week2_lecture: Foundation I

머신러닝

김희원

Based on: Probabilistic Machine Learning by Kevin Murphy

숭실대학교
Soongsil University

글로벌미디어학부

Outline

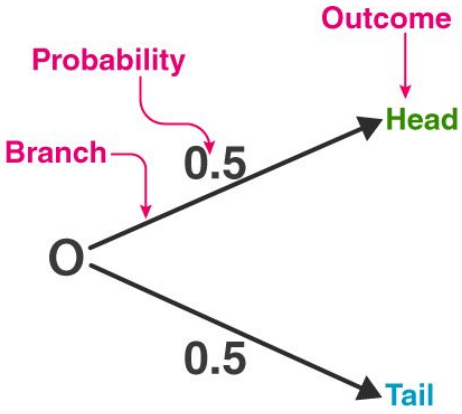
- Basic Probability Theory
  - Univariate Models
  - Multivariate Models
- Statistic

Probability

Probability theory is nothing but common sense reduced to calculation.

— Pierre Laplace, 1812

Probability



Frequentist Interpretation.

Flip the coin many times, we expect it to land heads about half the time.

Bayesian Interpretation.

Quantify our uncertainty or belief about something

Random Variable

- Is a mathematical formalization of a quantity or object which depends on random events.
- It is a mapping or a function from possible outcomes in a sample space to a measurable space, often the real numbers.

Discrete variable	Continuous variable
Countable numbers of value	Infinitely many real values
E.g. number of heads	E.g. time taken to complete a task
Distribution defined by probability mass function	Distribution defined by probability density function

Bayes’ rule

- “Bayesian” is used to refer to inference methods that represent “degrees of certainty” using probability theory, and which leverage Bayes’ rule, to update the degree of certainty given data.

- *Layman term: Update our belief based on prior belief and data so that we infer something with a certain degree of uncertainty.*

Bayes’ rule

Math term:	Posterior Probability	Prior Probability	Likelihood
Layman term:	Updated belief	Prior belief	Data

$$p(H = h|Y = y) = \frac{p(H = h)p(Y = y|H = h)}{p(Y = y)}$$

Bayes' rule - Example

- ❖ Marie is getting married tomorrow at an outdoor ceremony in the desert.
- ❖ In recent years, it has rained only 5 days each year.
- ❖ When it actually rains, the weatherman has forecast rain 90% of the time.
- ❖ When it doesn't rain, he has forecast rain 10% of the time.
- ❖ Unfortunately, the weatherman is forecasting rain for tomorrow.
- ❖ What is the probability it will rain on the day of Marie's wedding?

Let

Event Y = weatherman forecast it rains

Event H = Rain

Prior Probability

$$P(H) = \frac{5}{365} = 0.0137$$

Likelihood

$$P(Y|H) = 0.9$$

$$P(Y|{\sim}H) = 0.1$$

Posterior Probability

$$P(H|Y) = ??$$

Bayes’ rule - Example

We know

$$P(H) = \frac{5}{365} = 0.0137$$

$$P(Y|H) \ = \ 0.9$$

$$P(Y|{\sim}H) \ = \ 0.1$$

$$P(H|Y) \ = ??$$

$$P(H|Y) = \frac{P(H)P(Y|H)}{P(Y)}$$

$$P(H|Y) = \frac{P(H)P(Y|H)}{P(Y|H)P(H) + P(Y|{\sim}H)P({\sim}H)}$$

$$P(H|Y) \ = \frac{0.0137(0.9)}{0.9(0.0137) + 0.1(1 - 0.0137)}$$

$$P(H|Y) = 0.111$$

The probability it will rain on the day of Marie's wedding, given the weatherman is forecasting rain for tomorrow is 0.111

Bernoulli Distributions

- A random variable follows a Bernoulli distribution if it only has two possible outcomes: 0 or 1.

- For example, suppose we flip a coin one time. Let the probability that it lands on heads be *p*. This means the probability that it lands on tails is 1-*p*.

- Thus, we could write:

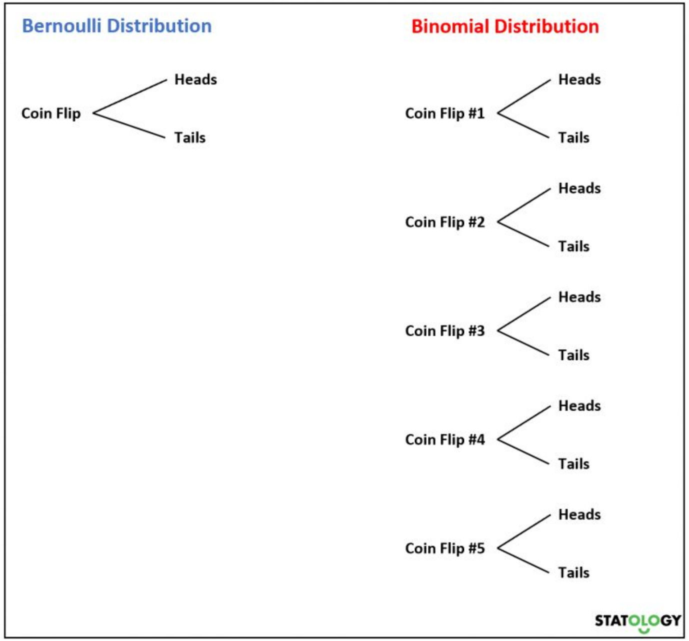
$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

- In this case, random variable *X* follows a Bernoulli distribution. It can only take on two possible values.

- **Now, if we flip a coin multiple times then the sum of the Bernoulli random variables will follow a Binomial distribution.**

https://www.statology.org/bernoulli-vs-binomial/

Binomial Distributions



If a random variable *X* follows a Binomial distribution, then the probability that *X* = *k* successes can be found by the following formula:

$$P(X = k) = {}_{n}C_{k} * pk * (1 - p)^{n-k}$$

n: number of trials
k: number of successes
p: probability of success on a given trial
$_n$C$_k$: the number of ways to obtain *k* successes in *n* trials

https://www.statology.org/bernoulli-vs-binomial/

Multinomial Distributions

- **Bernoulli distribution:** *K* = 2 outcomes, *n* = 1 trial
- **Binomial distribution:** *K* = 2 outcomes, n ≥ 1 trial
- **Multinomial distribution:** *K* ≥ 2 outcomes, n ≥ trials

Binomial	**Multinomial**
E.g. Coin Toss (H/T)	E.g. weather (sunny/windy/gloomy)
Sigmoid function	Softmax function – ensure total probability across all class equal to 1
$\sigma(a) \triangleq \frac{1}{1+e^{-a}}$	$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

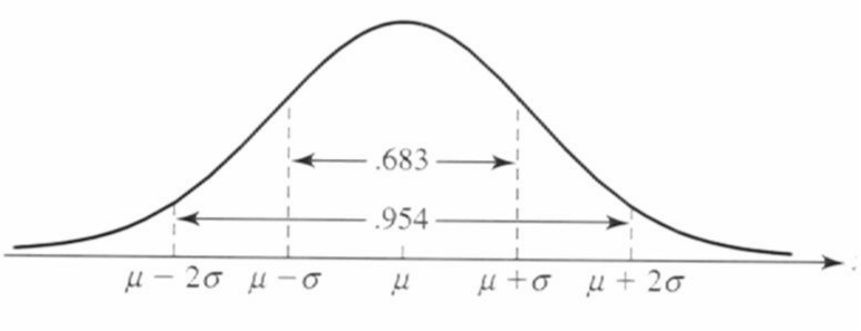
https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Univariate Gaussian Distribution

Defined by two parameters, **mean μ,** which is expected value of the distribution and **standard deviation σ** which corresponds to the expected squared deviation from the mean.

$$N(\mu, \sigma^2)$$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-[(x-\mu)/\sigma]^2/2} \qquad -\infty < x < \infty$$



Fun Facts of Gaussian

The fundamental nature of this distribution and its main properties were noted by Laplace when Gauss was six years old.

Gauss popularized the use of the distribution in the 1800s, and the term “Gaussian” is now widely used in science and engineering

Laplace

Gauss

Image taken from Wikipedia

Univariate Gaussian Distribution

The most widely used distribution of real-valued random variables is the Gaussian distribution, also called the normal distribution.

The following shows the cumulative distribution function (cdf) of a continuous random variable Y.

$$P(y) \triangleq \Pr(Y \leq y)$$

The following shows the probability of Y between a and b is given as following.

$$\Pr(a < Y \leq b) = P(b) - P(a)$$

Cdf of Gaussian is defined as

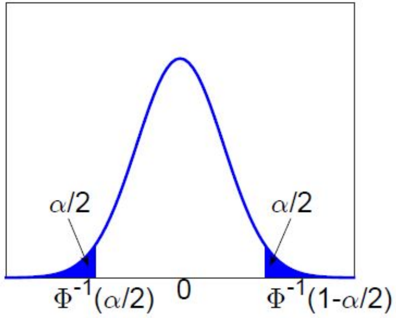
$$\Phi(y; \underset{\text{mean}}{\mu}, \underset{\text{variance}}{\sigma^2}) \triangleq \int_{-\infty}^{y} \mathcal{N}(z|\mu, \sigma^2) dz = \frac{1}{2}[1 + \operatorname{erf}(z/\sqrt{2})] \quad , \quad \begin{array}{l} z = (y - \mu)/\sigma \\ \operatorname{erf}(u) \triangleq \dfrac{2}{\sqrt{\pi}} \displaystyle\int_{0}^{u} e^{-t^2} dt \end{array}$$

Gaussian Distribution

The most widely used distribution of real-valued random variables is the Gaussian distribution, also called the normal distribution.

Probability distribution function (pdf) of Gaussian is

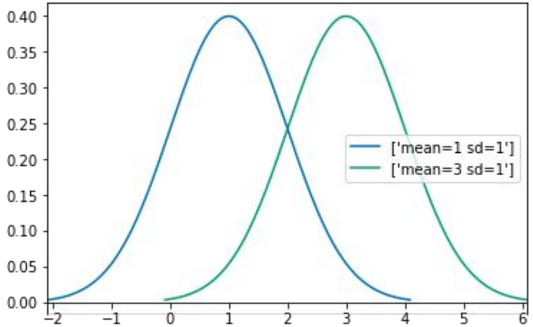
$$\mathcal{N}(y|\underbrace{\mu}_{\text{mean}}, \underbrace{\sigma^2}_{\text{variance}}) \triangleq \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}}}_{\text{Normalization – ensure area under curve = 1}} e^{-\frac{1}{2\sigma^2}(y-\mu)^2}$$



Gaussian Distribution

Mean of pdf of Gaussian is = shift

$$\mathbb{E}\left[Y\right] \triangleq \int_{\mathcal{Y}} y\, p(y)dy$$

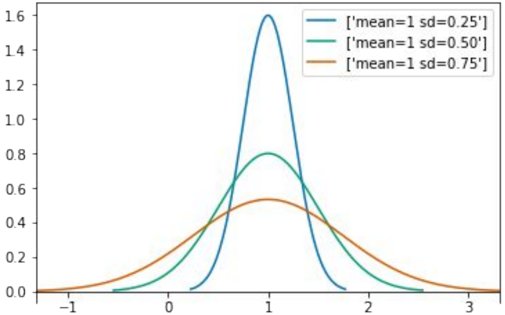


Variance of pdf of Gaussian is = Spread

$$\mathbb{V}\left[Y\right] \triangleq \mathbb{E}\left[(Y-\mu)^2\right] = \int (y-\mu)^2 p(y)dy$$

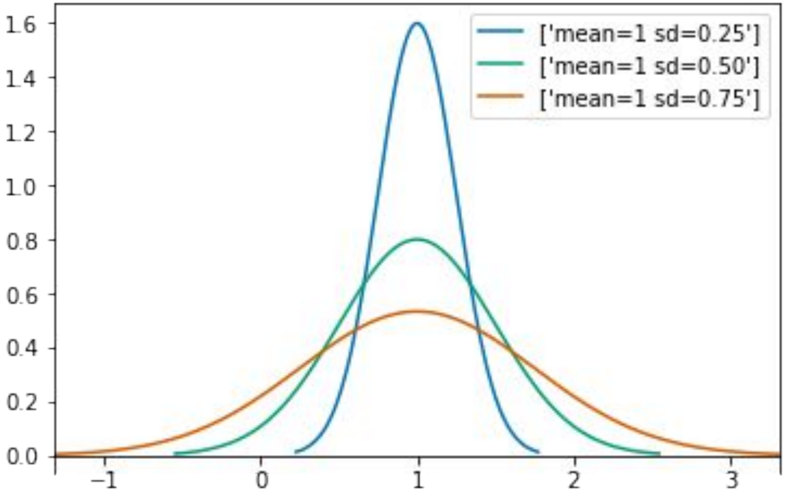
$$= \int y^2 p(y)dy + \mu^2 \int p(y)dy - 2\mu \int yp(y)dy = \mathbb{E}\left[Y^2\right] - \mu^2$$

$$\mathbb{E}\left[Y^2\right] = \sigma^2 + \mu^2$$



Dirac delta function

When variance of Gaussian goes to zero, the distribution becomes narrower.



$$\lim_{\sigma \to 0} \mathcal{N}(y|\mu, \sigma^2) \to \delta(y - \mu)$$

where δ is the **Dirac delta function**, defined by

$$\delta(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

where

$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

Sifting Property

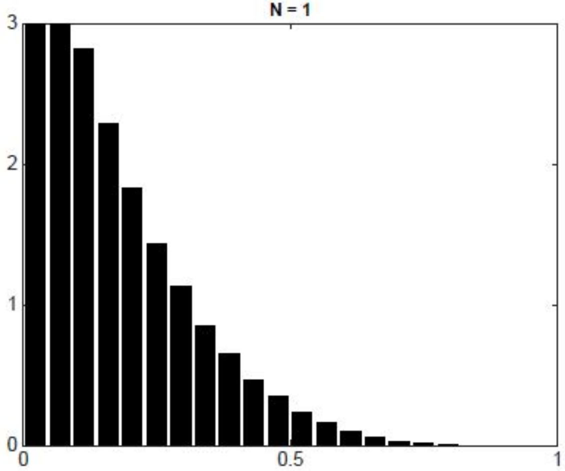
$$\int_{-\infty}^{\infty} f(y)\delta(x - y)dy = f(x)$$

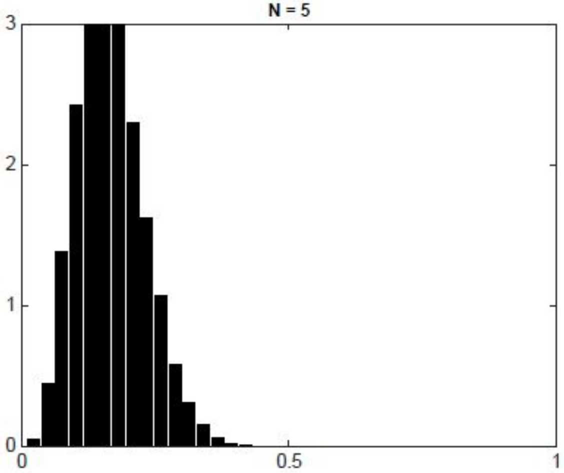
Other Univariate Distributions

- Student t-distribution – hypothesis testing
- Cauchy distribution – useful for Bayesian modelling (heavy tails)
- Laplace distribution – has heavy tails, used in robust linear regression and sparse linear regression
- Beta distribution
- Gamma distribution
    - Exponential distribution
    - Chi-square distribution – used in hypothesis testing
    - Inverse gamma distribution
- Empirical distribution

Central Limit Theorem

When independent random variables are summed up, the distribution converges to a normal distribution even if the original distribution themselves are not normally distributed.





```
sims = RV(Beta(a=2, b=3)).sim(1000000)
```

Monte Carlo Approximation

Suppose x is a random variable, and $y = f(x)$ is some function of x.

It is often difficult to compute the induced distribution $p(y)$ analytically.

One simple but powerful alternative is to draw a large number of samples from the x's distribution, and then to use these samples (instead of the distribution) to approximate $p(y)$ - Monte Carlo Apprx.

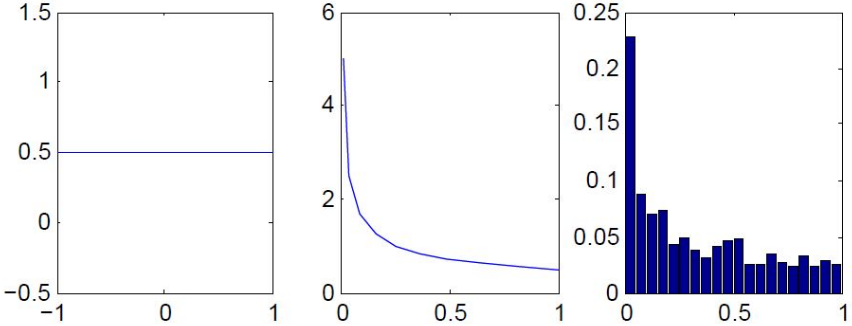


Figure 2.24: Computing the distribution of $y = x^2$, where $p(x)$ is uniform (left). The analytic result is shown in the middle, and the Monte Carlo approximation is shown on the right. Generated by code at figures.probml.ai/book1/2.24.

Let’s move to Colab

Multivariate

- When we have more than one variable → multivariate

- Univariate → Variance
- Multivariate → Covariance

$$\operatorname{Cov}[X,Y] \triangleq \underbrace{\mathbb{E}[\underbrace{(X-\mathbb{E}[X])}\underbrace{(Y-\mathbb{E}[Y])}_{\text{Describing how much data deviate from mean}}]}_{\text{Compute the average deviation}} = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$$

Multivariate

- If x is a D-dimensional random vector, its **covariance matrix** is defined to be the following **symmetric**, **positive semi definite matrix**:

$$
\begin{aligned}
\operatorname{Cov}[\boldsymbol{x}] &\triangleq \mathbb{E}\left[(\boldsymbol{x}-\mathbb{E}[\boldsymbol{x}])(\boldsymbol{x}-\mathbb{E}[\boldsymbol{x}])^{\mathsf{T}}\right] \triangleq \boldsymbol{\Sigma} \\
&= \begin{pmatrix}
\mathbb{V}[X_1] & \operatorname{Cov}[X_1, X_2] & \cdots & \operatorname{Cov}[X_1, X_D] \\
\operatorname{Cov}[X_2, X_1] & \mathbb{V}[X_2] & \cdots & \operatorname{Cov}[X_2, X_D] \\
\vdots & \vdots & \ddots & \vdots \\
\operatorname{Cov}[X_D, X_1] & \operatorname{Cov}[X_D, X_2] & \cdots & \mathbb{V}[X_D]
\end{pmatrix}
\end{aligned}
$$

Multivariate Gaussian Distribution

The most widely used **joint probability distribution** for continuous random variable is the multivariate Gaussian or multivariate normal (MVN).

MVN density is

$$\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right]$$

Mean vector, $E[y]$

Covariance matrix, $\operatorname{Cov}[y]$

Normalization – ensure area under curve = 1

$$\operatorname{Cov}[\boldsymbol{y}] \triangleq \mathbb{E}\left[(\boldsymbol{y}-\mathbb{E}[\boldsymbol{y}])(\boldsymbol{y}-\mathbb{E}[\boldsymbol{y}])^{\mathsf{T}}\right]$$

$$\mathbb{E}\left[\boldsymbol{y}\boldsymbol{y}^{\mathsf{T}}\right] = \boldsymbol{\Sigma}+\boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}$$

D = Dimension

Multivariate Gaussian Distribution

Pdf of MVN is represent by

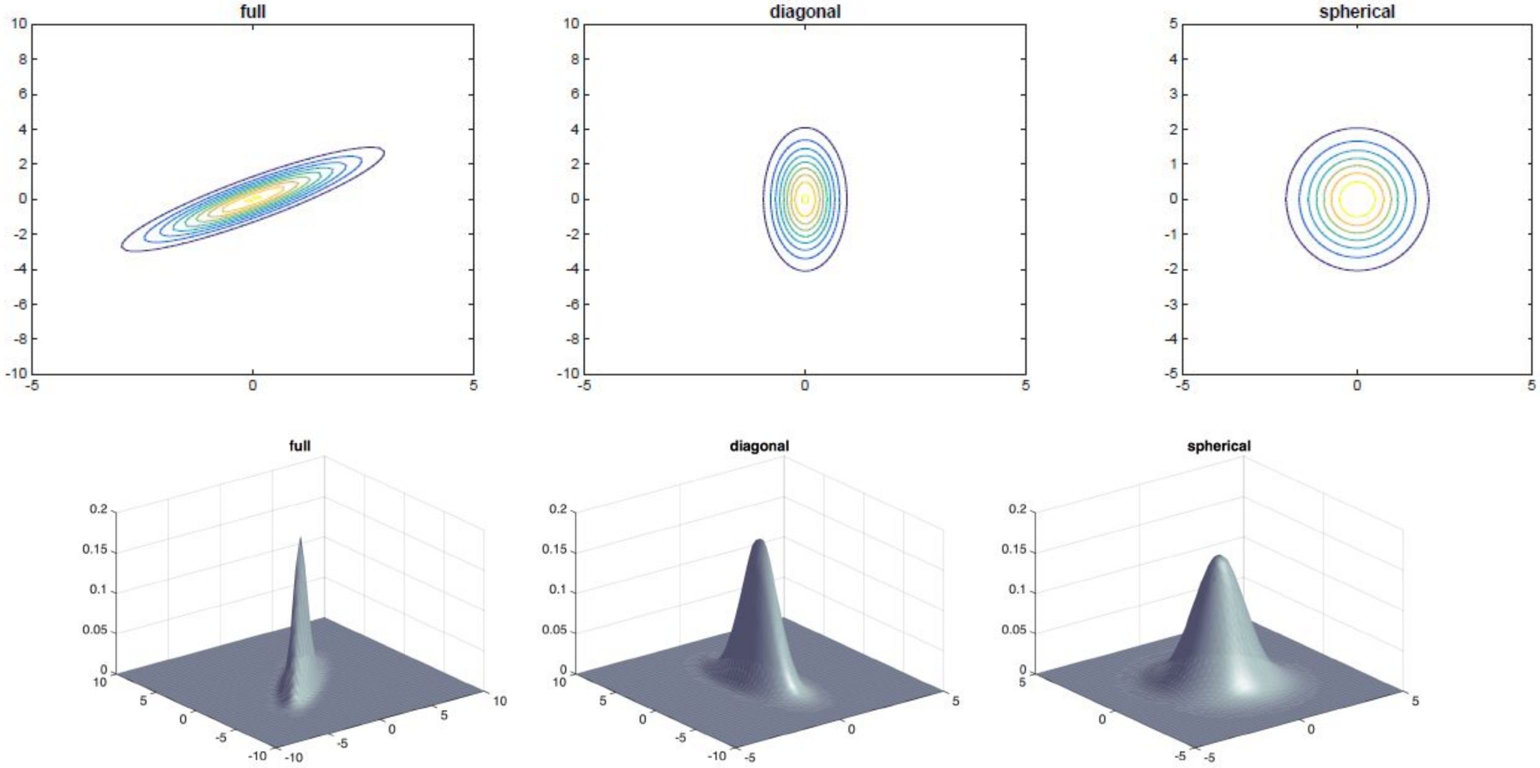
$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \text{ where } \boldsymbol{y} \in \mathbb{R}^2,\ \boldsymbol{\mu} \in \mathbb{R}^2$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_2^2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

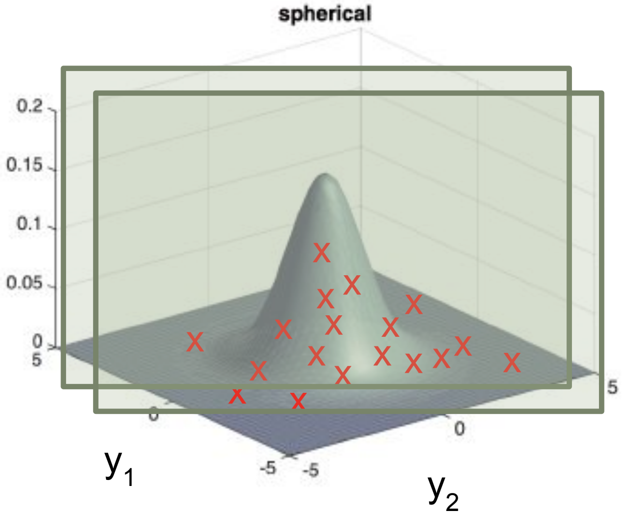
where ρ is the **correlation coefficient**, defined by

$$\operatorname{corr}[Y_1, Y_2] \triangleq \frac{\operatorname{Cov}[Y_1, Y_2]}{\sqrt{\mathbb{V}[Y_1]\,\mathbb{V}[Y_2]}} = \frac{\sigma_{12}^2}{\sigma_1\sigma_2}$$

Multivariate Gaussian Distribution



Example: Imputate missing value from MVN



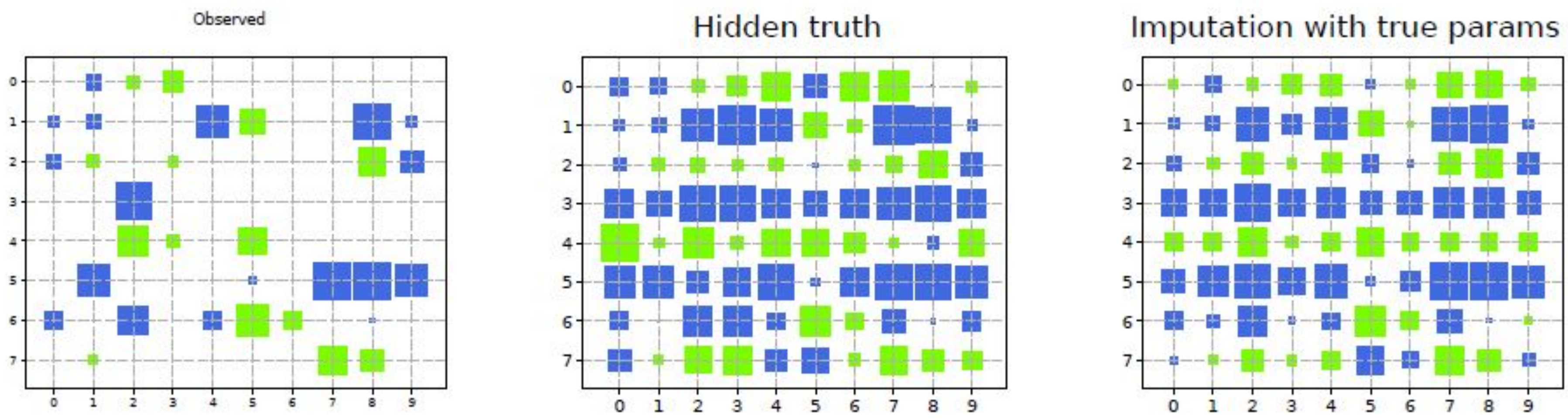
Guess the probability at $y_1 = 0, y_2 = -1$

$p(y_{1\,=}\,0|y_2 = -1)$

Posterior Conditional Formula

$$
\begin{aligned}
p(\mathbf{y}_1|\mathbf{y}_2) &= \mathcal{N}(\mathbf{y}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}) \\
\boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{y}_2 - \boldsymbol{\mu}_2) \\
&= \boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}(\mathbf{y}_2 - \boldsymbol{\mu}_2) \\
&= \boldsymbol{\Sigma}_{1|2}\left(\boldsymbol{\Lambda}_{11}\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{12}(\mathbf{y}_2 - \boldsymbol{\mu}_2)\right) \\
\boldsymbol{\Sigma}_{1|2} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \boldsymbol{\Lambda}_{11}^{-1}
\end{aligned}
$$

Multivariate - Gaussian



Observed

Hidden truth

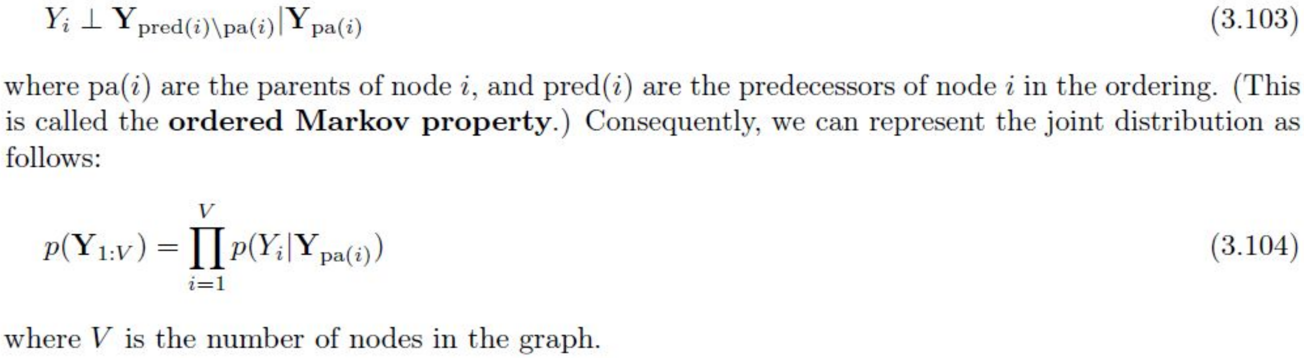
Imputation with true params

Probabilistic graphical models

A probabilistic graphical model or PGM is a joint probability distribution that uses a graph structure to encode conditional independence assumptions.

Node = Random variable
Edge = Direct dependency

Connect them such that each node is conditionally independent of all its predecessors given its parents



$$Y_i \perp \mathbf{Y}_{\mathrm{pred}(i)\setminus\mathrm{pa}(i)}|\mathbf{Y}_{\mathrm{pa}(i)} \tag{3.103}$$

where pa(i) are the parents of node i, and pred(i) are the predecessors of node i in the ordering. (This is called the **ordered Markov property**.) Consequently, we can represent the joint distribution as follows:

$$p(\mathbf{Y}_{1:V}) = \prod_{i=1}^{V} p(Y_i|\mathbf{Y}_{\mathrm{pa}(i)}) \tag{3.104}$$

where V is the number of nodes in the graph.

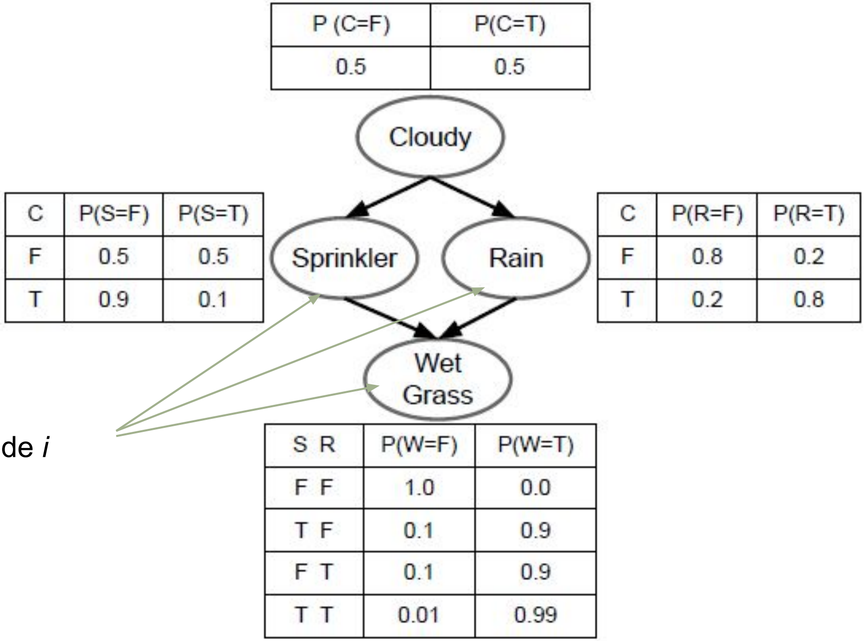
Example PGM: Water sprinkler network

Suppose we want to model the dependencies between 4 random variables:

- C (whether it is cloudy season or not),
- R (whether it is raining or not),
- S (whether the water sprinkler is on or not), and
- W (whether the grass is wet or not).

Joint distribution

$$p(C,S,R,W) = p(C)p(S|C)p(R|C,\cancel{S})p(W|S,R,\cancel{C})$$



P (C=F)	P(C=T)
0.5	0.5

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S R	P(W=F)	P(W=T)
F F	1.0	0.0
T F	0.1	0.9
F T	0.1	0.9
T T	0.01	0.99

Conditional probability distribution (CPD) for node *i*

Example PGM: Markov Chain Network

Suppose we want to create a joint probability distribution over variable-length sequences, p($y_{1:T}$).
MC model is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

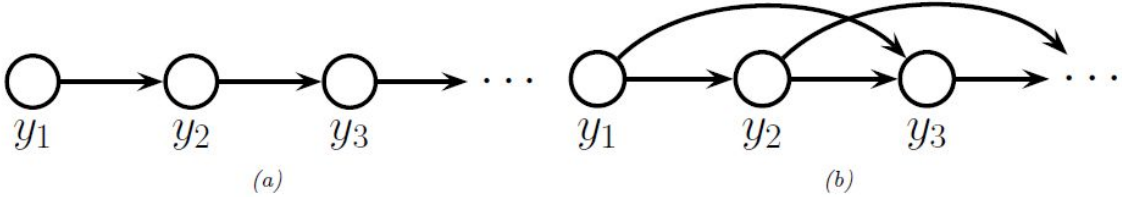


Figure 3.15: Illustration of first and second order autoregressive (Markov) models.

First order Markov Model

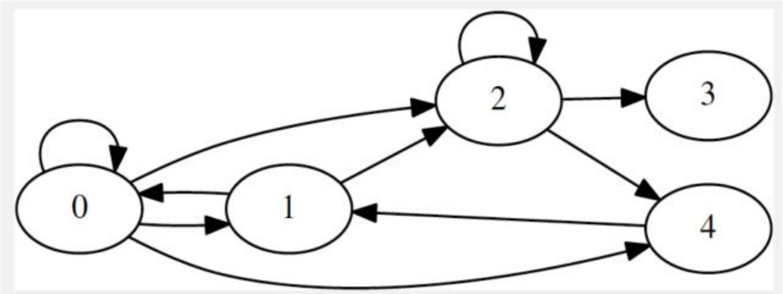
$$p(\mathbf{y}_{1:T}) = p(y_1)p(y_2|y_1)p(y_3|y_2)p(y_4|y_3)\ldots = p(y_1)\prod_{t=2}^{T} p(y_t|y_{t-1})$$

M^th^ order Markov Model

$$p(\mathbf{y}_{1:T}) = p(\mathbf{y}_{1:M}) \prod_{t=M+1}^{T} p(y_t|\mathbf{y}_{t-M:t-1})$$

Example

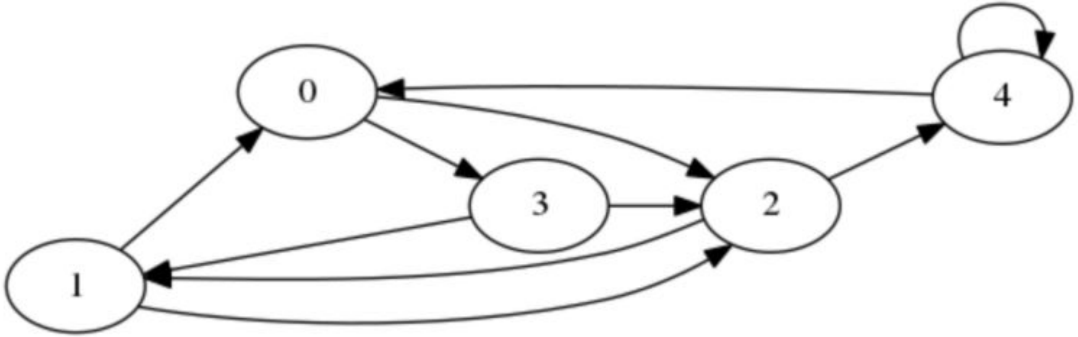
Given this PGM, the adjacent matrix is



	0	1	2	3	4
0	1	1	0	0	0
1	1	0	0	0	1
2	1	1	1	0	0
3	0	0	1	0	0
4	1	0	1	0	0

Example

For the graph given below, which is the adjacency matrix, assuming that all edge weights are 1?



A)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

B)

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

C)

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

D)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Let’s move to Colab

Model Fitting / Training

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \boxed{\mathcal{L}(\boldsymbol{\theta})}$$

Loss function / objective function

Maximum likelihood estimation

$$\hat{\boldsymbol{\theta}}_{\text{mle}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

Estimated Parameter

Probability

Model

Since most optimization algorithms are designed to minimize cost functions, we redefine the objective function to be the (conditional) negative log likelihood or NLL and we minimize NLL

$$\text{NLL}(\boldsymbol{\theta}) \triangleq -\log p(\mathcal{D}|\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

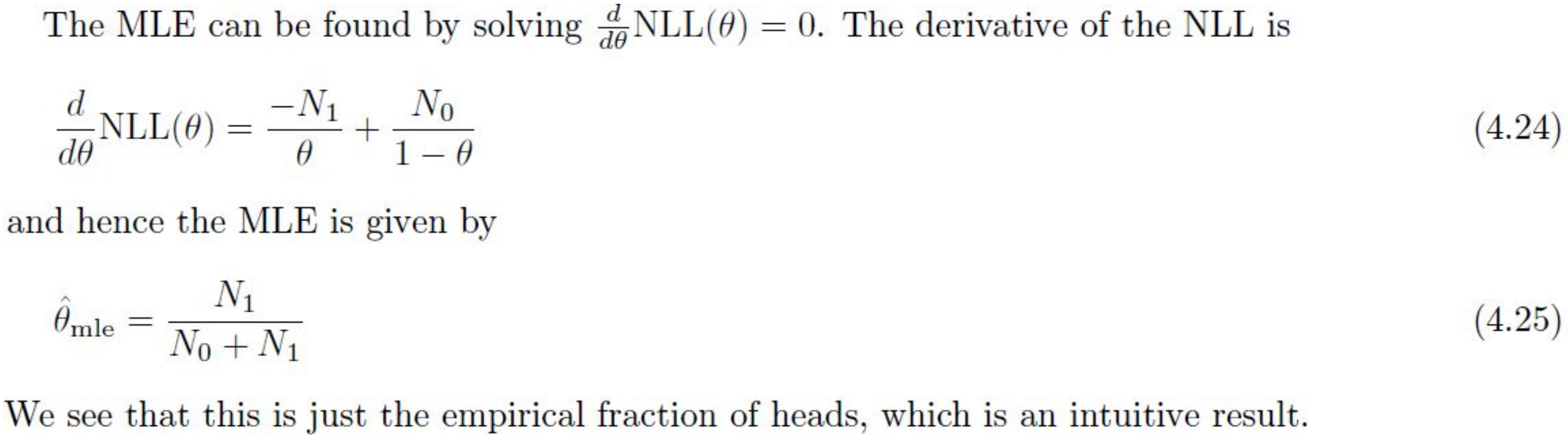
MLE for the Bernoulli distribution

Suppose Y is a random variable representing a coin toss, where the event Y = 1 corresponds to heads and Y = 0 corresponds to tails.
Let θ = p(Y = 1) be the probability of heads. The probability distribution for this rv is the Bernoulli.

The NLL for the Bernoulli distribution is given by

$$
\begin{aligned}
\mathrm{NLL}(\theta) &= -\log \prod_{n=1}^{N} p(y_n|\theta) \\
&= -\log \prod_{n=1}^{N} \theta^{\mathbb{I}(y_n=1)}(1-\theta)^{\mathbb{I}(y_n=0)} \\
&= -\sum_{n=1}^{N} \mathbb{I}(y_n = 1)\log\theta + \mathbb{I}(y_n = 0)\log(1-\theta) \\
&= -[N_1 \log\theta + N_0 \log(1-\theta)]
\end{aligned}
$$

MLE for the Bernoulli distribution



The MLE can be found by solving $\frac{d}{d\theta}\text{NLL}(\theta) = 0$. The derivative of the NLL is

$$\frac{d}{d\theta}\text{NLL}(\theta) = \frac{-N_1}{\theta} + \frac{N_0}{1-\theta} \tag{4.24}$$

and hence the MLE is given by

$$\hat{\theta}_{\text{mle}} = \frac{N_1}{N_0 + N_1} \tag{4.25}$$

We see that this is just the empirical fraction of heads, which is an intuitive result.

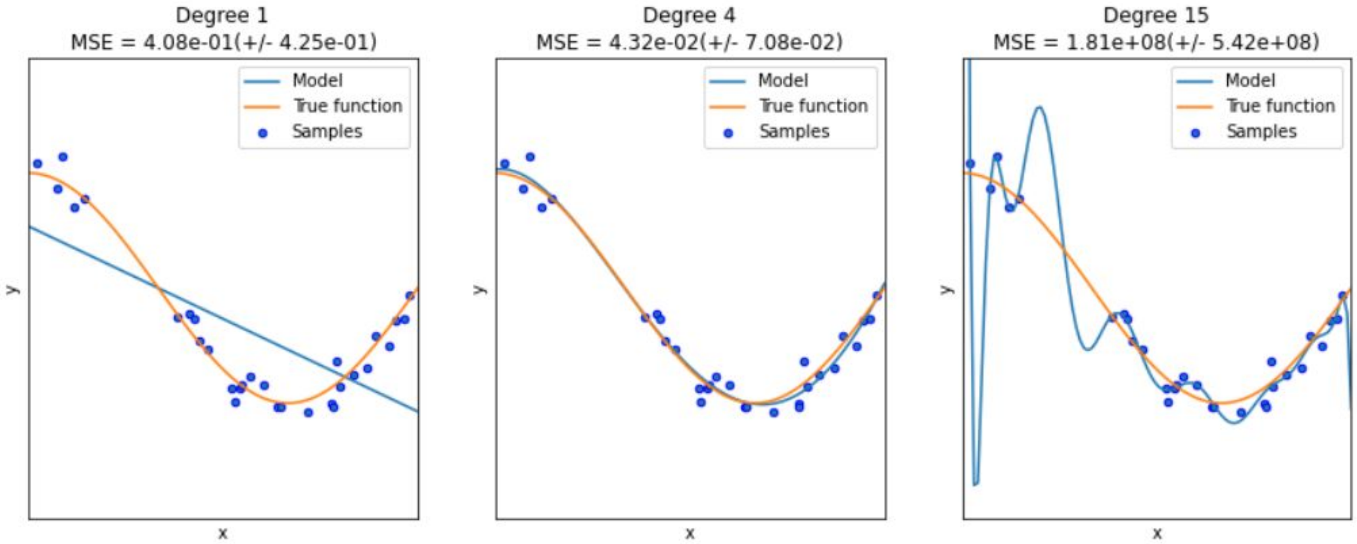
Empirical Risk Minimization

We can generalize MLE by replacing the (conditional) log loss term, with any other loss function.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\theta}; \boldsymbol{x}_n)$$

This is known as empirical risk minimization or ERM, since it is the expected loss where the expectation is taken wrt the empirical distribution.
See Decision Theory Section for more details.

Regularization

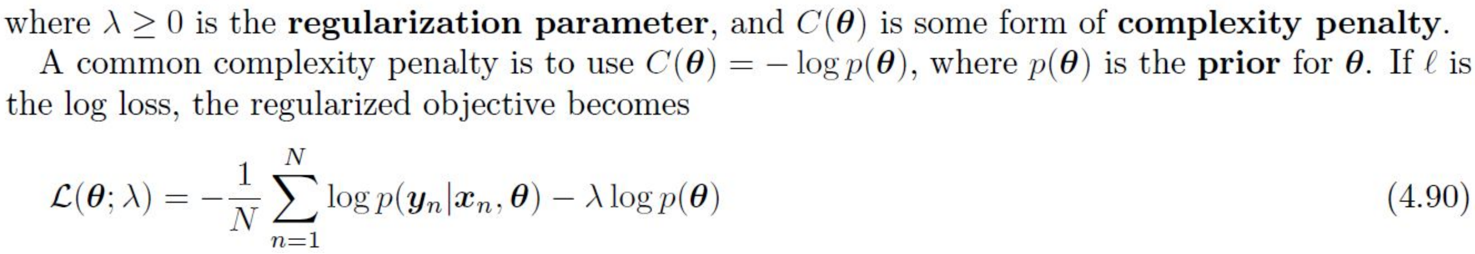


- Enough parameters to perfectly fit the training data.
- Most of the time, empirical distribution ≠ true distribution
- Model unable to predict novel future data → Overfitting

Regularization

- Add penalty term to loss function

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = \left[\frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\theta}; \boldsymbol{x}_n) \right] + \lambda C(\boldsymbol{\theta})$$



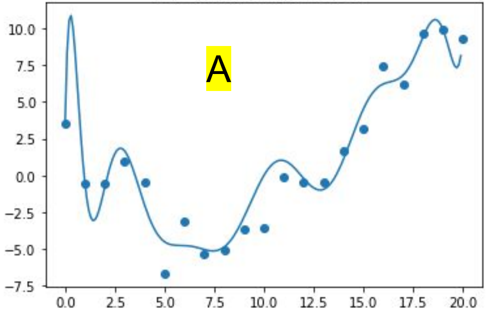
where $\lambda \geq 0$ is the **regularization parameter**, and $C(\boldsymbol{\theta})$ is some form of **complexity penalty**.

A common complexity penalty is to use $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ is the **prior** for $\boldsymbol{\theta}$. If ℓ is the log loss, the regularized objective becomes

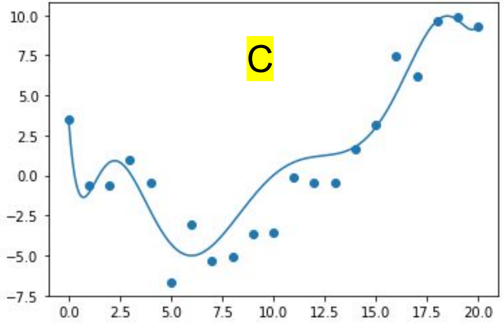
$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = -\frac{1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{y}_n | \boldsymbol{x}_n, \boldsymbol{\theta}) - \lambda \log p(\boldsymbol{\theta}) \tag{4.90}$$

Regularization - How to choose?

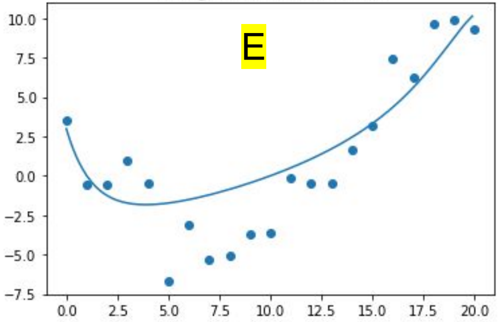
L2 regularizer 1.00000e-10



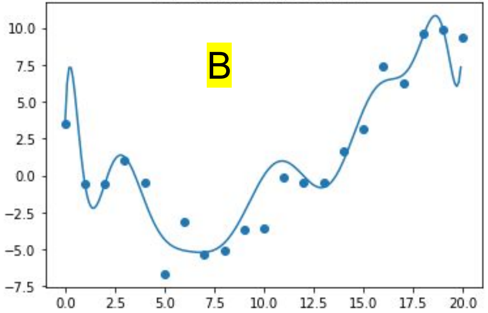
L2 regularizer 1.89574e-04



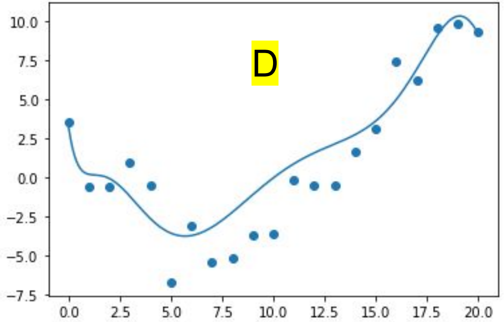
L2 regularizer 1.10776e+00



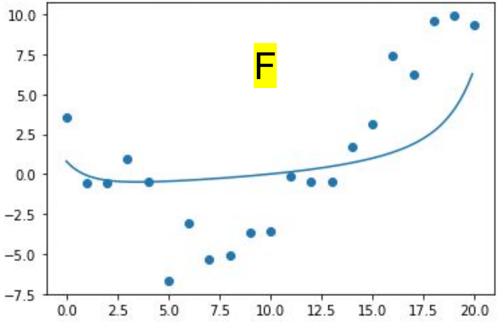
L2 regularizer 1.80117e-09



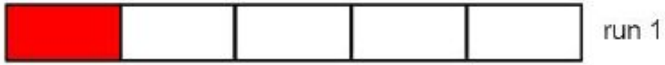
L2 regularizer 3.41455e-03



L2 regularizer 1.99526e+01



Regularization – Validation Set



1. Fit the model on D_{train} (for each setting of λ) with loss function

$$R_\lambda(\boldsymbol{\theta},\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(\boldsymbol{y}, f(\boldsymbol{x};\boldsymbol{\theta})) + \lambda C(\boldsymbol{\theta})$$

2. For each λ, we compute parameter estimate

$$\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}) = \operatorname*{argmin}_{\boldsymbol{\theta}} R_\lambda(\boldsymbol{\theta},\mathcal{D}_{\text{train}})$$

3. Then evaluate its performance on D_{valid} using validation data.

$$R_\lambda^{\text{val}} \triangleq R_0(\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}),\mathcal{D}_{\text{valid}})$$

4. We then pick the value of λ that results in the best validation performance.

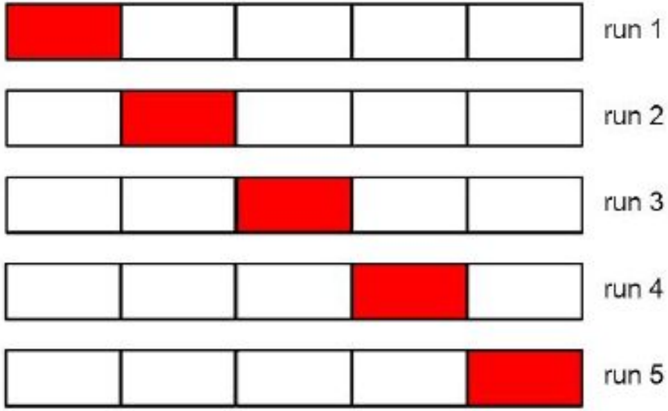
$$\lambda^* = \operatorname*{argmin}_{\lambda\in\mathcal{S}} R_\lambda^{\text{val}}$$

5. Fit the model on with D_{train} and D_{valid} using λ*

$$\hat{\boldsymbol{\theta}}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} R_{\lambda^*}(\boldsymbol{\theta},\mathcal{D})$$

If training data sample size is very small, will have problem.

Regularization – Validation Set



For first CV,

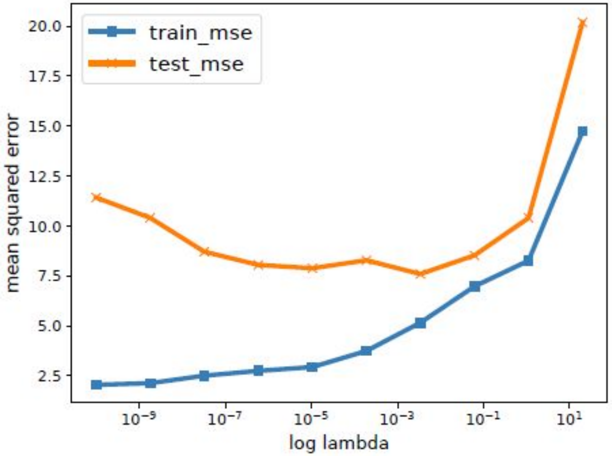
1. Fit the model on D_{train} (for each setting of λ) with loss function
2. For each λ, we compute parameter estimate, θ
3. Then evaluate its performance on D_{valid} (red) using validation data.
4. We then pick the value of λ that results in the best validation performance.
5. We have λ_1

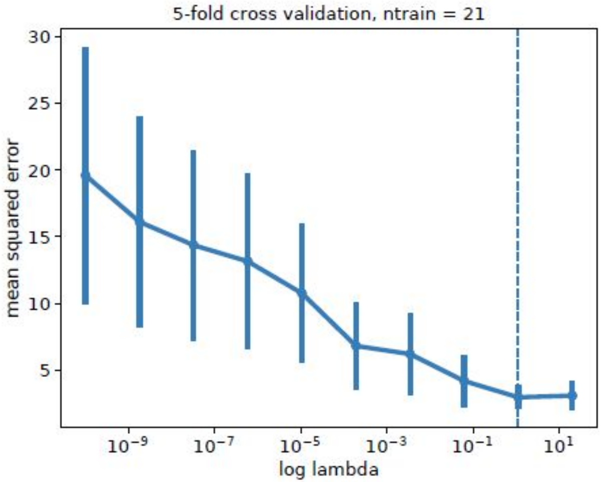
6. Repeat five times.
We will have
CV1: $\lambda_{1,}$ $Loss_1$; **CV2**: $\lambda_{2,}$ $Loss_2$; **CV3**: $\lambda_{3,}$ $Loss_3$; **CV4**: $\lambda_{4,}$ $Loss_4$;
CV5: $\lambda_{5,}$ $Loss_5$

7. We then pick the value of λ* that results in the best validation performance (lowest loss).

8. Fit the model on with D_{train} and D_{valid} using λ*

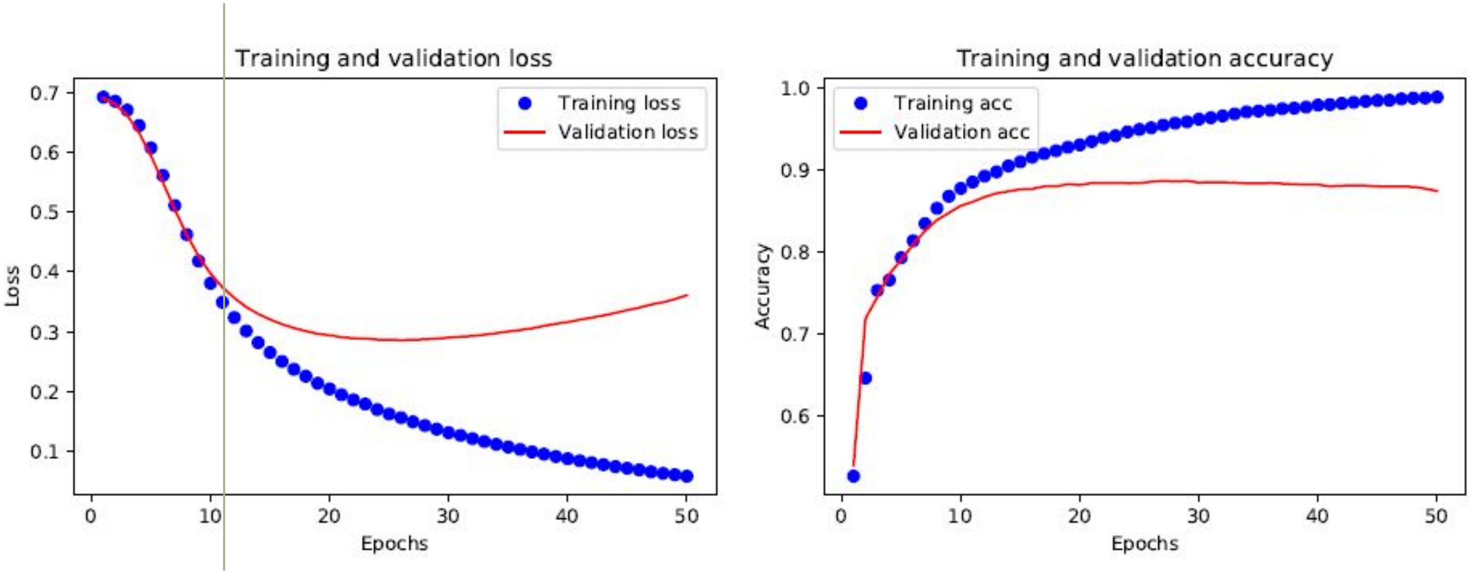
Regularization – Validation Set





Vertical line is the point chosen by the one standard error rule

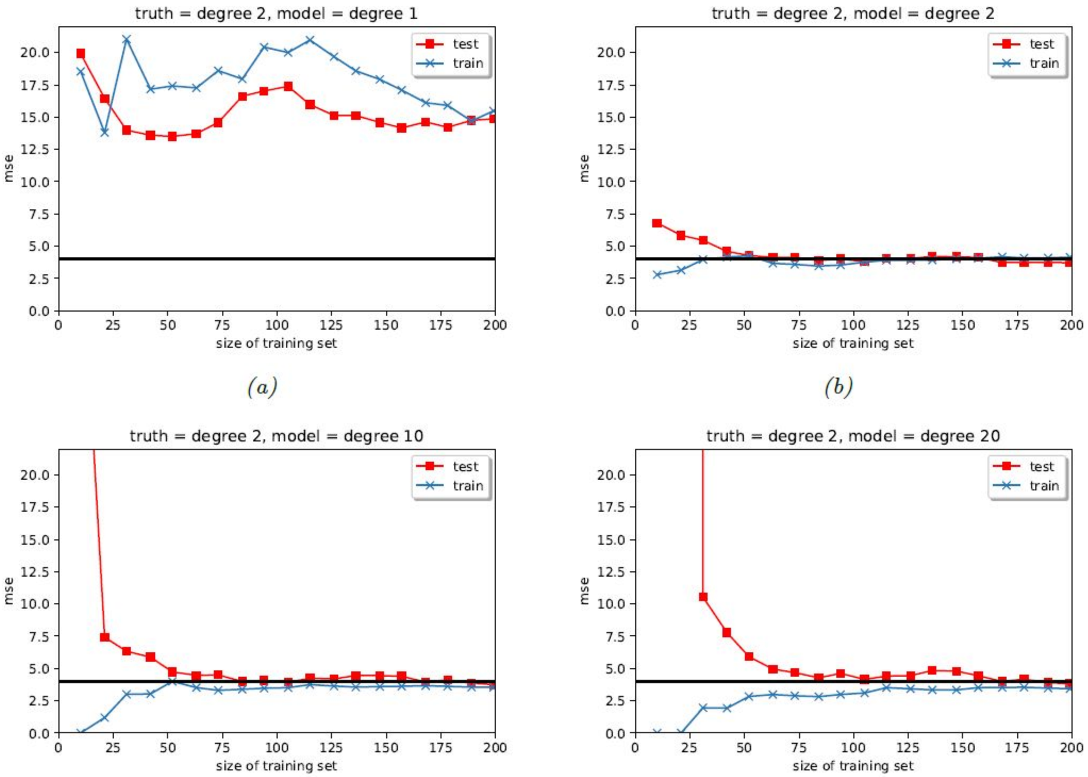
Regularization – Early Stopping



Training and validation loss

Training and validation accuracy

Regularization – More Data



(a)

(b)

- Simple model → underfitting
- Model complexity increases, train and test approaches true error (black line)
- Complex model → overfitting (big gap between train and test), but gap reduces as N increases
- Optimal model (degree 2) – best → converges fastest

Let’s move to Colab