Pipeline of Machine Learning

Chris Cornwell

Aug 26, 2025

Outline

The Pipeline

Supervised learning

Outline

The Pipeline

Supervised learning

Steps in an ML Project

Project Pipeline

0. Define the problem.
1. Collect data.
2. Design the features in the data.
3. Training of the model.
4. Test the model.

Discussion on the ML Project Pipeline

0. Define the problem.
    - Example: From a given image, determine if it is an image of a dog or not.
1. Collect data.
    - Example: Put together a large collection of images, some having dogs in them, others having a different animal, or no animal. Have a label (your output, “y”) for each image. Split into training set and test set.
2. Design the features in the data.
    - Not one thing that you always do here. Sometimes use experience/knowledge of what the data represents, sometimes use another learning algorithm to learn good features.

Discussion on the ML Project Pipeline

3. Training of the model.
    - Model is determined by set of parameters. In training, you alter the parameters iteratively – "tune" them – using optimization techniques (on the loss function).
4. Test the model.
    - Evaluate the trained model's performance on test data, measured by the same loss function.

What is Machine Learning?

Very general, academic definition by Tom Mitchell:

> *A "computer program" is said to* ***learn*** *from experience E, with respect to some task T and performance measure M if: its performance on T, as measured by M, improves with experience E.*

What is Machine Learning?

Very general, academic definition by Tom Mitchell:

> *A "computer program" is said to **learn** from experience E, with respect to some task T and performance measure M if: its performance on T, as measured by M, improves with experience E.*

- Often, the experience *E* is called "training" (updates to how program runs); based on observed data.

What is Machine Learning?

Very general, academic definition by Tom Mitchell:

> *A "computer program" is said to* ***learn*** *from experience E, with respect to some task T and performance measure M if: its performance on T, as measured by M, improves with experience E.*

- Often, the experience *E* is called "training" (updates to how program runs); based on observed data.
- "computer program," for us "learning algorithm", determines a function that produces output from given input (the data). After training, the resulting input-output function represents achieving the task *T*.

What is Machine Learning?

Very general, academic definition by Tom Mitchell:

> *A "computer program" is said to* ***learn*** *from experience E, with respect to some task T and performance measure M if: its performance on T, as measured by M, improves with experience E.*

- Often, the experience *E* is called "training" (updates to how program runs); based on observed data.
- "computer program," for us "learning algorithm", determines a function that produces output from given input (the data). After training, the resulting input-output function represents achieving the task *T*.
- In class, we will discuss algorithms made for *regression* tasks, and others for *classification* tasks, that fit this paradigm.

What is Machine Learning?

Very general, academic definition by Tom Mitchell:

> *A "computer program" is said to* ***learn*** *from experience E, with respect to some task T and performance measure M if: its performance on T, as measured by M, improves with experience E.*

- Often, the experience *E* is called "training" (updates to how program runs); based on observed data.
- "computer program," for us "learning algorithm", determines a function that produces output from given input (the data). After training, the resulting input-output function represents achieving the task *T*.
- In class, we will discuss algorithms made for *regression* tasks, and others for *classification* tasks, that fit this paradigm.
- Performance measure *M*: for us, called a *cost function* or *loss function*.

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a <u>good</u> underlying function from sample data.

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a good underlying function from sample data.

- Housing price prediction

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.
- Detect space debris, or trash on ocean surface.

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.
- Detect space debris, or trash on ocean surface.

Unsupervised learning: algorithm uses sample data, but it is unlabeled. Goal: discover something (a pattern, grouping, or some insight) about the data based on its coordinates (features).

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output “labels.” Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.
- Detect space debris, or trash on ocean surface.

Unsupervised learning: algorithm uses sample data, but it is unlabeled. Goal: discover something (a pattern, grouping, or some insight) about the data based on its coordinates (features).

- Market segmentation.

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output “labels.” Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.
- Detect space debris, or trash on ocean surface.

Unsupervised learning: algorithm uses sample data, but it is unlabeled. Goal: discover something (a pattern, grouping, or some insight) about the data based on its coordinates (features).

- Market segmentation.
- News feed (grouping similar news articles).

Two general categories in machine learning

Supervised learning: algorithm uses sample (input) data that have output "labels." Goal: determine a good underlying function from sample data.

- Housing price prediction
- Whether emails are junk or not.
- Detect space debris, or trash on ocean surface.

Unsupervised learning: algorithm uses sample data, but it is unlabeled. Goal: discover something (a pattern, grouping, or some insight) about the data based on its coordinates (features).

- Market segmentation.
- News feed (grouping similar news articles).
- Separate audio sources in a mixed signal.

Outline

The Pipeline

Supervised learning

The goal of Supervised learning

In Section 1.2, the textbook uses the term “Predictive learning” to mean same as Supervised learning.

- (Supervised $\iff$ labels)

The goal of Supervised learning

In Section 1.2, the textbook uses the term “Predictive learning” to mean same as Supervised learning.

- (Supervised $\iff$ labels)
- The labeled sample data is called **training data**. The goal is to “learn” a function from the training data that will do well labeling new data, not seen during learning process.

The goal of Supervised learning

In Section 1.2, the textbook uses the term “Predictive learning” to mean same as Supervised learning.

- (Supervised $\iff$ labels)
- The labeled sample data is called **training data**. The goal is to “learn” a function from the training data that will do well labeling new data, not seen during learning process.
- “Doing well” is measured by a loss function (M from Mitchell’s description).
- The learning algorithm starts with <u>some</u> function; it doesn’t do labeling well, but the algorithm uses the loss function to alter that function to something better. ($\longleftarrow$ “learning”)

Example Images

Example 1 - Supervised learning, Section 1.1 of textbook

Cat or Dog?

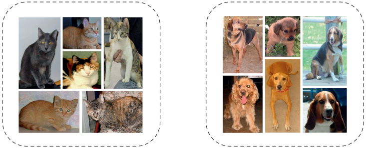

Figure 1.1 from textbook

- Training data. The images, each with label `'cat'` or `'dog'`.

Example 1 - Supervised learning, Section 1.1 of textbook

Cat or Dog?

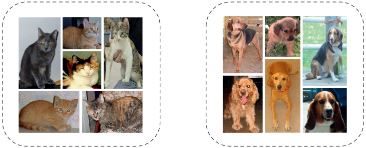

Figure 1.1 from textbook

- Training data. The images, each with label `'cat'` or `'dog'`.
- Computer sees each image, pixels in 2D array with RGB value (a vector in $\mathbb{R}^3$) at each pixel.

Example 1 - Supervised learning, Section 1.1 of textbook

Cat or Dog?

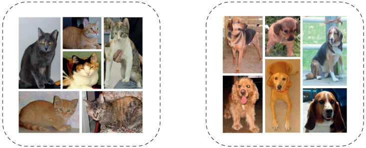

Figure 1.1 from textbook

- Training data. The images, each with label `'cat'` or `'dog'`.
- Computer sees each image, pixels in 2D array with RGB value (a vector in $\mathbb{R}^3$) at each pixel.
- **Designing features.** “Cartoon image” of ML model’s function: computes N **features** from each image $\rightsquigarrow$ vectors (points) in $\mathbb{R}^N$ $\rightsquigarrow$ points with one label separated from those with other label (by graph of linear function).

Compute N **features** from each image $\rightsquigarrow$ vectors (points) in $\mathbb{R}^N$ $\rightsquigarrow$ points with one label separated from those with other label (by graph of linear function).

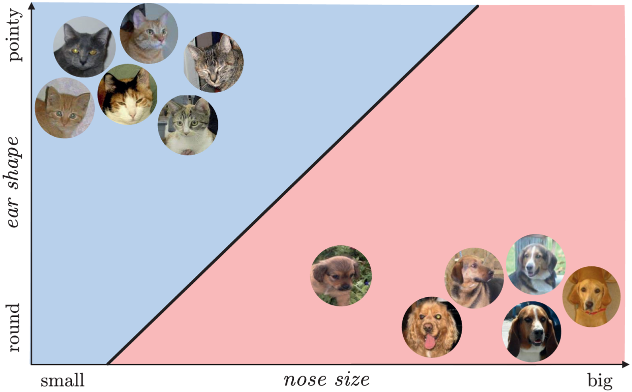

Figure 1.3 from textbook

“Doing well” once features are chosen

A way to assign positive or negative number to each, based on **decision boundary**’s side it is on; farther from 0 when distance is farther from decision boundary.

“Doing well” once features are chosen

A way to assign positive or negative number to each, based on **decision boundary**’s side it is on; farther from 0 when distance is farther from decision boundary.

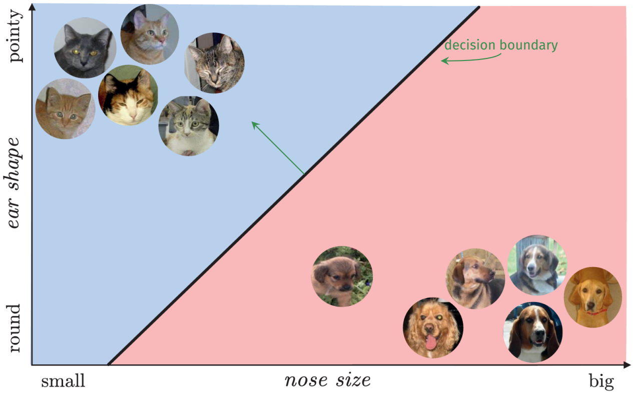

Figure 1.3 from textbook

“Doing well” once features are chosen

A way to assign positive or negative number to each, based on **decision boundary**’s side it is on; farther from 0 when distance is farther from decision boundary.

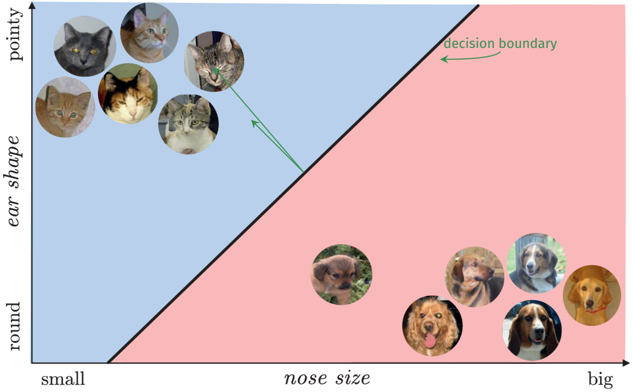

Figure 1.3 from textbook

Example 2 - Supervised learning, Section 1.2 of textbook

Predict share price

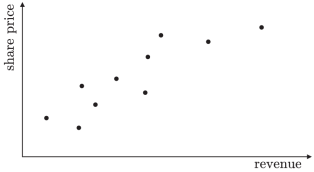

Figure 1.7, upper-left from textbook

- From training data, pick one feature: revenue. Label: the share price.

Example 2 - Supervised learning, Section 1.2 of textbook

Predict share price

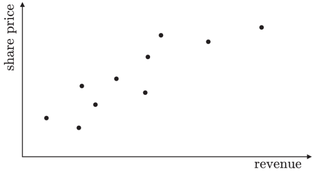

Figure 1.7, upper-left from textbook

- From training data, pick one feature: revenue. Label: the share price.
- Each revenue, a number in $\mathbb{R}$. One "independent variable," call it x.

Example 2 - Supervised learning, Section 1.2 of textbook

Predict share price

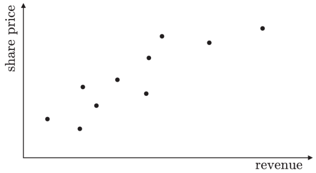

Figure 1.7, upper-left from textbook

- From training data, pick one feature: revenue. Label: the share price.
- Each revenue, a number in $\mathbb{R}$. One "independent variable," call it x.
- **Designing features.** Here, have used a feature already at hand. Not always best idea when there are multiple independent variables (we'll see examples later, where you design features).

Example 2 - Predict response variable, share price

Revenue value: x. Find a function $f(x) = \hat{m}x + \hat{b}$, so that if y is share price for x and we set $\hat{y} = f(x)$ then y and $\hat{y}$ are close, on average. (Linear Regression)

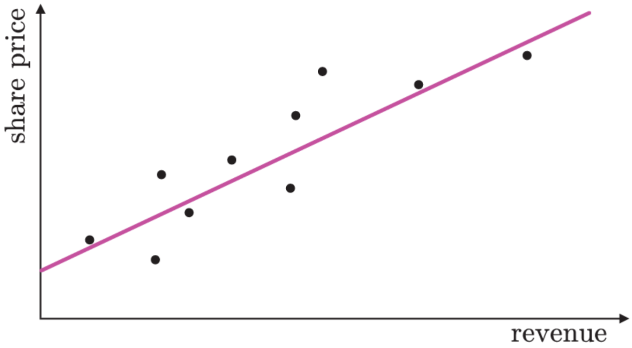

Figure 1.7, upper-right from textbook

Supervised learning, very generally

Have an "input space" (often is $\mathbb{R}^N$, for some N, or subset of it[1]). Have an output space, or label space, Y. (Y is a finite list of labels for classification; it is $\mathbb{R}$ or an interval in $\mathbb{R}$ for regression.)

[1] The domain (input space) *could* be some different space.

Supervised learning, very generally

Have an "input space" (often is $\mathbb{R}^N$, for some N, or subset of it[1]). Have an output space, or label space, Y. (Y is a finite list of labels for classification; it is $\mathbb{R}$ or an interval in $\mathbb{R}$ for regression.)

- Given a sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{P}$, with $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in Y$, drawn from an (unknown) joint probability distribution $\rho_{X,Y} : \mathbb{R}^N \times Y \longrightarrow [0, \infty)$.

[1]The domain (input space) *could* be some different space.

Supervised learning, very generally

Have an "input space" (often is $\mathbb{R}^N$, for some N, or subset of it[1]). Have an output space, or label space, Y. (Y is a finite list of labels for classification; it is $\mathbb{R}$ or an interval in $\mathbb{R}$ for regression.)

- Given a sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^P$, with $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in Y$, drawn from an (unknown) joint probability distribution $\rho_{X,Y} : \mathbb{R}^N \times Y \longrightarrow [0, \infty)$.
- Goal: to learn, from $\mathcal{S}$, a function $f^* : \mathbb{R}^N \longrightarrow Y$ that "fits" (*approximates well*) the distribution $\rho_{X,Y}$.

[1] The domain (input space) *could* be some different space.

Supervised learning, very generally

Have an "input space" (often is $\mathbb{R}^N$, for some N, or subset of it[1]). Have an output space, or label space, Y. (Y is a finite list of labels for classification; it is $\mathbb{R}$ or an interval in $\mathbb{R}$ for regression.)

- Given a sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{P}$, with $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in Y$, drawn from an (unknown) joint probability distribution $\rho_{X,Y} : \mathbb{R}^N \times Y \longrightarrow [0, \infty)$.
- Goal: to learn, from $\mathcal{S}$, a function $f^* : \mathbb{R}^N \longrightarrow Y$ that "fits" (*approximates well*) the distribution $\rho_{X,Y}$.
- Might not be possible for points on graph of f^* to be typically "close" to samples from $\rho_{X,Y}$. However, for an $\mathbf{x} \in \mathbb{R}^N$, the corresponding $f^*(\mathbf{x})$ should be near expected value of y, given $\mathbf{x}$.

[1] The domain (input space) *could* be some different space.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

[2] Sometimes called a *hypothesis class*.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

- Have space of parameters Ω; an $\omega \in \Omega$ determines a function $f_\omega : \mathbb{R}^N \longrightarrow Y$. The parameterized class is the set of all such f_ω.

[2] Sometimes called a *hypothesis class*.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

- Have space of parameters Ω; an $\omega \in \Omega$ determines a function $f_\omega : \mathbb{R}^N \longrightarrow Y$. The parameterized class is the set of all such f_ω.
- Change parameters to find a function that fits well.

[2] Sometimes called a *hypothesis class*.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

- Have space of parameters Ω; an $\omega \in \Omega$ determines a function $f_\omega : \mathbb{R}^N \longrightarrow Y$. The parameterized class is the set of all such f_ω.
- Change parameters to find a function that fits well.

How to change parameters? Select a performance measure: **(empirical) loss function** $\mathcal{L}_{\mathcal{S}} : \Omega \to \mathbb{R}$.

[2] Sometimes called a *hypothesis class*.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

- Have space of parameters Ω; an $\omega \in \Omega$ determines a function $f_\omega : \mathbb{R}^N \longrightarrow Y$. The parameterized class is the set of all such f_ω.
- Change parameters to find a function that fits well.

How to change parameters? Select a performance measure: **(empirical) loss function** $\mathcal{L}_{\mathcal{S}} : \Omega \to \mathbb{R}$.

- Want to make value of $\mathcal{L}_{\mathcal{S}}$ small; use the function itself to do this.

[2] Sometimes called a *hypothesis class*.

Achieving the general goal

Most often, we choose a *parameterized class* of functions[2], and we get f^* from that class.

- Have space of parameters Ω; an $\omega \in \Omega$ determines a function $f_\omega : \mathbb{R}^N \longrightarrow Y$. The parameterized class is the set of all such f_ω.
- Change parameters to find a function that fits well.

How to change parameters? Select a performance measure: **(empirical) loss function** $\mathcal{L}_\mathcal{S} : \Omega \to \mathbb{R}$.

- Want to make value of $\mathcal{L}_\mathcal{S}$ small; use the function itself to do this.
- Ideally, converge to some ω^*, a minimizer of $\mathcal{L}_\mathcal{S}$, and set $f^* = f_{\omega^*}$.

[2]Sometimes called a *hypothesis class*.

Questions?