Notes for Understanding

Machine Learning

A personal learning textbook

Ioanna Stamou

Contents

Introduction

Imagine you have a robot friend.

You give the robot a task:

"Look at these examples, learn the pattern, and make predictions."

This is machine learning.

Machine learning = finding patterns in data.

Example:

You give a model 1000 houses with their *price*, *size*, and *number of rooms*. The model learns the relationship between these features and the price. Then, when you give it a *new* house, it predicts the price.

That's all.

> **Key Idea**
>
> Machine learning is not magic. It is a systematic way to:
>
> 1. collect data,
> 2. find patterns,
> 3. use those patterns to make predictions or decisions.

The rest of these notes are about:

- the **statistics** you need to talk precisely about patterns,
- the **Python tools** you will use to work with data,
- the **machine learning algorithms** that actually learn from examples.

1

Statistics You Need for Machine Learning

In this chapter we will develop the basic statistical ideas that appear everywhere in machine learning: random variables, distributions, expectation, variance, covariance, correlation, and a few key theorems that explain why learning from data can work.

1.1 Types of Variables

Numerical Variables

Numerical variables take quantitative values.

- **Continuous:** $x \in \mathbb{R}$ Examples: height, temperature, house price.
- **Discrete:** $x \in \mathbb{Z}$ Examples: number of rooms, number of customers.

Categorical Variables

Categorical variables represent non-numerical classes.

- **Nominal (unordered)** Examples: color, country, type of food.
- **Ordinal (ordered)** Examples: rating levels, education level.

Important: Machine learning models expect numerical inputs. Categorical variables must be encoded (one-hot, label encoding, etc.).

1.2 Descriptive Statistics

1.2.1 Mean

The mean (average) of n observations is:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

1.2.2 Median

The median is the middle value of the sorted data. It is more robust to outliers than the mean.

1.2.3 Variance

Variance measures how far the values are spread from the mean:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2.$$

1.2.4 Standard Deviation

Standard deviation is the square root of the variance:

$$\sigma = \sqrt{\mathrm{Var}(X)}.$$

It measures the typical deviation from the mean.

1.2.5 Covariance

Variance tells us how a single variable varies. Covariance tells us how *two* variables vary *together*.

$$\mathrm{Cov}(X,Y) = \mathbb{E}\left[(X - \mu_X)(Y - \mu_Y)\right].$$

Interpretation:

- Positive covariance: when X is above its mean, Y tends to be above its mean.
- Negative covariance: when X is above its mean, Y tends to be below its mean.
- Zero covariance: no linear relationship.

1.3 Correlation

Correlation measures the strength of linear dependence between two variables.

- Range: $[-1, 1]$
- Indicates how two variables move together

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}.$$

- $+1$: perfect positive relationship
- 0: no linear relationship
- -1: perfect negative relationship

Machine learning uses correlation to select and weight relevant features.

1.4 Probability Theory Basics

Machine learning models predict probabilities. Even regression models assume probabilistic noise.

Key ideas:

- **Joint probabilities**: describe combined events.
- **Conditional probabilities**: describe relationships.
- **Marginal probabilities**: describe overall tendencies.

This logic becomes the backbone of all ML algorithms.

1.4.1 Joint Probability

For events A and B:

$$P(A,B) = P(A \cap B).$$

1.4.2 Conditional Probability

$$P(A \mid B) = \frac{P(A,B)}{P(B)}.$$

1.4.3 Marginal Probability

Discrete:

$$P(A) = \sum_{b} P(A, b).$$

Continuous:

$$P(A) = \int P(A, b)\, db.$$

Probability quantifies uncertainty. Conditional probability expresses relationships between events.

1.5 Bayes' Theorem: Updating Beliefs from Data

1.5.1 Formula

Bayes' theorem describes how we update our belief about an event A after observing some evidence B:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}.$$

What does this mean?

Bayes' theorem tells us:

Posterior = Likelihood $\times$ Prior / Evidence

More explicitly:

Key Idea

$$P(A \mid B) = \underbrace{P(B \mid A)}_{\text{How well } A \text{ explains the data}} \quad \underbrace{P(A)}_{\text{What we believed before}} \Big/ \underbrace{P(B)}_{\text{How expected the data is overall}}.$$

- $P(A)$ [**prior**] : What we believed before seeing any data.
- $P(B \mid A)$ [**likelihood**] : How compatible the observed data B is with our hypothesis A.
- $P(B)$ [**evidence**] : The probability of seeing the data under all possible explanations.
- $P(A \mid B)$ [**posterior**] : Our updated belief about A after incorporating evidence B.

Intuition

Bayes' theorem answers the question:

"Given what I just observed, how should I update what I believe?"

It combines two ideas:

1. **What we believed before** (the prior)
2. **How surprising the data is under each hypothesis** (the likelihood)

The evidence $P(B)$ ensures everything stays a valid probability distribution (sums to 1).

Why this matters in Machine Learning

Bayes' theorem is the foundation of:

- Naive Bayes classifier
- Bayesian Linear Regression
- Bayesian Neural Networks
- All probabilistic and generative models

It formalizes **learning from data**. We start with a belief (prior), then see data, and update our belief (posterior).

1.6 Likelihood

The concept of **likelihood** answers a very important question in statistics and machine learning:

Key Idea

"If this parameter θ were true, how well would it explain the data I observed?"

This is different from asking about the probability of future events. Likelihood is specifically about evaluating how plausible a parameter is, given the data we already have.

Definition

For a dataset $x = (x_1, x_2, \ldots, x_n)$ and a model with parameter θ, the likelihood is defined as:

$$L(\theta \mid x) = P(x \mid \theta)$$

This expression should be read as:

> **"Given θ, what is the probability of observing the dataset x?"**

Interpretation

- $x = (x_1, x_2, \ldots, x_n)$: the observed data. These values are **fixed**. We already saw them.
- θ: the parameter of the model. This is **unknown** and is what we want to estimate.
- $P(x \mid \theta)$: the model's prediction about how likely the data is, if θ were the true parameter.

Important Distinction: Probability vs. Likelihood

- **Probability:** θ is fixed, data is random. (Used to predict future observations.)
- **Likelihood:** data is fixed, θ is variable. (Used to evaluate or estimate parameters.)

This is a subtle but essential idea in statistical learning.

The Likelihood Function

Note

The likelihood function is simply the probability of the data, viewed as a function of the parameter:

$$L(x; \theta) = P(x \mid \theta)$$

It tells us how much support the data gives to each possible value of θ.

"Better-fitting parameters give higher likelihood."

Crucially: likelihood is *not* a probability distribution over θ. It does **not** integrate to 1. It is just a scoring function that says which values of θ explain the data best.

The i.i.d. Assumption

In most machine learning models, the data points are assumed to be:

- **independent:** knowing one datapoint tells nothing about another,
- **identically distributed:** all datapoints come from the same distribution.

Under this assumption:

$$L(x \mid \theta) = \prod_{i=1}^{n} P(x_i \mid \theta)$$

Why a product?

Because the joint probability of independent events equals the product of their individual probabilities.

This is a key simplification that makes likelihood computation possible for large datasets.

Summary

- Likelihood measures how well a model with parameter θ explains the observed data.
- It is the central tool for parameter estimation.
- Almost all ML algorithms rely on likelihood:
  - Linear Regression
  - Logistic Regression
  - Neural Networks
  - Decision Trees and Random Forests (implicitly)
  - XGBoost
- Training a model often means:

$$\text{maximize likelihood} \quad \text{or} \quad \text{minimize negative log-likelihood.}$$

In short, likelihood connects your model to your data. It is the mathematical backbone of statistical learning.

1.7 Probability Distributions: Shape of Data

A probability distribution describes **how data behaves**. It tells us which values are likely, which are rare, and what kind of uncertainty or randomness we should expect in the data.

Machine learning models often assume certain distributions because they determine the model's behavior and how it handles noise.

1.7.1 Discrete Distributions

Discrete distributions describe random variables that take values from a finite or countable set (e.g., 0, 1, 2, 3, ...).

Bernoulli Distribution A Bernoulli variable represents a single binary outcome:

$$P(X = x) = p^x(1-p)^{1-x}, \qquad x \in \{0,1\}.$$

Interpretation:

- Models "success or failure" events.
- Examples: coin flip, email is spam/not spam, churning/not churning customer.
- Parameter p is the probability of success.

Binomial Distribution The binomial distribution models the number of successes in n independent Bernoulli trials:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Interpretation:

- Repeating the same experiment n times.
- Examples: number of heads in n coin flips, number of defective items in a batch.
- Gives the distribution of the *count of successes*.

Poisson Distribution The Poisson distribution models the number of events occurring in a fixed time or space interval:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Interpretation:

- Used for rare events happening unpredictably.
- Examples: number of incoming calls per minute, number of website hits per second, number of accidents per day.
- Parameter λ is both the mean and the variance.

1.7.2 Continuous Distributions

Continuous distributions describe variables that take real-number values (e.g., height, temperature, time).

Gaussian (Normal Distribution)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Interpretation:

- The most important distribution in statistics.
- Data clusters around a mean μ with spread σ.
- Many ML models assume Gaussian noise.
- Examples: measurement error, natural variations, height.

Exponential Distribution

$$f(x) = \lambda e^{-\lambda x}$$

Interpretation:

- Models the time between independent random events.
- Examples: time until next earthquake, time until customer arrival.
- Memoryless property: past values do not change future probability.

Multivariate Gaussian

$$f(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Interpretation:

- A Gaussian distribution in multiple dimensions.
- μ is a mean vector, Σ is a covariance matrix.
- Models correlated variables.
- Used heavily in:
  - PCA (Principal Component Analysis)
  - Gaussian Mixture Models (GMMs)
  - Bayesian inference

Why Distributions Matter in Machine Learning

Different distributions encode different assumptions about uncertainty:

- Bernoulli / Binomial $\rightarrow$ binary or count data
- Gaussian $\rightarrow$ continuous data with natural noise
- Poisson $\rightarrow$ rare event counts
- Exponential $\rightarrow$ waiting times
- Multivariate Gaussian $\rightarrow$ correlated continuous data

Choosing an appropriate distribution helps us build models that accurately reflect the data-generating process.

1.8 Statistical Inference: Learning Parameters From Data

Statistical inference is about using data to learn the best values for the parameters of a model. In machine learning, almost every algorithm can be understood as an inference method.

We focus on two fundamental ideas:

- **Maximum Likelihood Estimation (MLE)**
- **Maximum A Posteriori Estimation (MAP)**

These appear throughout regression, classification, probabilistic models, and even deep learning.

1.8.1 Maximum Likelihood Estimation (MLE)

MLE answers a simple question:

Key Idea

"Which parameter value makes the observed data the most likely?"

If $x = (x_1, \ldots, x_n)$ is the observed data and θ is an unknown parameter, the MLE estimate is:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta \mid x)$$

where

$$L(\theta \mid x) = P(x \mid \theta)$$

is the likelihood of the data under the parameter θ.

Why do we take the log?

Because products of probabilities become sums:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \log L(\theta \mid x)$$

This is numerically stable and makes optimization easier.

Intuition

- The data x is considered fixed.
- The parameter θ varies.
- We choose the θ that best *explains* the data.

This idea lies behind most ML algorithms:

- Linear regression (least squares)
- Logistic regression
- Naive Bayes
- Neural networks (via cross-entropy)

All of them optimize a likelihood or its negative log.

1.8.2 Maximum A Posteriori Estimation (MAP)

MAP estimation is the Bayesian version of learning parameters.

Instead of asking:

"Which parameter makes the data most likely?"

we ask:

Key Idea

"Which parameter is most plausible, given the data and my prior beliefs?"

The MAP estimate is:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} P(\theta \mid x)$$

Using Bayes' theorem:

$$P(\theta \mid x) = \frac{P(x \mid \theta)\,P(\theta)}{P(x)}$$

Since $P(x)$ does not depend on θ, we ignore it in maximization:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \left[\log P(x \mid \theta) + \log P(\theta)\right]$$

Interpretation

- **MLE**: only cares about the data.
- **MAP**: cares about data *and* prior knowledge.

Example

Examples of priors:
Priors express what we believe about the parameter θ before seeing data. In machine learning, they often appear naturally as regularization.
1. Prior: "Parameters should be small"
This corresponds to a Gaussian prior:

$$P(\theta) \propto \exp\left(-\theta^2\right),$$

which leads to the penalty θ^2 after taking $-\log$. This is exactly **L2 regularization**.
2. Prior: "Many parameters should be zero"
This corresponds to a Laplace prior:

$$P(\theta) \propto \exp(-|\theta|),$$

which leads to the penalty $|\theta|$. This is **L1 regularization**, which encourages sparsity.

Connection between MLE and MAP

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

- If the prior is flat (uninformative), MAP = MLE.
- If the prior is strong, MAP pulls the estimate toward the prior belief.

MAP can be seen as:

$$\text{MAP} = \text{MLE} + \text{regularization}.$$

This is why many ML algorithms with regularization (ridge, lasso, weight decay) are actually MAP estimators in disguise.

1.9 Linear Regression: Statistical Interpretation

Linear regression is one of the simplest and most fundamental models in machine learning. It describes a relationship between input features and a continuous output.

Model

We assume the data is generated according to:

$$y = X\beta + \varepsilon$$

- X is the matrix of input features.
- β is a vector of unknown parameters.
- ε is random noise (unpredictable variation).

The key probabilistic assumption:

$$\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

This means:

- noise is Gaussian,
- noise has zero mean,
- noise is independent with constant variance.

With this assumption, each y_i is normally distributed around $X_i\beta$.

Least Squares Estimation

The most common way to estimate β is to minimize the sum of squared errors:

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2$$

This objective measures how well the model's predictions match the data.

The minimizer has a closed-form solution:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

This solution exists when $X^T X$ is invertible.

Probabilistic Interpretation

Ordinary Least Squares (OLS) is not only a geometric method—it is also a probabilistic one.

If the noise is Gaussian, then the likelihood of observing y given β is:

$$P(y \mid X, \beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|^2\right)$$

Maximizing this likelihood is equivalent to minimizing the squared error.

Thus:

$$\text{OLS} = \text{MLE under Gaussian noise.}$$

Interpretation:

- Linear regression assumes data is generated as a linear trend + Gaussian noise.
- Fitting the model means choosing β that makes the data most likely.
- This gives a deep statistical foundation to the method.

Linear regression is therefore both:

- a geometric projection problem,
- a probabilistic parameter estimation problem.

1.10 Logistic Regression: Probabilistic Classification

Linear regression predicts a continuous outcome. Logistic regression adapts the idea to predict *probabilities* for binary outcomes (0 or 1).

Model

We model the probability of the outcome being 1 as:

$$P(y = 1 \mid x) = \sigma(w^T x)$$

using the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Properties of the sigmoid:

- outputs values between 0 and 1,
- smoothly increases, S-shaped,
- interpretable as a probability.

This means logistic regression predicts:

"How likely is it that $y = 1$ given the input x?"

Log-Likelihood

Assume each data point (x_i, y_i) is drawn independently. The likelihood of the entire dataset under parameters w is:

$$\ell(w) = \sum_{i=1}^{n} \left[y_i \log \sigma(w^T x_i) + (1 - y_i) \log\big(1 - \sigma(w^T x_i)\big) \right]$$

This is the log-likelihood of the Bernoulli model.

Training = maximize the log-likelihood.

This chooses the w that makes the observed labels most probable.

Connection to Cross-Entropy Loss

Instead of maximizing $\ell(w)$, we minimize its negative:

$$\mathscr{L}(w) = -\ell(w)$$

This is the **cross-entropy loss**, the most widely used loss in classification and neural networks.

Interpretation:

- If the model assigns a high probability to the true label, the loss is small.
- If the model assigns a low probability to the true label, the loss is large.

Thus training logistic regression is about making correct labels more probable.

Summary

Logistic regression is:

- a probabilistic model for binary classification,
- trained via maximum likelihood,
- equivalent to minimizing cross-entropy,
- foundational for neural networks, softmax classifiers, and deep learning.

It takes the linear model $w^T x$ and turns it into a probability through the sigmoid function.

1.11 Bias–Variance Tradeoff: Why Models Fail

When we train a model, we want its predictions $\hat{f}(x)$ to be close to the true output y. A key question in machine learning is:

Why do models make errors, even after training?

The answer is given by the bias–variance decomposition:

$$\mathbb{E}[(y-\hat{f}(x))^2] = \underbrace{\text{Bias}[\hat{f}(x)]^2}_{\text{model too simple}} + \underbrace{\text{Var}[\hat{f}(x)]}_{\text{model too complex}} + \sigma^2_{\text{noise}}.$$

This formula says that prediction error comes from *three* different sources.

1. Bias: Error from Wrong Assumptions

Bias measures how far the model's average prediction is from the true relationship.

- High bias means the model is too simple.
- It cannot capture the true pattern.
- It makes the same mistakes no matter how much data we have.

Examples:

- Using a straight line to fit a quadratic curve.

- A linear model for data with strong nonlinear structure.

This is called **underfitting**.

2. Variance: Error from Sensitivity to Data

Variance measures how much predictions change if we train on a different dataset.

- High variance means the model is too flexible.
- It memorizes random noise in the training data.
- Small changes in the data lead to large changes in the model.

Examples:

- A decision tree that grows too deep.
- A neural network trained with too few data points.

This is called **overfitting**.

3. Irreducible Noise

$$\sigma^2_{\text{noise}}$$

This is randomness in the data that no model can ever explain.

Examples:

- Measurement error.
- Natural variability in human behavior.
- Random fluctuations in physical or economic systems.

Even a perfect model cannot reduce this part of the error.

Why This Matters

The bias–variance tradeoff explains:

- **Underfitting**: high bias, low variance.
- **Overfitting**: low bias, high variance.
- **Regularization**: reduces variance by simplifying the model.
- **Cross-validation**: detects overfitting by testing on unseen data.
- **Ensembles**: average predictions to reduce variance.

The goal of learning is to find a model with a good balance: low bias *and* low variance.

2

Python for Machine Learning

This chapter will introduce the practical tools in Python:

We will use them to load, explore, and visualise real datasets.

3

Machine Learning Algorithms

Here we will study the main learning algorithms: linear regression, logistic regression, decision trees, ensembles, and neural networks, always connecting them back to the statistics and Python tools from the previous chapters.

4

Neural Networks & Deep Learning