classes.

Bayes' Theorem

Bayes' theorem connects these two concepts:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

Where:

- $P(Y|X)$ is the posterior probability: This is the probability of the class Y given the observed data X. It is what we ultimately want to calculate in classification tasks, and it tells us how probable a class is given the observed features.
- $P(X|Y)$ is the likelihood: This is the probability of observing the data (features) X given a particular class Y. It's used to measure how likely the observed data is for different possible classes.
- $P(Y)$ is the prior probability of the class Y.
- $P(X)$ is the evidence (overall probability of the data)

Maximum Likelihood Estimation (MLE)

In statistics, **maximum likelihood estimation (MLE)** is a method of estimating the parameters of an assumed probability distribution, given some observed data. This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable.

$$\mathrm{L}(\theta|X) := P(X|\theta) = P(\vec{x}_1, \ldots, \vec{x}_n|\theta) = \prod_{i=1}^{n} P(\vec{x}_i|\theta) = \prod_{i=1}^{n} p_\theta(\vec{x}_i)$$

$$\hat{\theta}_{\mathrm{MLE}} = \arg\max_\theta \mathrm{L}(\theta|X) = \arg\max_\theta P(X|\theta) = \arg\max_\theta \prod_{i=1}^{n} P(\vec{x}_i|\theta) = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\vec{x}_i)$$

Case Study: Email Classification

We want to classify emails as "spam" or "not spam" based on features like the presence of certain words, email length, etc.

Data

- **Features (X):** Presence of words (e.g., "free", "win", "click"), email length, number of links, etc.
- **Labels (Y):** "spam" or "not spam"

Step 1: Bayes Classifier

The Bayes classifier aims to find the class y (spam or not spam) that maximizes the posterior probability $P(Y = y \mid X = x)$.

$$f(x) = \arg\max_{y \in \{\text{spam, not spam}\}} P(Y = y \mid X = x)$$

Using Bayes' theorem:

$$P(Y = y \mid X = x) = \frac{P(X=x \mid Y=y) \cdot P(Y=y)}{P(X=x)}$$

For classification purposes, we can ignore $P(X = x)$ because it is the same for both classes:

$$f(x) = \arg\max_{y \in \{\text{spam, not spam}\}} P(X = x \mid Y = y) \cdot P(Y = y)$$

Step 2: Estimating Probabilities Using MLE

To use the Bayes classifier, we need to estimate the following probabilities from our data:

1. **Prior Probability** $P(Y = y)$: The probability that an email is "spam" or "not spam".
2. **Likelihood** $P(X = x \mid Y = y)$: The probability of observing the features x given that the email is "spam" or "not spam".

Estimating Prior Probability $P(Y = y)$

$$P(Y = \text{spam}) = \frac{\text{Number of spam emails}}{\text{Total number of emails}}$$

$$P(Y = \text{not spam}) = \frac{\text{Number of not spam emails}}{\text{Total number of emails}}$$

Estimating Likelihood $P(X = x \mid Y = y)$ **Using MLE**

We assume that the features X (e.g., presence of words) follow a certain distribution. For simplicity, let's assume that the features are binary (presence or absence of certain words) and follow a Bernoulli distribution.

For each feature X_i:

$$P(X_i = 1 \mid Y = \text{spam}) = \theta_{i,\text{spam}}$$

$$P(X_i = 0 \mid Y = \text{spam}) = 1 - \theta_{i,\text{spam}}$$

Where $\theta_{i,\text{spam}}$ is the probability of the word i appearing in a spam email. We use MLE to estimate $\theta_{i,\text{spam}}$.

MLE for Bernoulli Distribution

For a binary feature X_i:

$$\mathrm{L}(\theta_{i,\text{spam}}) = \prod_{j=1}^{n} \theta_{i,\text{spam}}^{x_{ij}} (1 - \theta_{i,\text{spam}})^{1-x_{ij}}$$

Where x_{ij} is the value of the i-th feature for the j-th email (1 if the word is present, 0 otherwise).

The log-likelihood is:

$$\log \mathrm{L}(\theta_{i,\text{spam}}) = \sum_{j=1}^{n} [x_{ij} \log \theta_{i,\text{spam}} + (1 - x_{ij}) \log(1 - \theta_{i,\text{spam}})]$$

Taking the derivative with respect to $\theta_{i,\text{spam}}$ and setting it to zero:

$$\frac{\partial}{\partial \theta_{i,\text{spam}}} \log \mathrm{L}(\theta_{i,\text{spam}}) = \sum_{j=1}^{n} [\frac{x_{ij}}{\theta_{i,\text{spam}}} - \frac{1-x_{ij}}{1-\theta_{i,\text{spam}}}] = 0$$

Solving for $\theta_{i,\text{spam}}$:

$$\hat{\theta}_{i,\text{spam}} = \frac{\sum_{j=1}^{n} x_{ij}}{n}$$

This is the proportion of emails where the word i appears in spam emails.

Step 3: Applying Bayes Classifier with MLE Estimates

Using the estimated probabilities:

$$f(x) = \arg\max_{y \in \{\text{spam, not spam}\}} (P(X = x \mid Y = y) \cdot P(Y = y))$$

For each class y:

$$P(X = x \mid Y = y) = \prod_{i=1}^{m} \hat{\theta}_{i,y}^{x_i} (1 - \hat{\theta}_{i,y})^{1-x_i}$$

Where $\hat{\theta}_{i,y}$ is the MLE estimate of the likelihood of feature i given class y.

Example

Assume we have the following data:

- **Spam emails**: 3 out of 10
- **Features (words):** "free" (1 if present, 0 if not)

Emails:

- Spam: [1, 1, 0]
- Not spam: [0, 0, 1, 0, 0, 0, 0, 0]

Calculate:

- $P(Y = \text{spam}) = 0.3$
- $P(Y = \text{not spam}) = 0.7$

MLE for feature "free":

- $\hat{\theta}_{\text{free, spam}} = \frac{2}{3}$
- $\hat{\theta}_{\text{free, not spam}} = \frac{1}{7}$

Given a new email with "free" (X = [1]):

Calculate the posterior probabilities:

- $P(X = [1] \mid Y = \text{spam}) = \frac{2}{3}$
- $P(X = [1] \mid Y = \text{not spam}) = \frac{1}{7}$

Bayes classifier: $f([1]) = \arg\max_y P(X = [1] \mid Y = y) \cdot P(Y = y) = \arg\max(\frac{2}{3} \cdot 0.3, \frac{1}{7} \cdot 0.7) = \arg\max(0.2, 0.1) = \text{spam}$