CS-576 Machine Learning

Nazar Khan

PUCIT

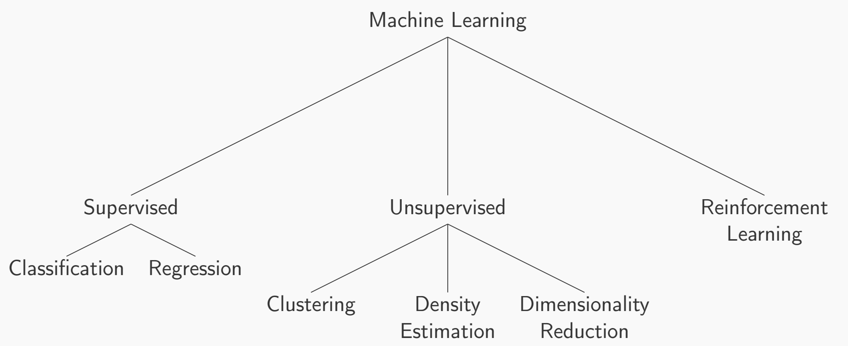
Lectures 1-4
Oct 20,22,27,29 2014

Introduction

Machine Learning and Pattern Recoginition are different names for essentialy the same thing.

- Pattern Recognition arose out of Engineering.
- Machine Learning arose out of Computer Science.
- Both are concerned with automatic discovery of regularities in data

Machine Learning



Supervised Learning

- **Classification**: Assign $\mathbf{x}$ to *discrete* categories.
  - Examples: Digit recognition, face recognition, *etc.*.
- **Regression**: Find *continuous* values for $\mathbf{x}$.
  - Examples: Price prediction, profit prediction.

Unsupervised Learning

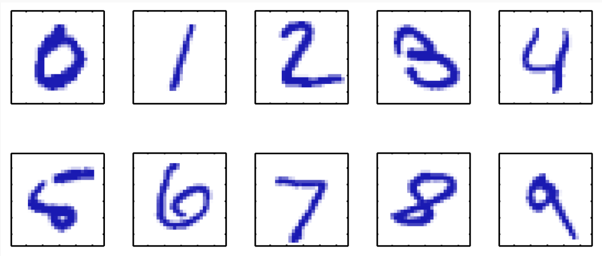
- **Clustering**: Discover groups of similar examples.
- **Density Estimation**: Determine probability distribution of data.
- **Dimensionality Reduction**: Map data to a lower dimensional space.

Reinforcement Learning

- Find actions that maximise a reward. Examples: chess playing program competing against a copy of itself.
- Active area of ML research.
- We will not be covering reinforcement learning in this course.

Classical Algorithms vs. Machine Learning

Problem: Given an image $\mathbf{x}$ of a digit, classify it between $0, 1, \ldots, 9$.



Non-trivial due to high variability in hand-writing.

Classical Algorithms vs. Machine Learning

Classical Approach: Make hand-crafted rules or heuristics for distinguishing digits based on shapes of strokes.
Problems:

- Need lots of rules.
- Exceptions to rules and so on.
- Almost always gives poor results.

Classical Algorithms vs. Machine Learning

ML Approach:

- Collect a large **training set** $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of hand-written digits with known labels $t_1, \ldots, t_N$.
- Learn/tune the parameters of an **adaptive** model.
  - The model can adapt so as to reproduce correct labels for all the training set images.

Classical Algorithms vs. Machine Learning

- Every sample $\mathbf{x}$ is mapped to $f(x)$.
- ML determines the mapping f during the **training phase**. Also called the **learning phase**.
- Trained model f is then used to label a new **test image** $\mathbf{x}_{\text{test}}$ as $f(\mathbf{x}_{\text{test}})$.

Terminology

- **Generalization**: ability to correctly label **new** examples.
  - Very important because training data can only cover a tiny fraction of all possible examples in practical applications.
- **Pre-processing**: Transform data into a new space where solving the problem becomes
  - easier, and
  - faster.

  Also called **feature extraction**. The extracted features should
  - be quickly computable, and
  - preserve useful discriminatory information.

Essential Topics for ML

1. Probability theory – deals with uncertainty.
2. Decision theory – uses probabilistic representation of uncertainty to make optimal predictions.
3. Information theory

Example: Polynomial Curve Fitting

Problem: Given N observations of input x_i with corresponding observations of output t_i, find function $f(x)$ that predicts t for a new value of x.

First, let's generate some data.

```
N=10;
x=0:1/(N-1):1;
t=sin(2*pi*x);
plot(x,t,'o');
```

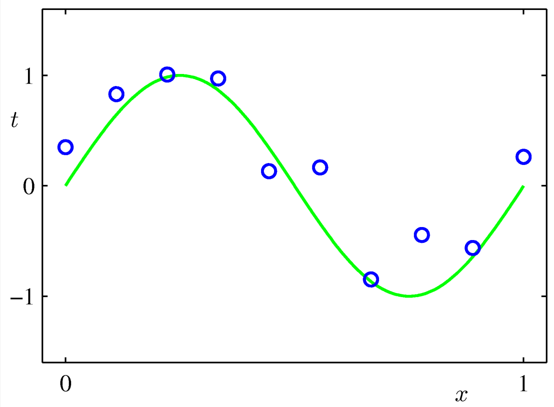
Notice that the data is generated through the function $\sin(2\pi x)$.
Real-world observations are always 'noisy'.
Let's add some noise to the data

```
n=randn(1,N)*0.3;
t=t+n;
plot(x,t,'o');
```


Real-world Data

Real-world data has 2 important properties

1. underlying regularity,
2. individual observations are corrupted by noise.



Learning corresponds to discovering the underlying regularity of data (the $\sin(\cdot)$ function in our example).

Polynomial curve fitting

- We will fit the points (x, t) using a polynomial function

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

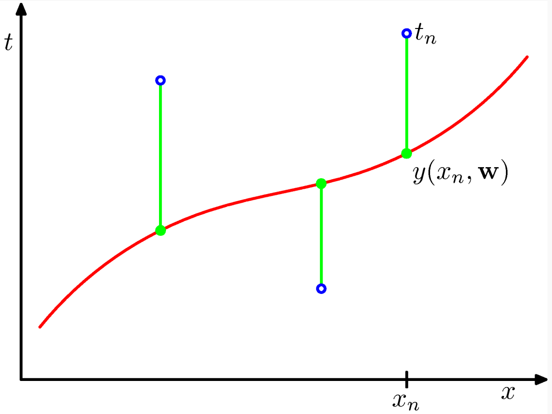
  where M is the **order** of the polynomial.
- Function $y(x, \mathbf{w})$ is a
  - non-linear function of the input x, but
  - a linear function of the parameters $\mathbf{w}$.
- So our model $y(x, \mathbf{w})$ is a **linear model**.

Polynomial curve fitting

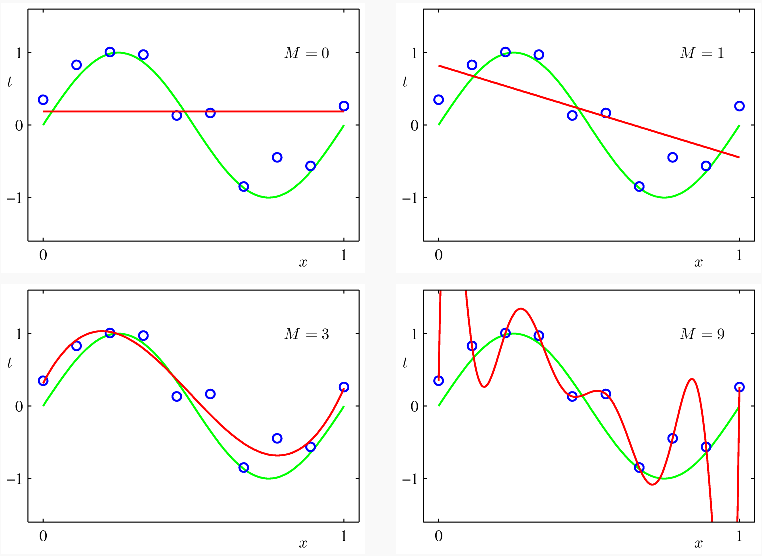
- Fitting corresponds to finding the optimal $\mathbf{w}$. We denote it as $\mathbf{w}^*$.
- Optimal $\mathbf{w}^*$ can be found by **minimising** an **error function**

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

- Can $E(\mathbf{w})$ ever be negative?
- Can $E(\mathbf{w})$ ever be zero?



Geometric interpratation of the sum-of-squares error function.



Over-fitting

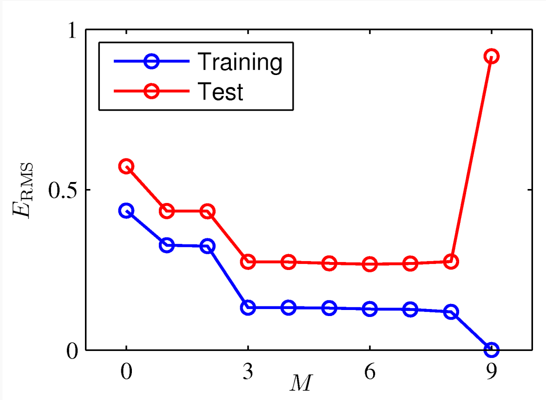
- Lower order polynomials can't capture the variation in data.
- Higher order leads to **over-fitting**.
  - Fitted polynomial passes *exactly* through each data point.
  - But it oscillates wildly in-between.
  - Gives a very poor representation of the real underlying function.
- Over-fitting is bad because it gives bad generalization.

Over-fitting

- To check generalization performance of a certain $\mathbf{w}^*$, compute $E(\mathbf{w}^*)$ on a *new* test set.
- Alternative performance measure: root-mean-square error (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\mathbf{w}^*)}{N}}$$

- Mean ensures datasets of different sizes are treated equally.
- Square-root brings the *squared* error scale back to the scale of the target variable t.



Root-mean-square error on training and test set for various polynomial orders M.

Paradox

- A polynomial of order M contains all polynomials of lower order.
- So higher order should *always* be better than lower order.
- **BUT**, it's not better. Why?
  - Because higher order polynomial starts fitting the noise instead of the underlying function.

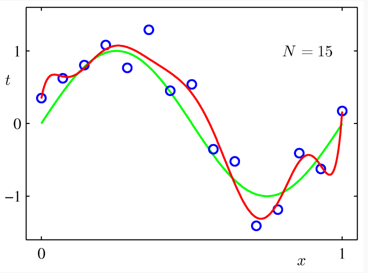
Over-fitting

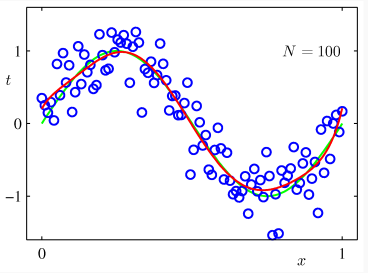
	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- Typical magnitude of the polynomial coefficients is increasing dramatically as M increases.
- This is a sign of over-fitting.
- The polynomial is trying to fit the data points exaclty by having larger coefficients.

Over-fitting

- Large $M \implies$ more flexibility $\implies$ more tuning to noise.
- **BUT**, if we have more data, then over-fitting is reduced.





- Fitted polynomials of order $M = 9$ with $N = 15$ and $N = 100$ data points. More data reduces the effect of over-fitting.
- Rough heuristic to avoid over-fitting: Number of data points should be greater than $k|\mathbf{w}|$ where k is some multiple like 5 or 10.

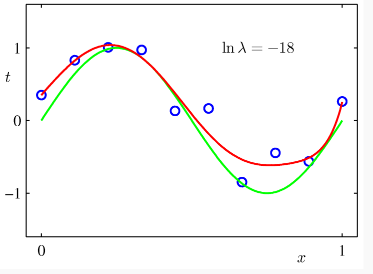
How to avoid over-fitting

- Since large coefficients $\implies$ over-fitting, *discourage large coefficents* in $\mathbf{w}$.

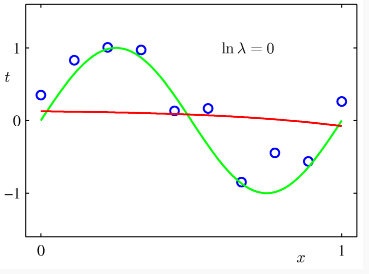
$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

  where $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$ and λ controls the relative importance of the regularizer compared to the error term.
- Also called **regularization, shrinkage, weight-decay**.

For a polynomial of order 9



No over-fitting $\lambda = e^{-18}$

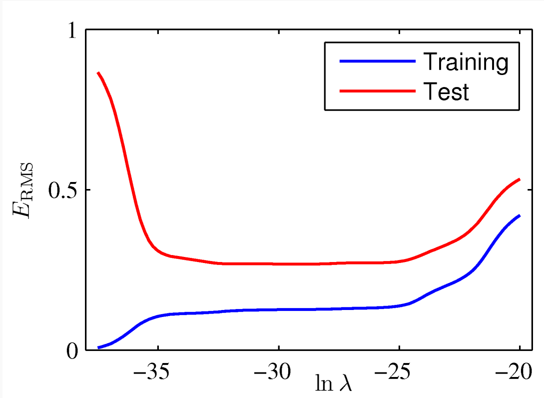


Too much smoothing (no fitting) for $\lambda = 1$.

Effect of regularization

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

- As λ increases, the typical magnitude of coefficients gets smaller.
- We go from over-fitting ($\lambda = 0$) to no over-fitting ($\lambda = e^{-18}$) to poor fitting ($\lambda = 1$).
- Since $M = 9$ is fixed, regularization controls the degree of over-fitting.



Graph of root-mean-square (RMS) error of fitting the $M = 9$ polynomial as λ is increased.

How to avoid over-fitting

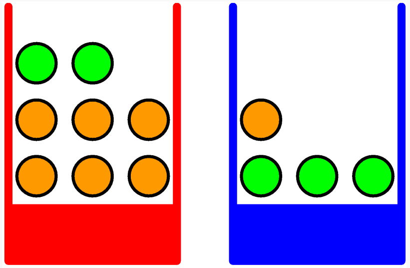
- A more principled approach to control over-fitting is the **Bayesian approach** (to be covered later).
  - Determines the *effective* number of parameters automatically.
- We need the machinery of **probability** to understand the Bayesian approach.
- Probability theory also offers a more principled approach for our polynomial fitting example.

Probability Theory

- **Uncertainty** is a key concept in pattern recognition.
- Uncertainty arises due to
  - Noise on measurements.
  - Finite size of data sets.
- Uncertainty can be **quantified** via probability theory.

Probability

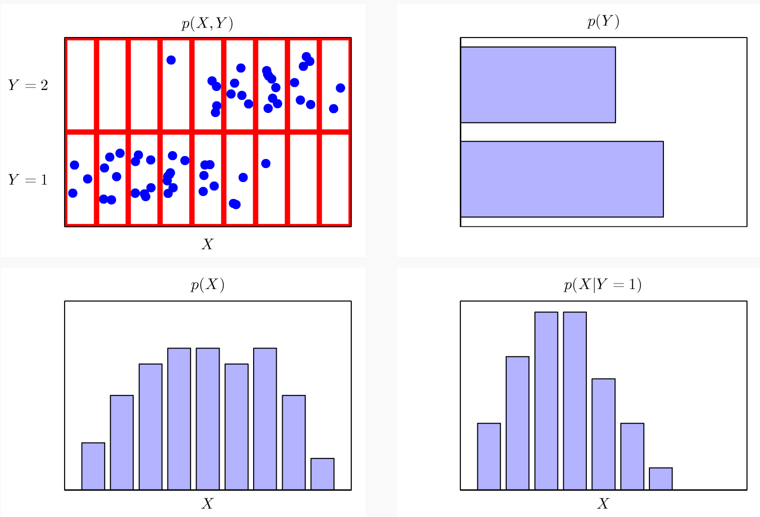
- P(event) is fraction of times event occurs out of total number of trials.
- $P = \lim_{N\to\infty} \frac{\#successes}{N}$.



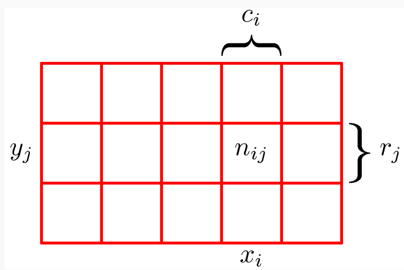
$P(B=b) = 0.6, P(B=r) = 0.4$ $p(apple) = p(F=a) = ?$
p(blue box given that apple was selected) $= p(B=b|F=a) = ?$

Terminology

- Joint $P(X, Y)$
- Marginal $P(X)$
- Conditional $P(X|Y)$



Elementary rules of probability



Elementary rules of probability

- Sum rule: $p(X) = \sum_Y p(X, Y)$
- Product rule: $p(X, Y) = p(Y|X)p(X)$

These two simple rules form the basis of *all* the probabilistic machinery in this course.

- The sum and product rules can be combined to write

$$p(X) = \sum_Y p(X|Y)p(Y)$$

- A fancy name for this is **Theorem of Total Probability**.
- Since $p(X, Y) = p(Y, X)$, we can use the product rule to write another very simple rule

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

- Fancy name is **Bayes' Theorem**.
- Plays a *central role* in machine learning.

Terminology

- If you don't know which fruit was selected, and I ask you which box was selected, what will your answer be?
  - The box with greater probability of being selected.
  - Blue box because $P(B = b) = 0.6$.
  - This probability is called the **prior probability**.
  - Prior because the data has not been observed yet.

Terminology

- Which box was chosen given that the selected fruit was orange?
  - The box with greater $p(B|F = o)$ (via Bayes' theorem).
  - Red box
  - This is called the **posterior probability**.
  - Posterior because the data has been observed.

Independence

- If joint $p(X, Y)$ factors into $p(X)p(Y)$, then random variables X and Y are **independent**.
- Using the product rule, for independent X and Y, $p(Y|X) = p(Y)$.
- Intuitively, if Y is independent of X, then knowing X does not change the chances of Y.
- Example: if fraction of apples and oranges is same in both boxes, then knowing which box was selected does not change the chance of selecting an apple.

Probability density

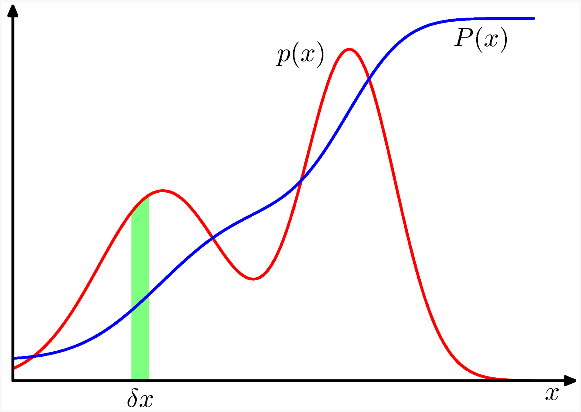
- So far, our set of events was discrete.
- Probability can also be defined for continuous variables via

$$p(x \in (a,b)) = \int_a^b p(x)dx$$

- **Probability density** $p(x)$ is always non-negative and integrates to 1.
- Probability that x lies in $(-\infty, z)$ is given by the **cumulative distribution function**

$$P(z) = \int_{\infty}^{z} p(x)dx$$

- $P'(x) = p(x)$.



Probability density

- Sum rule: $p(x) = \int p(x,y)dy$.
- Product rule: $p(x,y) = p(y|x)p(x)$
- Probability density can also be defined for a multivariate random variable $\mathbf{x} = (x_1, \dots, x_D)$.

$$p(\mathbf{x}) \geq 0$$
$$\int_{\mathbf{x}} p(\mathbf{x})d\mathbf{x} = \int_{x_D} \dots \int_{x_1} p(x_1, \dots, x_D)dx_1 \dots dx_D = 1$$

Expectation

- Expectation is a weighted average of a function.
- Weights are given by $p(x)$.

$$\mathbb{E}[f] = \sum_x p(x)f(x) \qquad \longleftarrow \text{For discrete } x$$
$$\mathbb{E}[f] = \int_x p(x)f(x)dx \qquad \longleftarrow \text{For continuous } x$$

- When data is finite, expectation $\approx$ ordinary average. Approximation becomes exact as $N \to \infty$ (Law of large numbers).

Expectation

- Expectation of a function of several variables

$$\mathbb{E}_x\left[f(x,y)\right] = \sum_x p(x) f(x,y) \qquad \text{(function of } y\text{)}$$

- **conditional expectation**

$$\mathbb{E}_x\left[f|y\right] = \sum_x p(x|y) f(x) dx$$

Variance

Measures variability of a random variable around its mean.

$$\begin{aligned} var\left[f\right] &= \mathbb{E}\left[(f(x) - \mathbb{E}\left[f(x)\right])^2\right] \\ &= \mathbb{E}\left[(f(x)^2\right] - \mathbb{E}\left[f(x^2)\right] \end{aligned}$$

Covariance

- For 2 random variables, **covariance** expresses how much x and y vary together.

$$\begin{aligned} cov\left[x,y\right] &= \mathbb{E}_{x,y}\left[\{x - \mathbb{E}\left[x\right]\}\{y - \mathbb{E}\left[y\right]\}\right] \\ &= \mathbb{E}_{x,y}\left[xy\right] - \mathbb{E}\left[x\right]\mathbb{E}\left[y\right] \end{aligned}$$

- For independent random variables x and y, $cov\left[x,y\right] = 0$.

Covariance

- For multivariate random variables, $cov\left[\mathbf{x},\mathbf{y}\right]$ is a matrix.
- Expresses how each element of $\mathbf{x}$ varies with each element of $\mathbf{y}$.

$$\begin{aligned} cov\left[\mathbf{x},\mathbf{y}\right] &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\{\mathbf{x} - \mathbb{E}\left[\mathbf{x}\right]\}\{\mathbf{y} - \mathbb{E}\left[\mathbf{y}\right]\}^T\right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\mathbf{x}\mathbf{y}^T\right] - \mathbb{E}\left[\mathbf{x}\right]\mathbb{E}\left[\mathbf{y}\right]^T \end{aligned}$$

- Covariance of multivariate $\mathbf{x}$ with itself can be written as $cov\left[\mathbf{x}\right] \equiv cov\left[\mathbf{x},\mathbf{x}\right]$.
- $cov\left[\mathbf{x}\right]$ expresses how each element of $\mathbf{x}$ varies with every other element.

Bayesian View of Probability

- So far we have considered probability as the *frequency of random, repeatable events*.
- What if the events are not repeatable?
  - Was the moon once a planet?
  - Did the dinosaurs become extinct because of a meteor?
  - Will the ice on the North Pole melt by the year 2100?
- For non-repeatable, yet uncertain events, we have the **Bayesian view** of probability.

Bayesian View of Probability

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

- Measures the uncertainty in $\mathbf{w}$ <u>after</u> observing the data $\mathcal{D}$.
- This uncertainty is measured via conditional $p(\mathcal{D}|\mathbf{w})$ and prior $p(\mathbf{w})$.
- Treated as a function of $\mathbf{w}$, the conditional probability $p(\mathcal{D}|\mathbf{w})$ is also called the **likelihood function**.
- Expresses how likely the observed data is for a given value of $\mathbf{w}$.