Access Code: **726882**

CMP9794M
Advanced Artificial Intelligence

Heriberto Cuayahuitl

UNIVERSITY OF
LINCOLN

School of Engineering and Physical Sciences

Last Four Weeks

We have studied discrete Bayesian Networks in a fair amount of detail to answer the following:

1. Where do the probabilities come from?

   A: MLE with Laplace/Additive/Dirichlet Smoothing

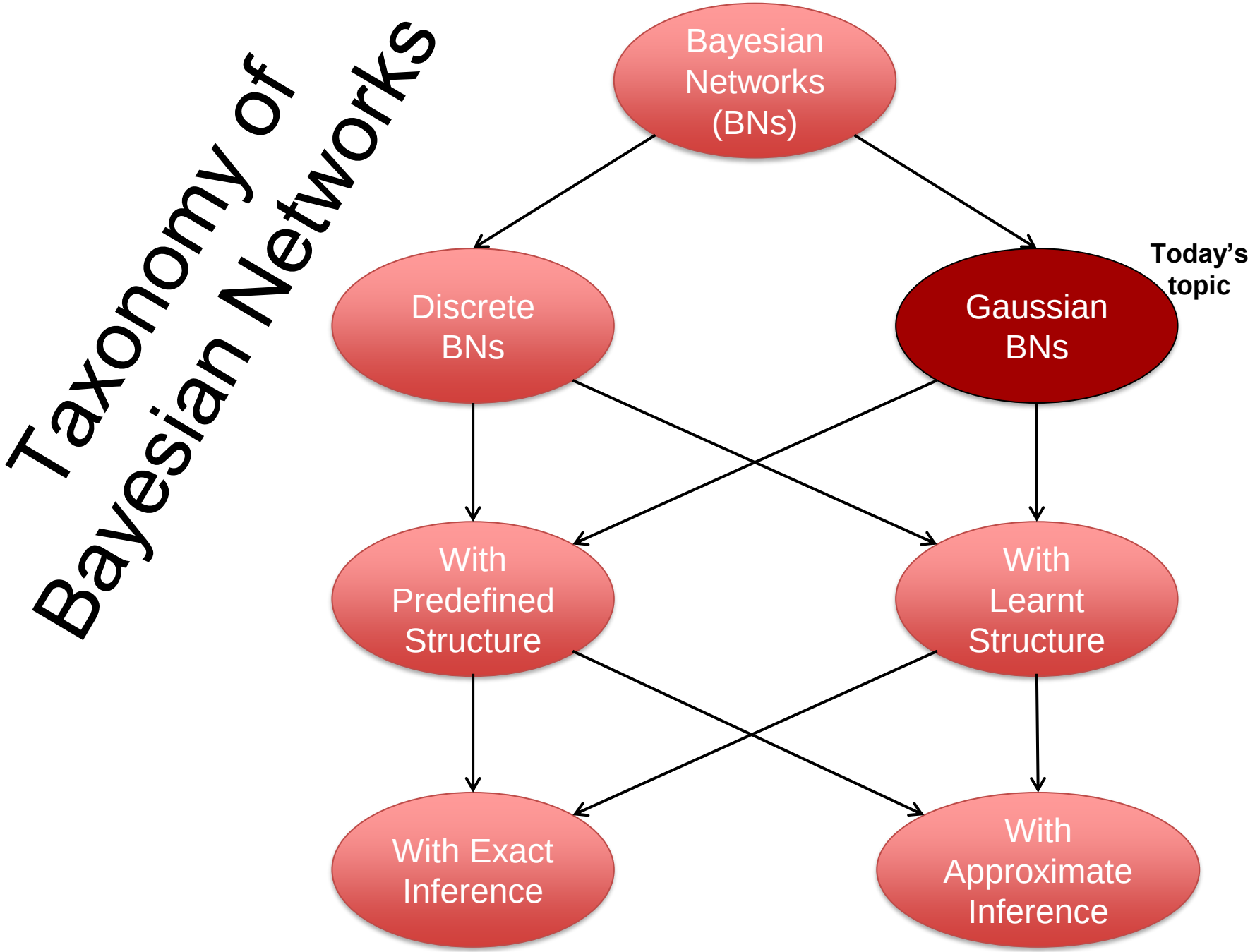
2. Where does the structure come from?

   A: Structure learning via PC-Stable, Hill Climbing, MMHC

3. How do we do probabilistic inference?

   A: Inference by enumeration, variable elimination, rejection sampling, likelihood weighting, Gibbs sampling

Taxonomy of Bayesian Networks



Today

- Preliminaries
  - **Gaussian probability distributions**
  - Gaussian naïve Bayes classifier
- Gaussian Bayesian Networks (GBNs)
  - Representation, parameter learning and inference
- Conditional Gaussian Bayesian Nets (CGBNs)
  - Representation, parameter learning and inference

Discrete vs. Continuous Random Variables

We have used variables with finite sets of values, but *what do we do with random variables such as position, velocity, temperature, pressure?*

	Discretise it	**Use as is: Continuous**
Pro	Reduced problem, efficient learning, interpretability, etc.	More suitable and natural for its type
Con	Loss of information & unnatural solution (e.g. steering wheel)	Computationally expensive, limited scalability.

Gaussian/Normal Distribution

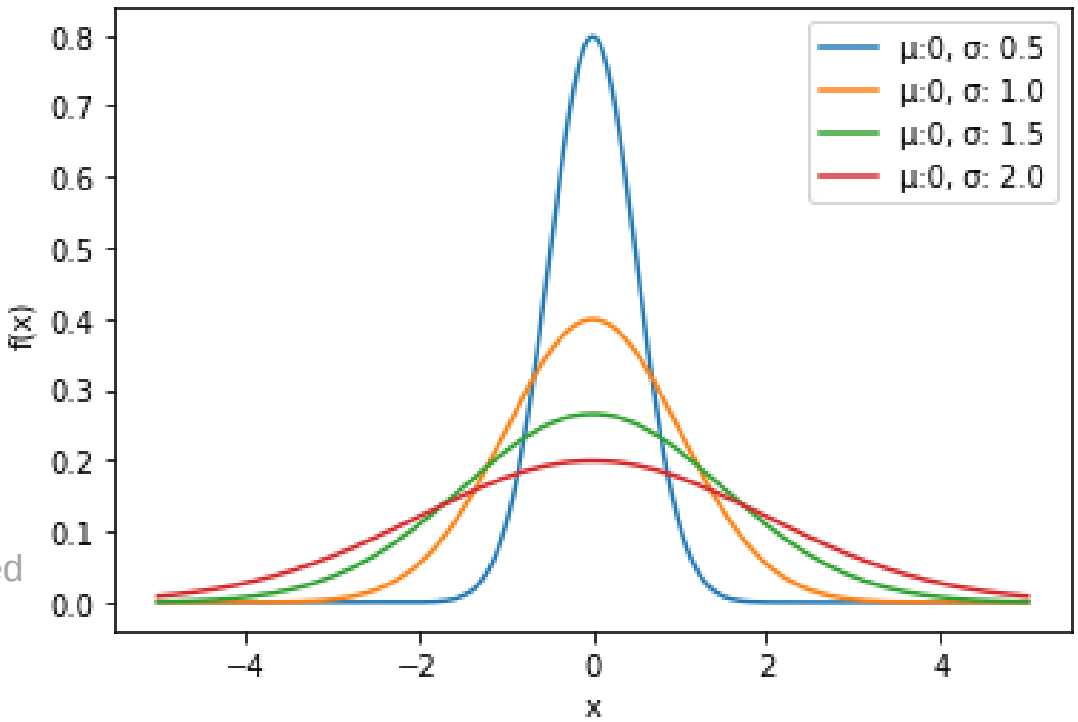
A Gaussian (a.k.a. normal distribution) is a **probability density function (PDF)** defined as

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

Mean

Standard deviation

Examples:



Probability density

(how **densely** the probability is distributed around that point)

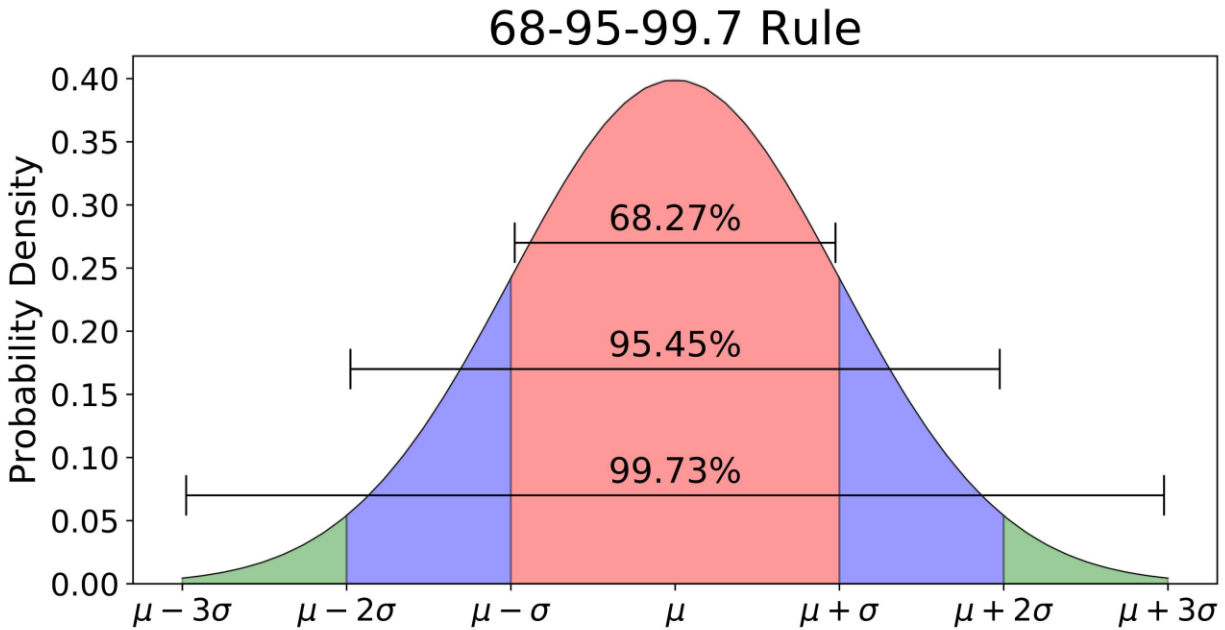
Gaussian/Normal Distribution

A Gaussian distribution is a **probability density function** (likelihood of a continuous random variable) defined as

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

Mean

Standard deviation



Source

Univariate Gaussian Distribution

A Gaussian (a.k.a. normal distribution) is a **PDF** defined as

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

Mean

Standard deviation

$e = 2.71828 \ldots$

$\pi = 3.14159 \ldots$

Equivalent to:

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

Variance

Also written as:

$$f(x) = \mathcal{N}(\mu, \sigma^2)$$

or

$$f(x) = \mathcal{N}(\mu, \sigma)$$

Standard Deviation vs. Variance

Both are measures of dispersion of a set of values.

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

The **standard deviation** (SD) is the squared root of the variance—the lower the SD the closer to the mean.

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

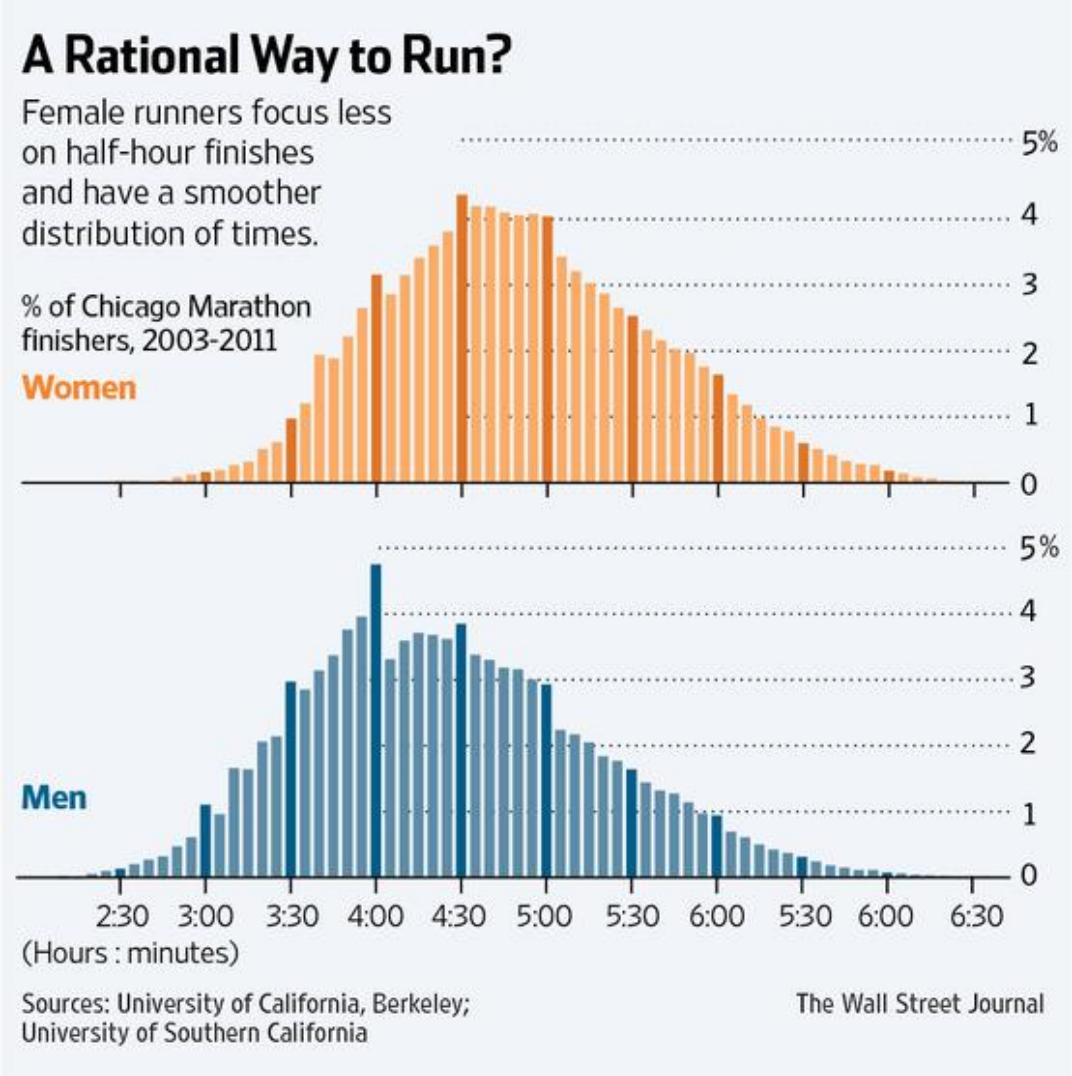
The **variance** tells us how far a set of values is from the mean.

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Example: consider vector $X = \{3,7,7,19,5,10,6\}$, with mean $\mu = 8.14$, variance $\sigma^2 = 23.55$, and SD $\sigma = \sqrt{\sigma^2} = 4.85$.

Whilst variance gives us an amount in terms of square units, the standard deviation converts them back to original units.

Gaussians from Real World Data



Source

Multivariate Gaussian Distribution

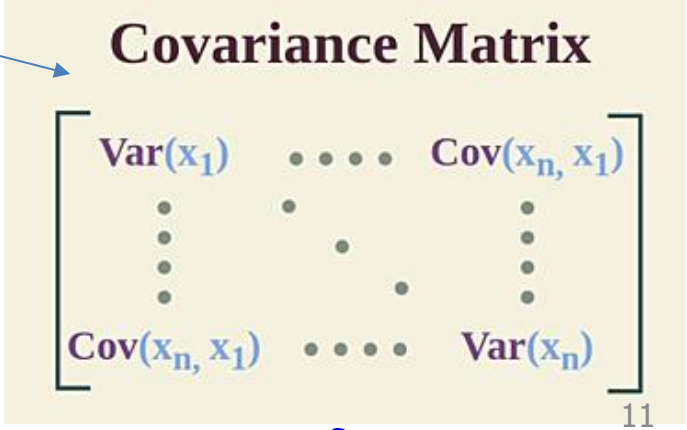
A multivariate Gaussian is a generalisation of the 1D Gaussian to multiple continuous random variables, defined as

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

Notation:

D = dimentionality (number of random variables)
μ =mean vector (D real numbers)
$\sum$ =covariance matrix ($D \times D$),
where $\Sigma_{ij} = \mathrm{cov}(y_i, \mathrm{y_j})$

Covariance (cov) tells us how two random variables change together.
+ cov: variables head in same direction
- cov: variables head in opposite direction
0 cov: no relationship between variables



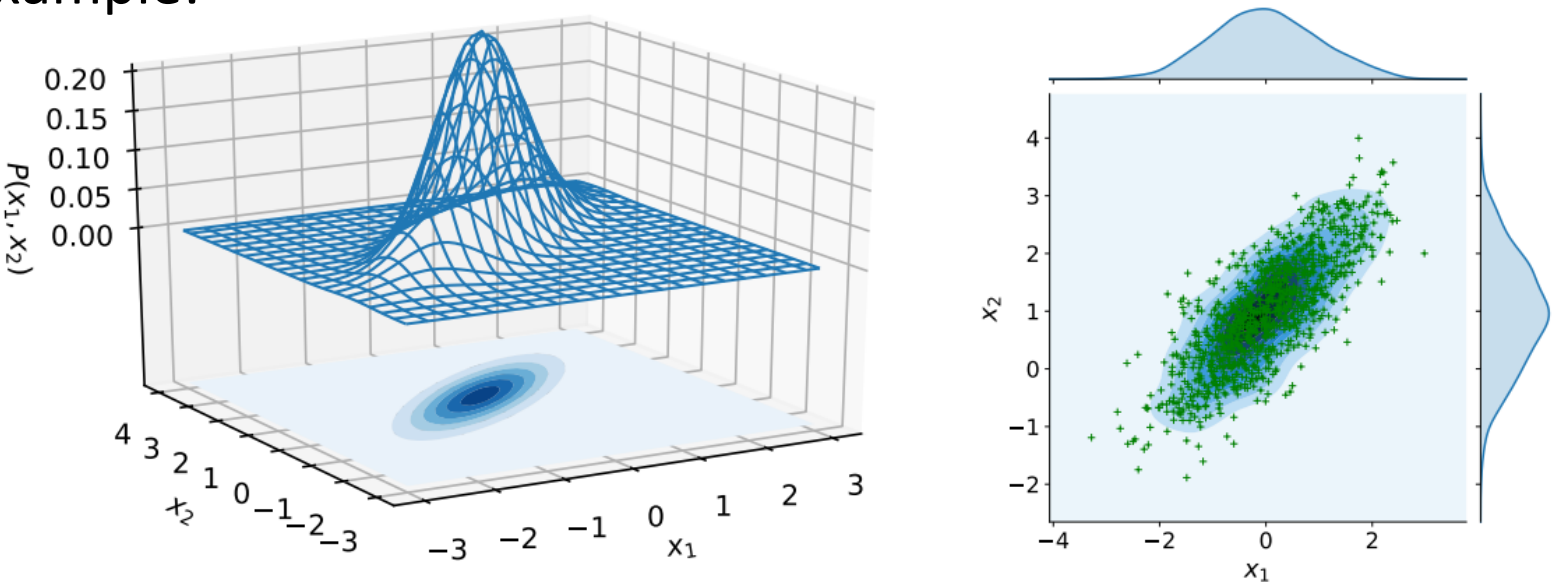
Source

Multivariate Gaussian Distribution

A multivariate Gaussian is a generalisation of the 1D Gaussian to multiple continuous random variables, defined as

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

Example:



(a) 3-d bell curve

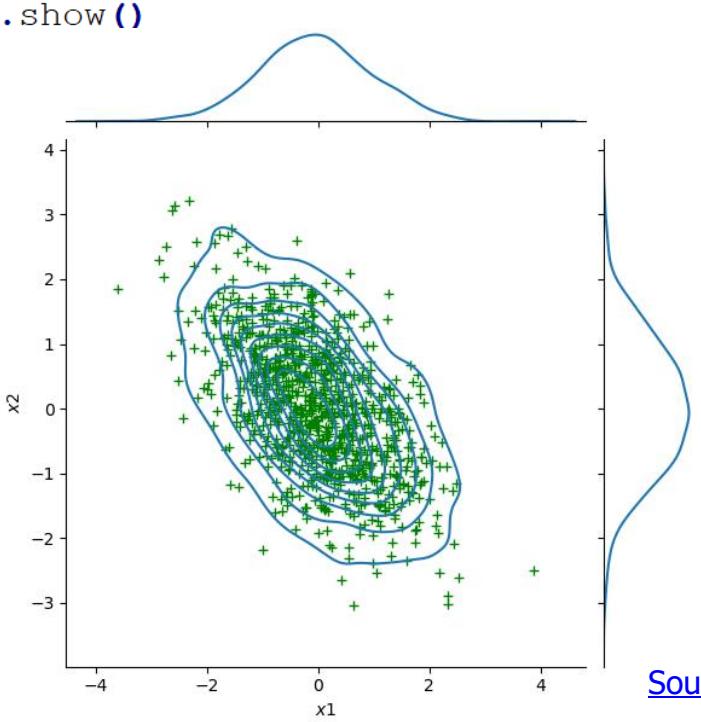
(b) 2-d ellipse contours

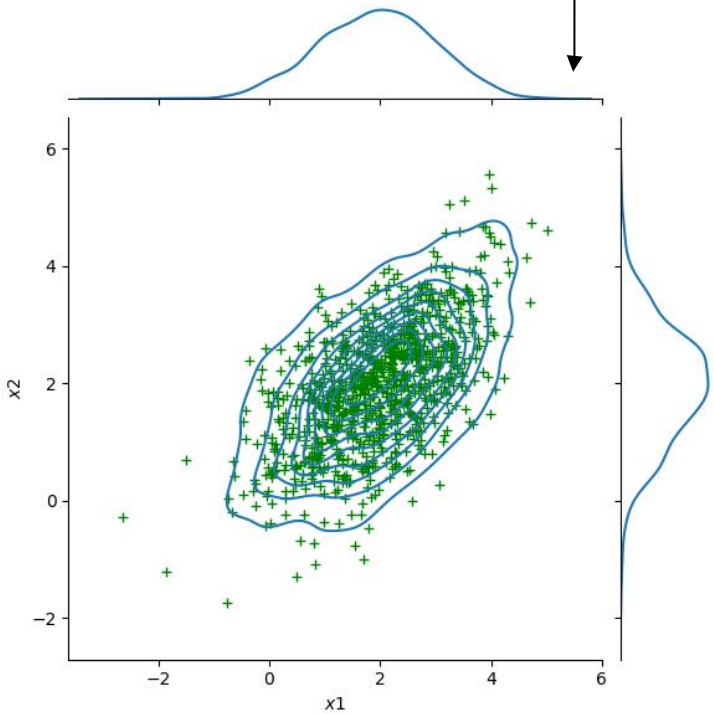
Source

Multivariate Gaussian Distribution

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

mean, cov = [0., 0.], [(1., -0.6), (-0.6, 1.)]
#mean, cov = [2., 2.], [(1., 0.6), (0.6, 1.)]
data = np.random.multivariate_normal(mean, cov, 1000)
df = pd.DataFrame(data, columns=["x1", "x2"])
g = sns.jointplot("x1", "x2", data=df, kind="kde")
g.plot_joint(plt.scatter, c="g", s=30, linewidth=1, marker="+")
g.set_axis_labels("$x1$", "$x2$");
plt.show()
```





Source

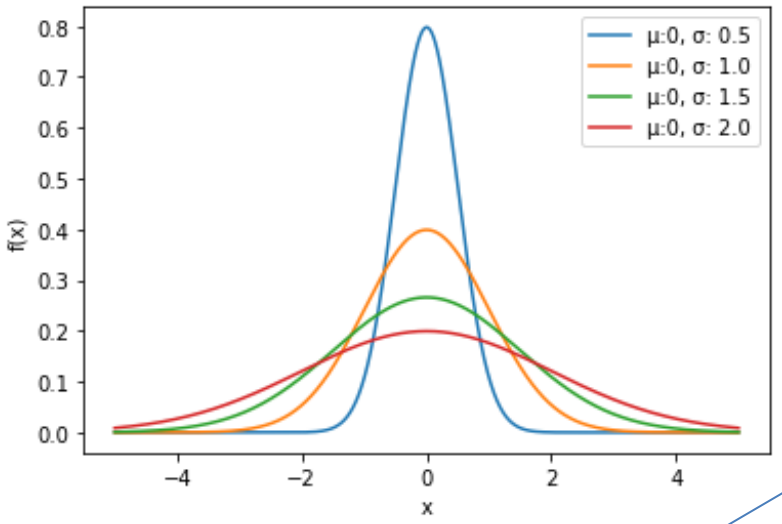
So, Gaussian Distributions are …

Univariate or Multivariate probability density functions (PDFs)

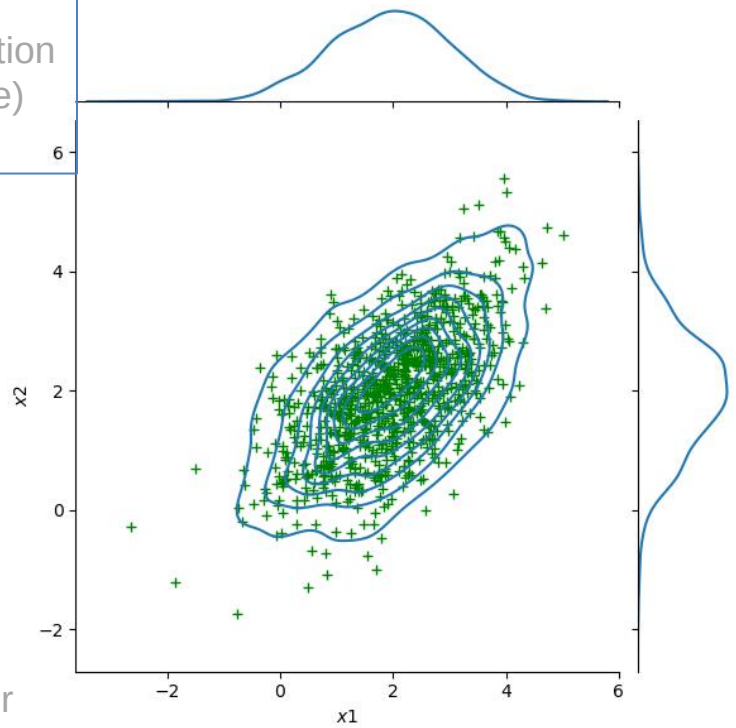
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Mean

Standard deviation (sqrt of variance)



or



Mean vector

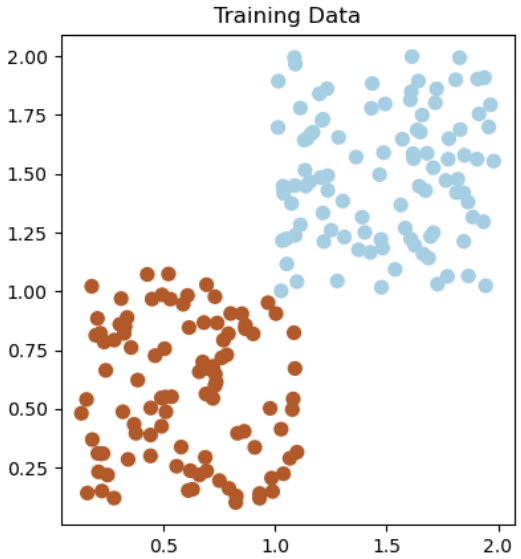
Covariance matrix

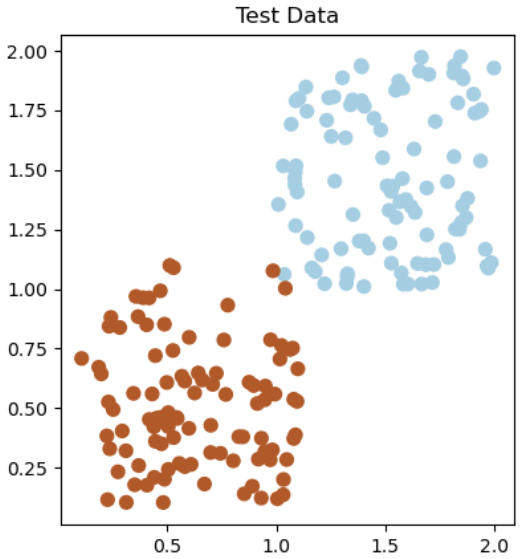
$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu)\right]$$

Today

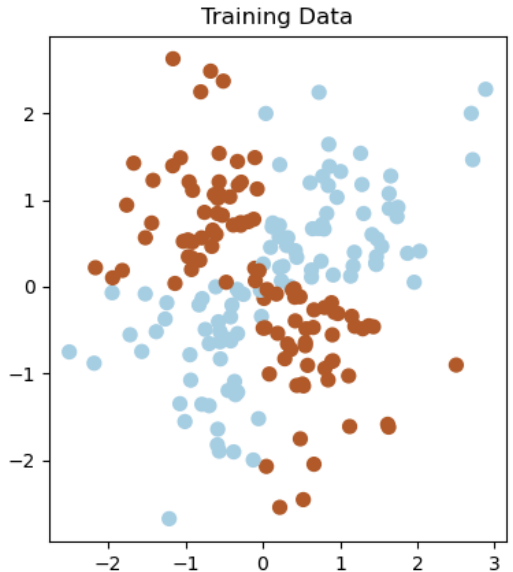
- Preliminaries
  - Gaussian probability distributions
  - **Gaussian naïve Bayes classifier**
- Gaussian Bayesian Networks (GBNs)
  - Representation, parameter learning and inference
- Conditional Gaussian Bayesian Nets (CGBNs)
  - Representation, parameter learning and inference

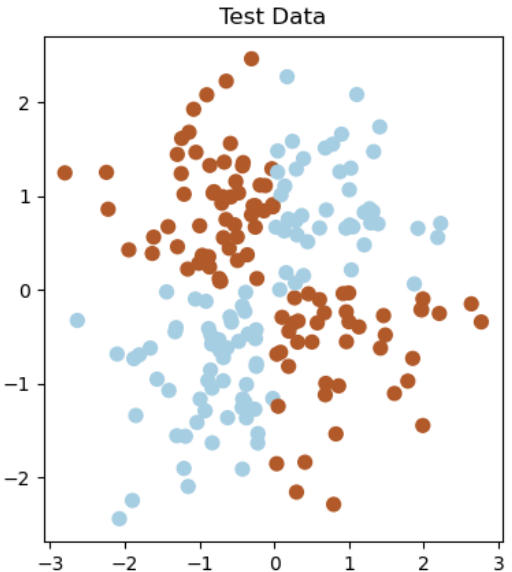
Example Toy Data: Continuous Inputs, Discrete Output





Inputs X (two dimensional)
Outputs Y (blue/brown dots)





Naïve Bayes for Continuous Inputs

- Naïve Bayes treats inputs X_i as univariate Gaussian distributions, which need estimates of their means

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \, \delta(Y^j = y_k)$$

and variances (or standard deviations $\hat{\sigma} = \sqrt{\hat{\sigma}^2}$)

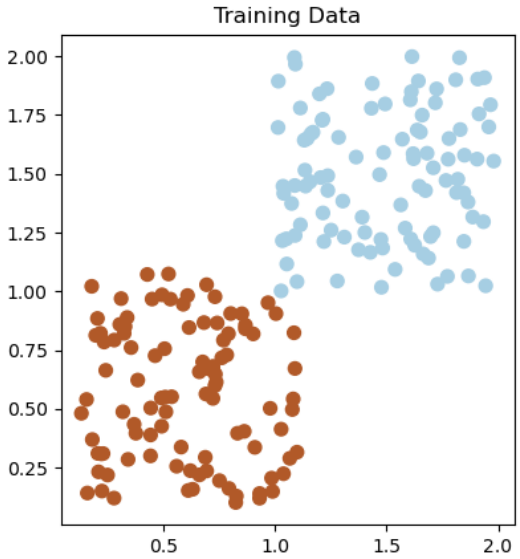
$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j \left(X_i^j - \hat{\mu}_{ik}\right)^2 \delta(Y^j = y_k)$$

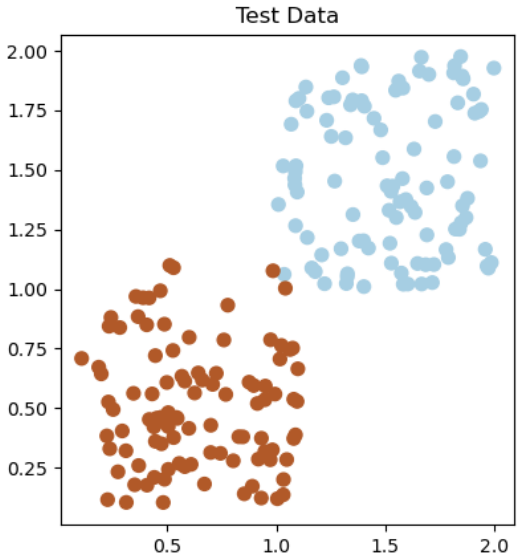
where $\delta(Y^j = y_k) = \begin{cases} 1 & \text{if } Y = y_k \\ 0, & \text{otherwise} \end{cases}$

Indexes:
i refers to random vars.,
j to the training example,
k to the label

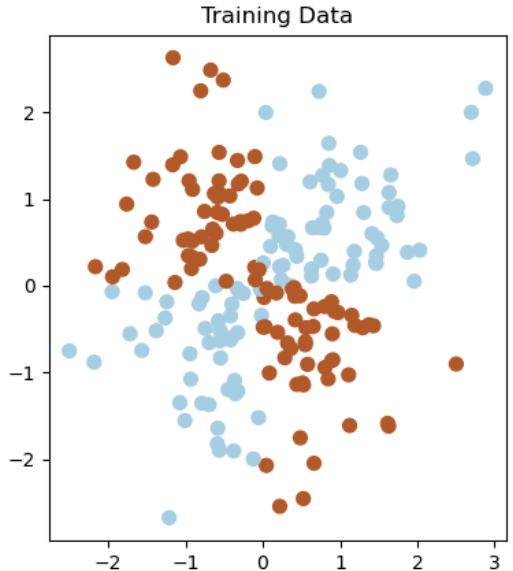
Mitchell, T. 2017; 2nd Ed. Section 2.4

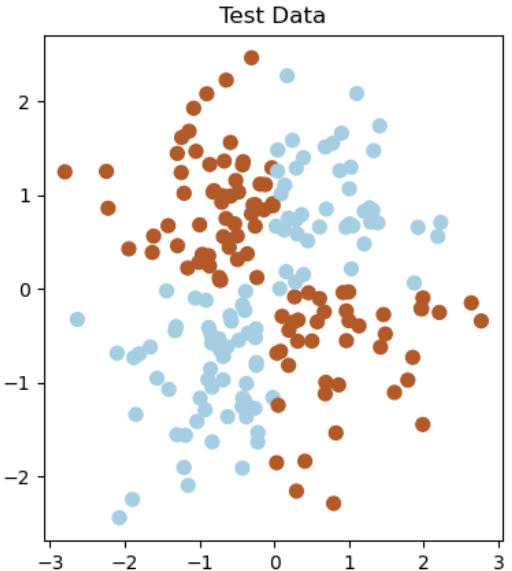
How Many Means and Variances are Needed for the Example Toy Data?





Inputs X (two dimensional)
Outputs Y (blue/brown dots)





Example: Estimated Means

$$\hat{\mu}_{i=1,k=1} = \frac{1}{\sum_j \delta(Y^j = blue)} \sum_j X_{i=1}^j \, \delta\left(Y^j = blue\right)$$

$$\hat{\mu}_{i=1,k=2} = \frac{1}{\sum_j \delta(Y^j = brown)} \sum_j X_{i=1}^j \, \delta\left(Y^j = brown\right)$$

$$\hat{\mu}_{i=2,k=1} = \frac{1}{\sum_j \delta(Y^j = blue)} \sum_j X_{i=2}^j \, \delta\left(Y^j = blue\right)$$

$$\hat{\mu}_{i=2,k=2} = \frac{1}{\sum_j \delta(Y^j = brown)} \sum_j X_{i=2}^j \, \delta\left(Y^j = brown\right)$$

Indexes: *i* refers to the attribute, *j* to the training example, and *k* to the label

Example: Estimated Variances

$$\hat{\sigma}^2{}_{i=1,k=1} = \frac{1}{\sum_j \delta(Y^j = blue)} \sum_j \left(X^j_{i=1} - \hat{\mu}_{i=1,k=1}\right)^2 \delta\left(Y^j = blue\right)$$

$$\hat{\sigma}^2{}_{i=1,k=2} = \frac{1}{\sum_j \delta(Y^j = brown)} \sum_j \left(X^j_{i=1} - \hat{\mu}_{i=1,k=1}\right)^2 \delta\left(Y^j = brown\right)$$

$$\hat{\sigma}^2{}_{i=2,k=1} = \frac{1}{\sum_j \delta(Y^j = blue)} \sum_j \left(X^j_{i=2} - \hat{\mu}_{i=2,k=1}\right)^2 \delta\left(Y^j = blue\right)$$

$$\hat{\sigma}^2{}_{i=2,k=2} = \frac{1}{\sum_j \delta(Y^j = brown)} \sum_j \left(X^j_{i=2} - \hat{\mu}_{i=2,k=2}\right)^2 \delta\left(Y^j = brown\right)$$

Indexes: *i* refers to the attribute, *j* to the training example, and *k* to the label

Parameters Obtained from Linearly Separable Toy Data

- Random variable x1:
  - means={blue: 1.4880285120376353, brown: 0.6018161084614317}
  - stdevs={blue: 0.2980038747620599, brown: 0.27387531964532413}

- Random variable x2:
  - means={blue: 1.4913689650292377, brown: 0.572709964826371}
  - stdevs={blue: 0.2730570493700515, brown: 0.2848876261616036}

Parameters Obtained from Non-Linearly Separable Toy Data

- Random variable x1:
  - means={blue: 0.2093539918826224, brown: -0.0665651623941042 3}
  - stdevs={blue: 1.0845128893988942, brown: 0.8893233116254 68}
- Random variable x2:
  - means={blue: 0.0648549992649345 3, brown: 0.0805037876743824 7}
  - stdevs={blue: 0.9896523189548784, brown: 1.0424180718504554}

Example: Probabilistic Inference

$$Y = \underset{y_k}{\operatorname{argmax}} P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

$$P(Y|X) =< P(y = blue)P(X_{i=1}|y = blue)P(X_{i=2}|y = blue),$$
$$P(y = brown)P(X_{i=1}|y = brown)P(X_{i=2}|y = brown) >$$

$$P(X_{i=1}|y = blue) = \frac{1}{\hat{\sigma}_{i=1,k=1}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_1 - \mu_{i=1,k=1}}{\hat{\sigma}_{i=1,k=1}}\right)^2}$$

$$P(X_{i=1}|y = brown) = \frac{1}{\hat{\sigma}_{i=1,k=2}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_1 - \mu_{i=1,k=2}}{\hat{\sigma}_{i=1,k=2}}\right)^2}$$

$$P(X_{i=2}|y = blue) = \frac{1}{\hat{\sigma}_{i=2,k=1}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_2 - \mu_{i=2,k=1}}{\hat{\sigma}_{i=2,k=1}}\right)^2}$$

$$P(X_{i=2}|y = brown) = \frac{1}{\hat{\sigma}_{i=2,k=2}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_2 - \mu_{i=2,k=2}}{\hat{\sigma}_{i=2,k=2}}\right)^2}$$

Note that $\sigma=\sqrt{\sigma^2}$

Example: Probabilistic Inference

$$Y = \underset{y_k}{\operatorname{argmax}} P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

$$P(Y|X) =< P(y = blue)P(X_{i=1}|y = blue)P(X_{i=2}|y = blue),$$
$$P(y = brown)P(X_{i=1}|y = brown)P(X_{i=2}|y = brown) >$$

$$P(y = blue) = 0.5$$
$$P(y = brown) = 0.5$$
$$P(X_{i=1}|y = blue) = \mathcal{N}(x_1; \mu_{i=1,k=1}, \hat{\sigma}_{i=1,k=1})$$
$$P(X_{i=1}|y = brown) = \mathcal{N}(x_1; \mu_{i=1,k=2}, \hat{\sigma}_{i=1,k=2})$$
$$P(X_{i=2}|y = blue) = \mathcal{N}(x_2; \mu_{i=2,k=1}, \hat{\sigma}_{i=2,k=1})$$
$$P(X_{i=2}|y = brown) = \mathcal{N}(x_2; \mu_{i=2,k=2}, \hat{\sigma}_{i=2,k=2})$$

Example w/Linearly Separable Data

For $X = [1.392, 1.932]$ and ground truth $y = blue$ we have

$$P(y = blue) = 0.5$$
$$P(y = brown) = 0.5$$
$$P(X_{i=1}|y = blue) = \mathcal{N}(1.3292; 1.4880, 0.2980) = 1.16153$$
$$P(X_{i=1}|y = brown) = \mathcal{N}(1.3292; 0.6018, 0.2738) = 0.04274$$
$$P(X_{i=2}|y = blue) = \mathcal{N}(1.932; 1.4913, 0.2730) = 0.39708$$
$$P(X_{i=2}|y = brown) = \mathcal{N}(1.932; 0.5727, 0.2848) = 0.00014$$

Thus,

$$P(Y|X) =< P(y = blue)P(X_{i=1}|y = blue)P(X_{i=2}|y = blue),$$
$$P(y = brown)P(X_{i=1}|y = brown)P(X_{i=2}|y = brown) >$$
$$P(Y|X) =< 0.5 * 1.16153 * 0.39708, 0.5 * 0.04274 * 0.00014 >$$
$$=<0.2306101662, 2.9918e\text{-}06>$$

← Correct prediction using unnormalised probabilities

Example w/Non-Linearly Separable Data

For $X = [1.392, -1.932]$ and ground truth $y = brown$ we have

$$P(y = blue) = 0.505$$
$$P(y = brown) = 0.495$$
$$P(X_{i=1}|y = blue) = \mathcal{N}(1.3292; 0.2093, 1.0845) = 0.21583$$
$$P(X_{i=1}|y = brown) = \mathcal{N}(1.3292; -0.0665, 0.8893) = 0.13091$$
$$P(X_{i=2}|y = blue) = \mathcal{N}(-1.932; 0.0648, 0.9896) = 0.05264$$
$$P(X_{i=2}|y = brown) = \mathcal{N}(-1.932; 0.0805, 1.0424) = 0.05935$$

Thus,

$$P(Y|X) =< P(y = blue)P(X_{i=1}|y = blue)P(X_{i=2}|y = blue),$$
$$P(y = brown)P(X_{i=1}|y = brown)P(X_{i=2}|y = brown) >$$
$$P(Y|X) =< 0.505 * 0.21583 * 0.05264, 0.495 * .13091 * .05935 >$$
$$=<0.0057374520, 0.00038459067074>$$

← Unnormalised probabilities but WRONG prediction

Today

- Preliminaries
  - Gaussian probability distributions
  - Gaussian naïve Bayes classifier
- **Gaussian Bayesian Networks (GBNs)**
  - Representation, parameter learning and inference
- Conditional Gaussian Bayesian Nets (CGBNs)
  - Representation, parameter learning and inference

Gaussian Bayesian Networks (GBNs): assumptions

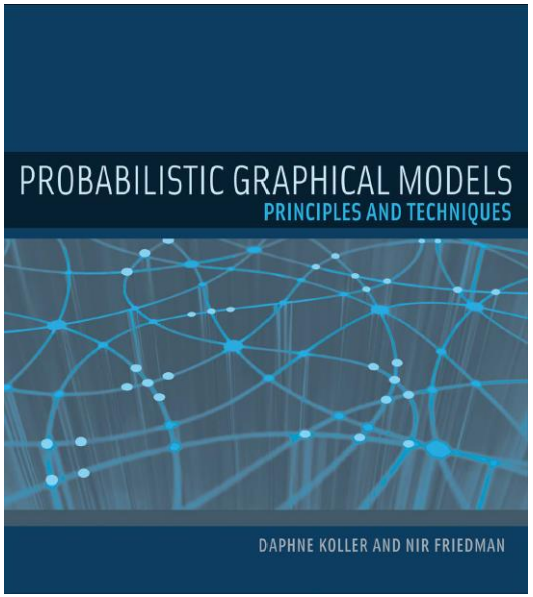
- GBNs also use directed acyclic graphs (DAGs), where every node follows a Gaussian distribution.
- Nodes without parents are described by their respective univariate Gaussian distribution.
- Nodes with parents can be expressed as a *Linear-Gaussian models* (a.k.a. linear Gaussians)—instead of multivariate Gaussians—for increased scalability.
- Each node has a variance that is specific to that node, which does not depend on the values of the parents.

Note: an implementation will be provided for GBNs – see the workshop materials of this week.

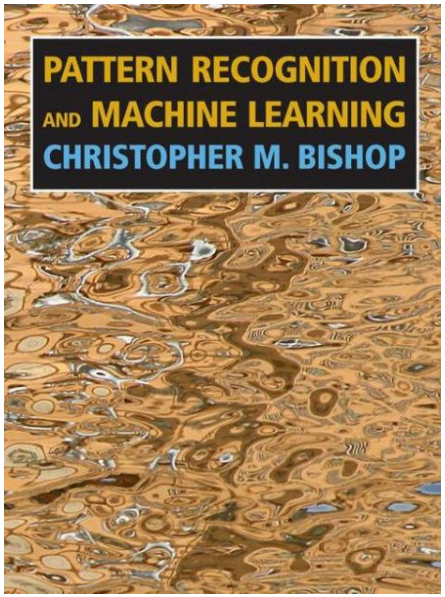
Equivalence Between Linear and Multivariate Gaussians

We will skip the demonstration of this equivalence in order to focus **linear Gaussians this week** and Multivariate Gaussians in the following lecture/workshop. But here some pointers for those inquisitive on this topic.

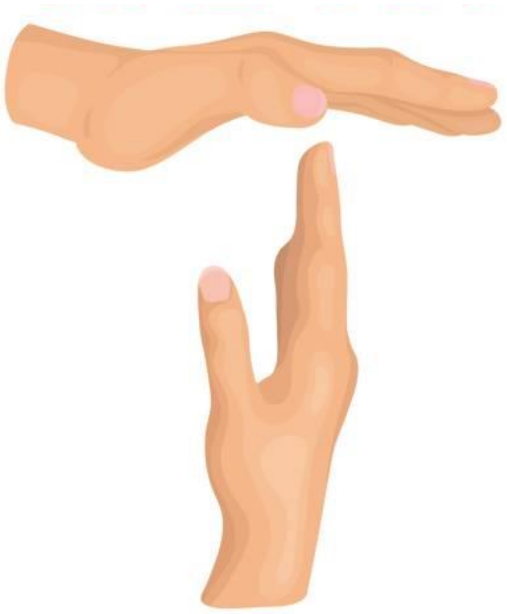
7.2 Gaussian Bayesian Networks



8.1.4 Linear-Gaussian models



Break



Linear Gaussian Models

- Let Y be a continuous random variable with continuous parents $X_1, \ldots, X_k$. A linear Gaussian can be defined as

$$P(Y|pa(Y)) = N(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_k X_k; \sigma^2)$$

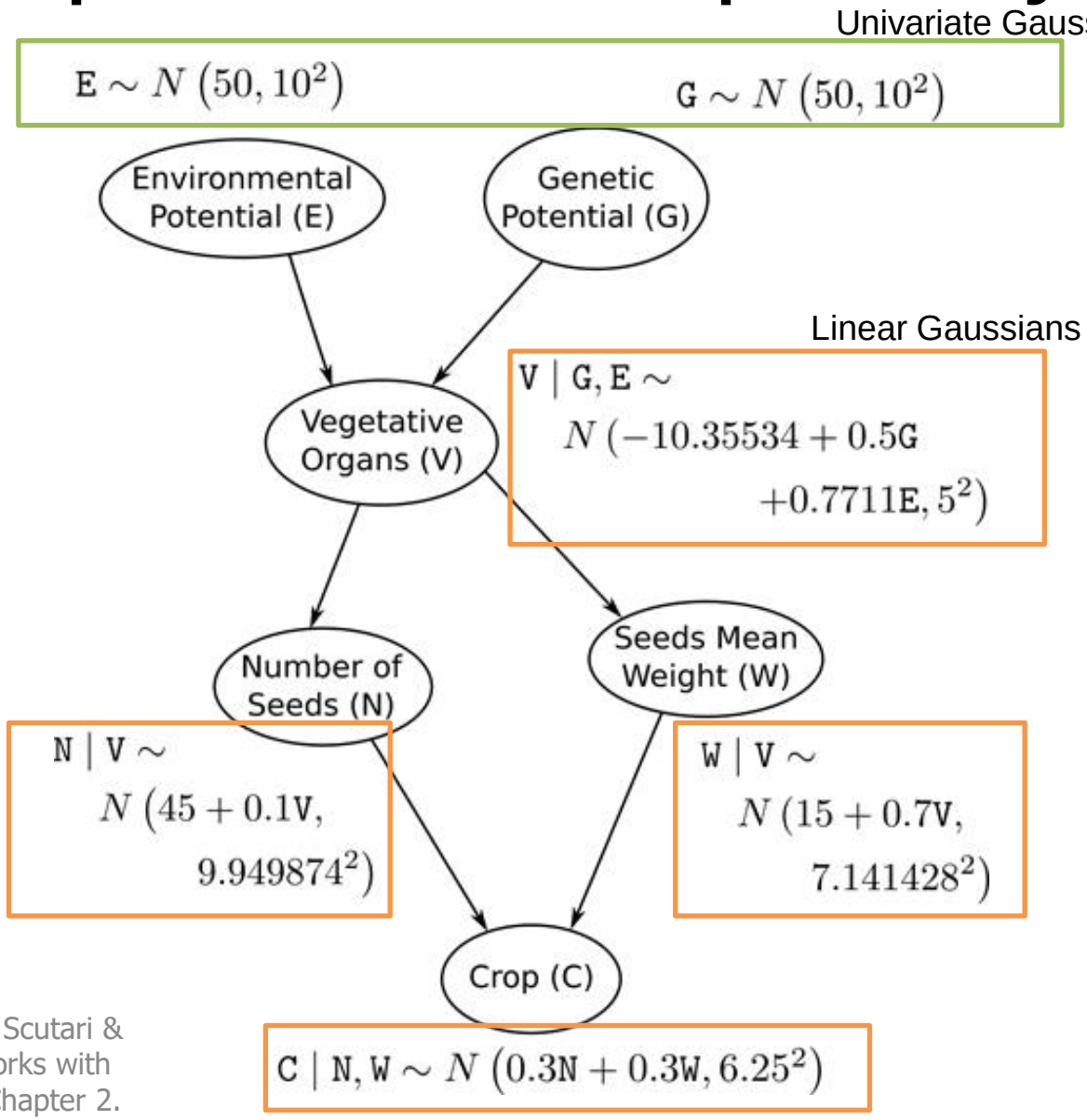
- This can also be viewed as Y being a linear function of variables $X_1, \ldots, X_k$ with the addition of a Gaussian random variable ϵ (with mean 0 and variance σ^2)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_k X_k + \epsilon$$

where β_i=coefficients and ϵ=the noise in the system.

Definition 5.14 in Koller & Friedman. Probabilistic Graphical Models, 2009. Chapter 5.

Example GBN: Crop Analysis



Example network from Scutari & Denis, Bayesian Networks with Examples in R, 2022. Chapter 2.

Example GBN: Probabilistic Representation

- Probability distributions:

$$G \sim N(50,\ 10^2)$$

$$E \sim N(50,\ 10^2)$$

$$\mathrm{V} \big| \mathrm{G} = g,\ \mathrm{E} = e \sim N(-10.35534 + 0.5g + 0.70711e,\ 5^2)$$

$$\mathrm{N} \big| \mathrm{V} = v \sim N(45 + 0.1v,\ 9.949874^2)$$

$$\mathrm{W} \big| \mathrm{V} = v \sim N(15 + 07v,\ 7.141428^2)$$

$$\mathrm{C} \big| \mathrm{N} = n,\ \mathrm{W} = w \sim N(0.3n + 0.7w,\ 6.25^2)$$

- Joint probability distribution:

$$f(G, E, V, N, W, C)$$
$$= f(G) f(E) f(V|G, E) f(N|V) f(W|V) f(C|N, W)$$

Parameter Estimation in GBNs

- Estimating means and standard deviations of univariate Gaussians can be done as in Gaussian Naïve Bayes
- Estimating the means of linear Gaussians can be done using regression methods such as RIDGE or LASSO. They have the form $C \sim \mu_C + N\ \beta_N + W\ B_W$

$$\{\hat{\beta}_{\mathrm{N}}^{\mathrm{RIDGE}}, \hat{\beta}_{\mathrm{W}}^{\mathrm{RIDGE}}\} = \underset{\beta_{\mathrm{N}},\beta_{\mathrm{W}}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (x_{i,\mathrm{C}} - \mu_{\mathrm{C}} - x_{i,\mathrm{N}}\beta_{\mathrm{N}} - x_{i,\mathrm{W}}\beta_{\mathrm{W}})^2 + \lambda_2(\beta_{\mathrm{N}}^2 + \beta_{\mathrm{W}}^2) \right\}$$

$$\left\{\hat{\beta}_{\mathrm{N}}^{\mathrm{LASSO}}, \hat{\beta}_{\mathrm{W}}^{\mathrm{LASSO}}\right\} =$$

$$= \underset{\beta_{\mathrm{N}},\beta_{\mathrm{W}}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (x_{i,\mathrm{C}} - \mu_{\mathrm{C}} - x_{i,\mathrm{N}}\beta_{\mathrm{N}} - x_{i,\mathrm{W}}\beta_{\mathrm{W}})^2 + \lambda_1(|\beta_{\mathrm{N}}| + |\beta_{\mathrm{W}}|) \right\}$$

intercept

regression coefficients

penalty parameter

Note: we will use libraries to train regressors and focus on their application to Bayes nets.

But Continuous Data can be Non-Linear

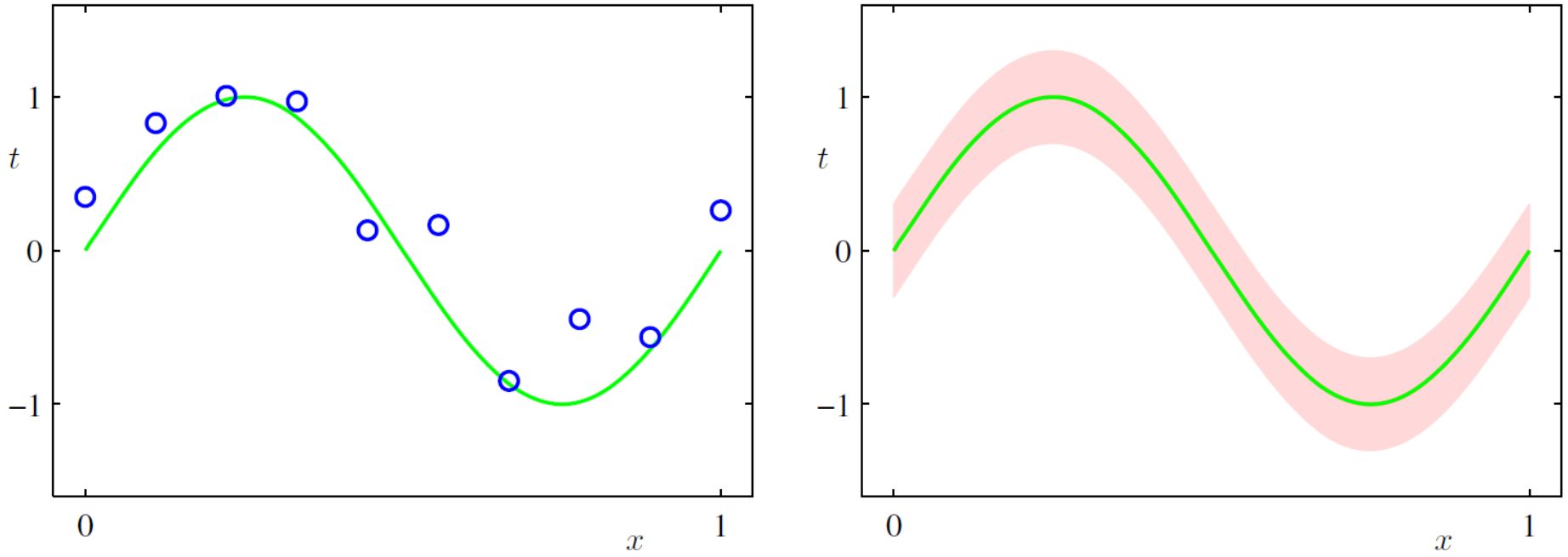


Figure A.6 The left-hand plot shows the synthetic regression data set along with the underlying sinusoidal function from which the data points were generated. The right-hand plot shows the true conditional distribution $p(t|x)$ from which the labels are generated, in which the green curve denotes the mean, and the shaded region spans one standard deviation on each side of the mean.

Image from: Bishop. C. Pattern Recognition and Machine Learning. 2006.

Linear vs. Non-Linear Gaussians

Since linear Gaussian models predict mean values via regression, they can also be predicted via non-linear regression using the kernel trick for example.

Thus,

$$Y = \underbrace{\beta_0 + \beta_1 X_1 + \beta_0 X_2 + \beta_k X_k} + \epsilon$$

This term can be estimated via linear (e.g., Ridge, Lasso) or non-linear regression (e.g., KernelRidge, Gaussian Processes).

Linear Gaussians via the Kernel Trick

Apply a feature transformation $\phi(.)$ to a higher-dimensional space instead of using the raw features:

$$C \sim \mu_C + \phi(N)\beta_N + \phi(W)B_W$$

With the kernel trick we get

$$C \sim \mu_C + \sum_{i=1}^{n} \alpha_{N\,i}\, K(N_i, N_j) + \sum_{j=1}^{m} \alpha_{W\,j}\, K(W_j, W_k),$$

number of data points

dual coefficients

kernel between training points

dual coefficients

kernel between training points

where K(.) is a kernel function such as the RBF kernel

$K(N_i, N_j) = \exp\left(\frac{(N_i - N_j)^2}{2\sigma^2}\right)$ – not the only choice.

Note: we will use libraries to do this type of regression.

Linear vs. Non-Linear Gaussians

Since linear Gaussian models predict mean values via regression, they can also be predicted via non-linear regression using a method of choice.

Thus,

$$Y = \underbrace{\beta_0 + \beta_1 X_1 + \beta_0 X_2 + \beta_k X_k}_{} + \epsilon$$

This term can be estimated via linear (such as Ridge, Lasso) or non-linear regression (e.g., KernelRidge).

***Question**: What method would you use?*

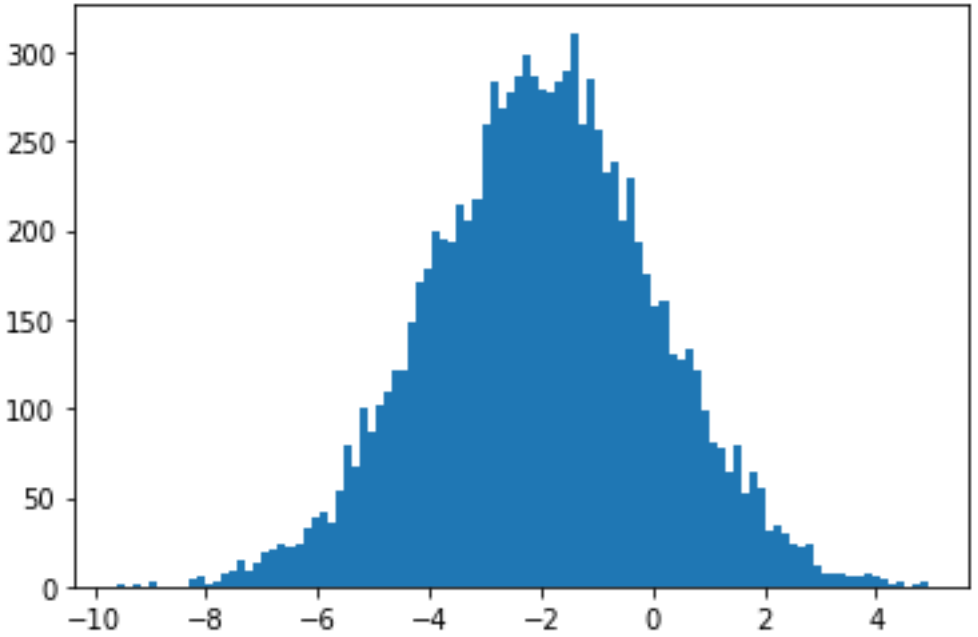
Probabilistic Inference in GBNs

- We can use the exact inference methods discussed in week 2 (inference by enumeration, variable elimination).
- We can also use the approximate inference methods discussed in week 4, but with Gaussian sampling.

```
"""
Gaussian sampling
"""
import numpy as np
import matplotlib.pyplot as plt

N = 10000
mean = -2
stdev = 2

samples = np.random.normal(mean, stdev, N)
plt.hist(samples, bins=100)
plt.show()
```



Gaussian Sampling

```python
import numpy as np

def sample_gaussian(mu, sigma):
    U1 = np.random.uniform(0, 1)
    U2 = np.random.uniform(0, 1)

    # Box-Muller Transform
    Z0 = np.sqrt(-2 * np.log(U1)) * np.cos(2 * np.pi * U2)

    return mu + sigma * Z0

mean=-2
stdev=2
N=30
samples = []
for i in range(0, N):
    sample = sample_gaussian(mean, stdev)
    print("i=%s sample=%s" % (i, sample))
    samples.append(sample)

sampled_mean = np.mean(samples)
print("sampled_mean=",sampled_mean)
```

```
(env_bnlearn) C:\Lincoln\slides\CMP9794M-2024
i=0 sample=-0.5086842775645648
i=1 sample=-1.3765755305451781
i=2 sample=-1.8747472558808422
i=3 sample=-1.0744164842867228
i=4 sample=1.6329411890404422
i=5 sample=-2.2907599731966988
i=6 sample=-4.472581014008463
i=7 sample=-2.14901494696537
i=8 sample=-3.9807721812559898
i=9 sample=-2.737736556033249
i=10 sample=-4.454112149154593
i=11 sample=-0.2050385611589336
i=12 sample=0.7635076144899493
i=13 sample=-3.5282382317028977
i=14 sample=0.7171743317224775
i=15 sample=-3.645851716031013
i=16 sample=-1.5203589365710706
i=17 sample=-6.352957829000616
i=18 sample=-4.895477764937984
i=19 sample=-3.630964708587826
i=20 sample=1.597530815181797
i=21 sample=-2.8583258544912464
i=22 sample=-1.732193209644164
i=23 sample=-0.209195973400492
i=24 sample=-2.5064337605530795
i=25 sample=-0.7747456640746142
i=26 sample=-6.81811852317821
i=27 sample=-0.11344622220511891
i=28 sample=-1.268033592557991
i=29 sample=-7.268091408767925
sampled_mean= -2.251190612510673
```

Today

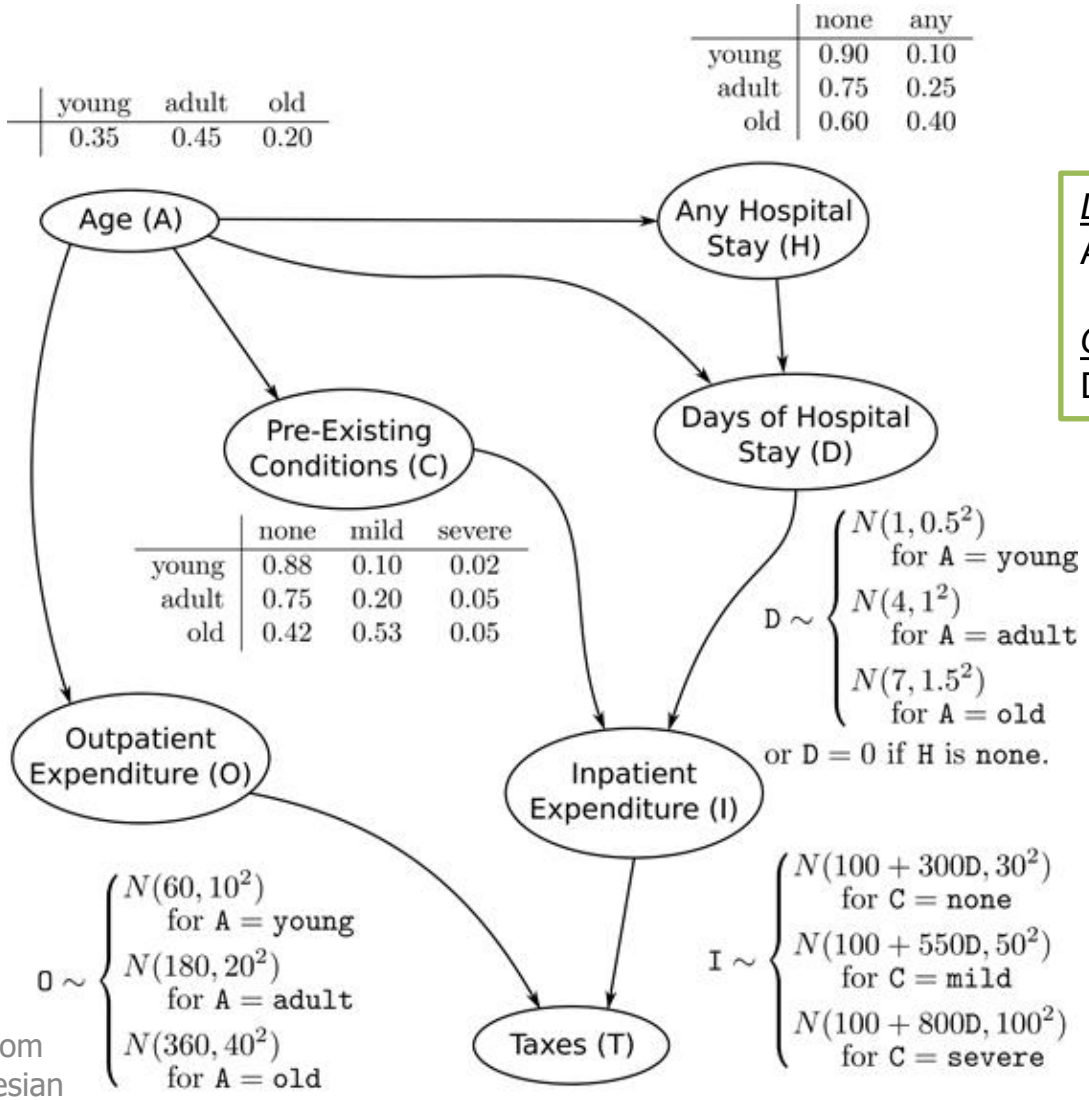
- Preliminaries
  - Gaussian probability distributions
  - Gaussian naïve Bayes classifier
- Gaussian Bayesian Networks (GBNs)
  - Representation, parameter learning and inference
- **Conditional Gaussian Bayesian Nets (CGBNs)**
  - Representation, parameter learning and inference

Conditional Gaussian Bayesian Networks (CGBNs): assumptions

- CGBNs (a.k.a. hybrid Bayesian networks) have both discrete and continuous variables/nodes
- Whilst **discrete nodes** follow multinomial distributions, **continuous nodes** follow Gaussian distributions.
- Continuous nodes with discrete parent(s) follow a mixture of Gaussian distributions (w/separate mean & variance ≡ independent set of parameters).
- Continuous nodes can have discrete/continuous parents, but discrete nodes can only have discrete parents.

Note: NO implementation will be provided for CGBNs.

Example CGBN: Healthcare Costs



Discrete nodes:
A, H, C

Continuous nodes:
D, O, I, T

Example network from Scutari & Denis, Bayesian Networks with Examples in R, 2022. Chapter 3.

Example CGBN: Probabilistic Representation

- Discrete nodes use Conditional Probability Tables (CPTs)
- Continuous nodes with discrete and continuous parents use a mixture of linear Gaussians such as

$$f(D|A,H) \sim \begin{Bmatrix} N(1,0.5^2), \text{for A} = \text{young}, \text{H} = \text{none} \\ N(4,1.0^2), \text{for A} = \text{adult}, \text{H} = \text{none} \\ N(7,1.5^2), \text{for A} = \text{old}, \text{H} = \text{none} \end{Bmatrix}$$

$$f(O|A) \sim \begin{Bmatrix} N(60,10^2), \text{for C} = \text{none} \\ N(180,20^2), \text{for C} = \text{mild} \\ N(350,40^2), \text{for C} = \text{severe} \end{Bmatrix}$$

- Linear Gaussian for continuous node w/continuous parents

$$f(T|O,I) \sim N(120 + 1.02I + 1.05O, 10^2)$$

Parameter Estimation and Probabilistic inference in CGBNs

- Estimating conditional probability tables (CPTs) can be done using MLE as discussed in weeks 1 and 2.
- Estimating the means of Gaussians can be done via linear Gaussians, estimating variances is rather trivial.
- Inducing linear Gaussians can be done via regression methods such as RIDGE, LASSO, KernelRidge, ….
- The methods discussed in weeks 2 and 4 can be used to do probabilistic inference with
  - Random sampling for discrete random variables, and
  - Gaussian sampling for continuous random variables.

Ideas for your Assignment (optional)

1. Compare different regressors for training GBNs with continuous inputs (look at ***PDF_Generator.py*** and *ModelEvaluator.py*) using learnt structures.

2. Compare discrete versus Gaussian Bayes nets—you have materials for a systematic analysis.

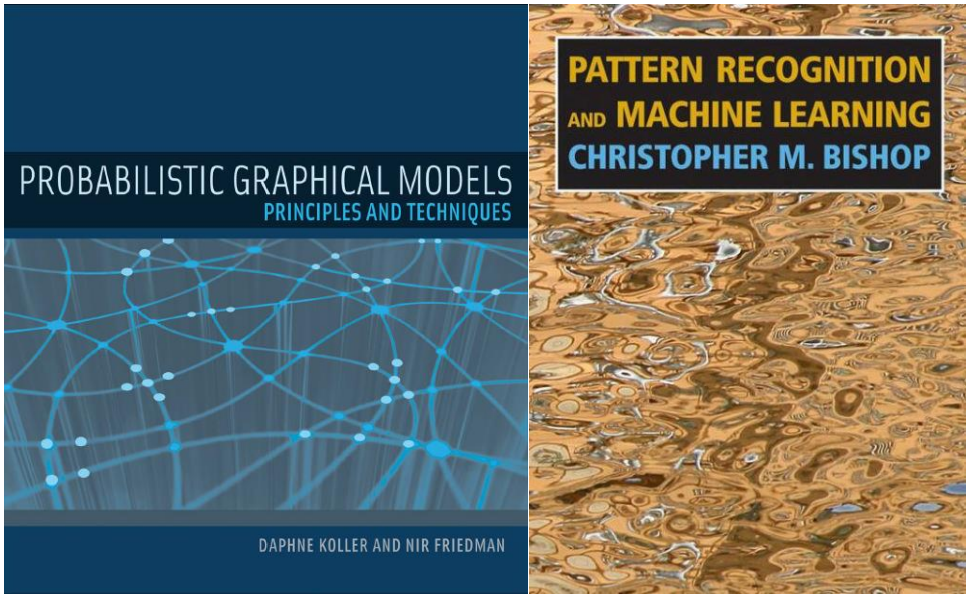
3. Implement approximate inference for GBNs using the algorithms covered in week 4. This functionality is not included in the module's code.

Today

Probabilistic reasoning with Gaussian Naïve Bayes, Gaussian Bayesian Networks (GBNs), and Conditional GBNs (hybrid Gaussian Bayes nets).

Readings:

Koller & Friedman 2009. Section 7.2
Bishop. C. 2006. Section 8.1.4



This and Next Week

Workshop (today):
Code for Gaussian Bayes nets applied to different datasets with continuous inputs & discrete outputs.

No lecture or workshop next week!!!
It is reading week, to catch up with your materials.

Lecture the week after next (w/c 28 Oct 2024):
Gaussian Processes

Any other questions?