Learning algorithms are the seeds, data is the soil, and the learned programs are the grown plants. The machine-learning expert is like a farmer, sowing the seeds, irrigating and fertilizing the soil, and keeping an eye on the health of the crop but otherwise staying out of the way.

Lecture 8
Naïve Bayes &
Review of the Basics

Haiping Lu - MLAI19

Review Preference Poll: **57**

Question 1: Multiple Answer **Avera**

Which part do you want me to review/explain in more detail in Lecture 8?

Correct	Answers	Percent Correct
☑	Bayesian regression	54.385%
☑	Pytorch & Deep learning general	45.614%
☑	PCA	56.14%
☑	K-means clustering	42.105%
☑	Autoencoder	47.368%
☑	Convolutional neural network	40.35%
☑	How to run lab 6 and 7 notebooks	19.298%

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- Naïve Bayes Classifier

Review

- Bayesian Regression
- PCA
- Autoencoder

Week 8 Contents / Objectives

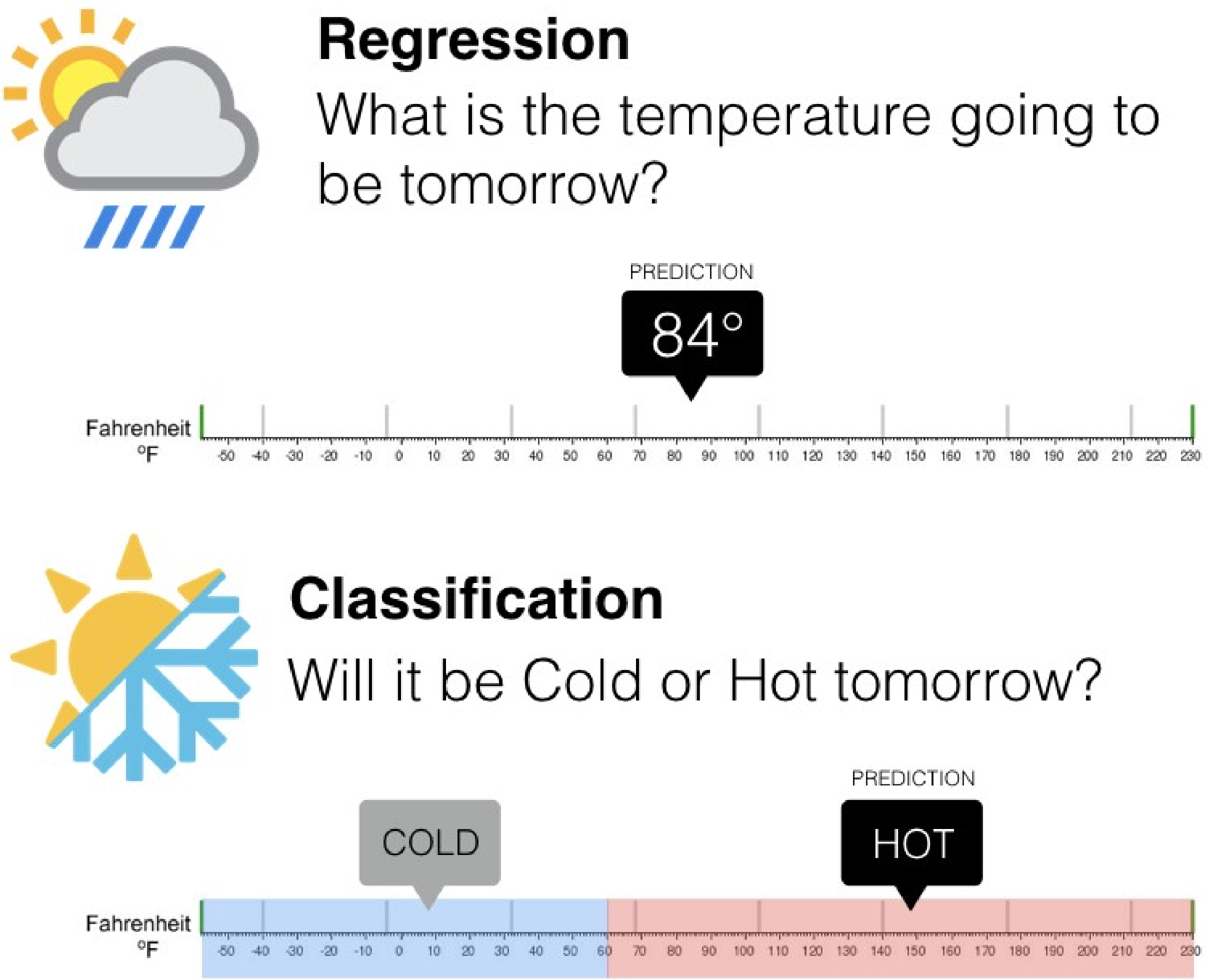
Part A

- **Probabilistic Classification**
- Naïve Bayes Classifier

Review

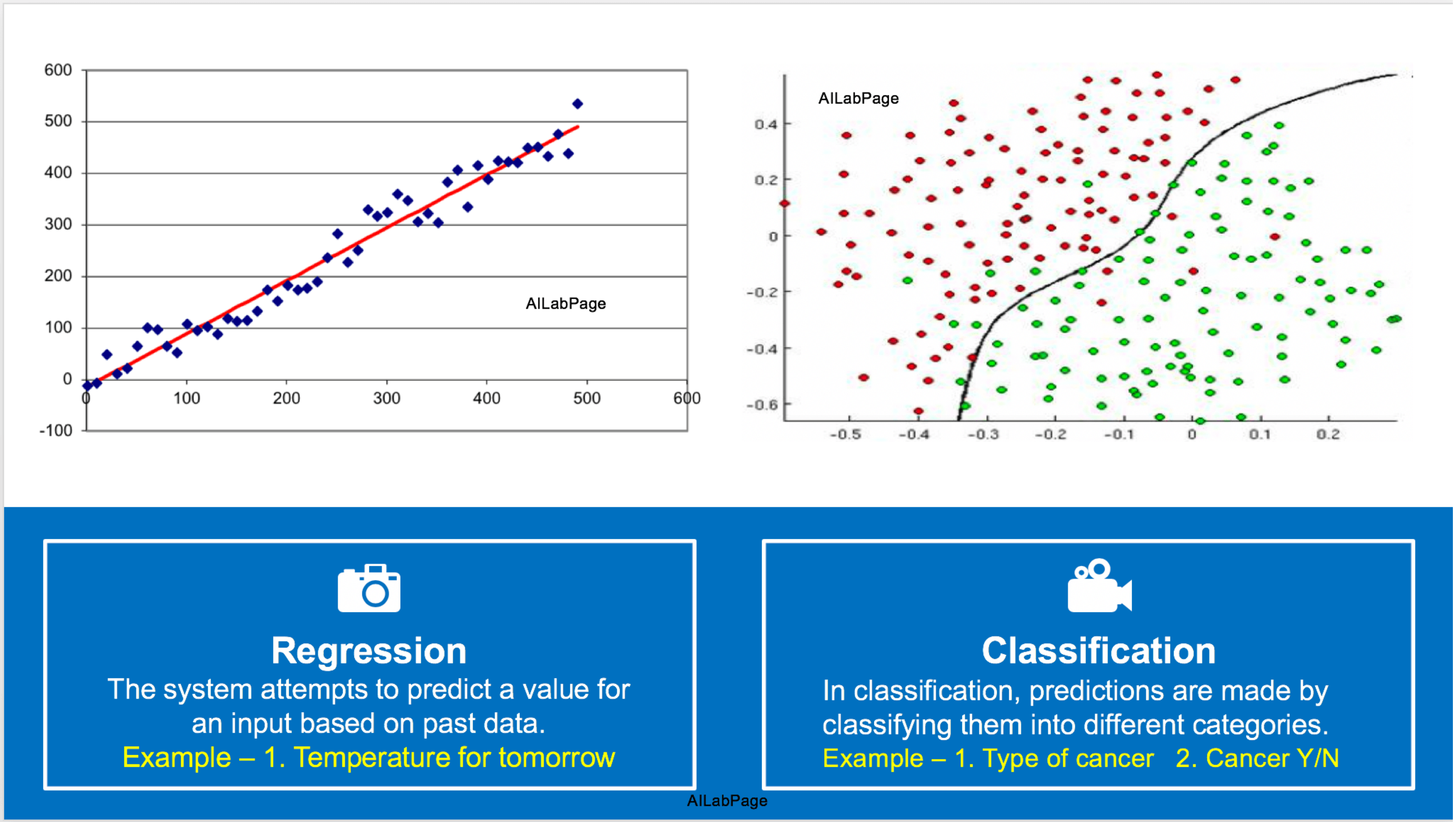
- Bayesian Regression
- PCA
- Autoencoder

Regression vs Classification



[Source: https://towardsdatascience.com/regression-or-classification-linear-or-logistic-f093e8757b9c](https://towardsdatascience.com/regression-or-classification-linear-or-logistic-f093e8757b9c)

Regression vs Classification



[Source: https://vinodsblog.com/2018/11/08/classification-and-regression-demystified-in-machine-learning/](https://vinodsblog.com/2018/11/08/classification-and-regression-demystified-in-machine-learning/)

Probabilistic Classification

- Training classifiers: estimating f: X $\rightarrow$ Y, or P(Y|X)
- **Discriminative** classifiers
  - Assume some functional form for P(Y|X)
  - Estimate parameters of P(Y|X) directly from training data
- **Generative** classifiers
  - Assume some functional form for P(X|Y), P(X)
  - Estimate parameters of P(X|Y), P(X) directly from training data
  - Use Bayes rule to calculate P(Y|X= x_i)
- **Question**: Is Bayesian regression discr/gener.?

Bayes Classifier: MAP

- Training set: p-dimensional data $\mathbf{X} = (x_1, x_2, \ldots, x_p)$ and associated class label $y \in \{C_1, C_2, \ldots, C_m\}$, i.e., there are m classes. We have n such pairs $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_n, y_n)$
- Classification is to derive the maximal $P(C_i|\mathbf{X})$
- Generative classifiers: the Maximum A Posteriori (MAP) derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Question: we can ignore P(**X**) for classification. Why?

- $P(X)$ is constant for all classes → we only need to maximize $P(\mathbf{X}|C_i)P(C_i)$, i.e., compute **1)** prior, and **2)** likelihood.

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- **Naïve Bayes Classifier**

Review

- Bayesian Regression
- PCA
- Autoencoder

Naïve Bayes Classifier

- Simplified assumption: variables are independent conditioned on the class label

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{p} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_p|C_i)$$

- Greatly reduce the #parameters & computational cost
- For categorical variables, we can estimate their probability by *counting its #occurrence divided by the total number of related samples*
- For continuous variables, we can use Gaussian to model them or *convert to categorical via splitting/binning*

Naïve Bayes Example: Data

Split into 3

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

New data *to be classified*:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Example:

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = “yes”) =

  P(buys_computer = “no”) =

- Compute $P(\mathbf{X}|C_i)$ for each class

  P(age = “<=30” | buys_computer = “yes”) =

  P(income = “medium” | buys_computer = “

  P(student = “yes” | buys_computer = “yes) =

  P(credit_rating = “fair” | buys_computer = “

  Similarly work out those for buys_computer = “no”

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(\mathbf{X}|C_i)$ **:** P(**X**|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(**X**|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(\mathbf{X}|C_i)P(C_i)$ **:** P(**X**|buys_computer = “yes”) x P(buys_computer = “yes”) = 0.028

P(**X**|buys_computer = “no”) x P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$P(\text{p34=M} \mid \text{Y}) \times P(\text{p61=M} \mid \text{Y}) \times P(\text{BMI=H} \mid \text{Y}) \times P(\text{cancer} = \text{Y})$

$P(\text{p34=M} \mid \text{N}) \times P(\text{p61=M} \mid \text{N}) \times P(\text{BMI=H} \mid \text{N}) \times P(\text{cancer} = \text{N})$

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$\boldsymbol{P(\text{p34=M} \mid \text{Y})} \times P(\text{p61=M} \mid \text{Y}) \times P(\text{BMI=H} \mid \text{Y}) \times P(\text{cancer} = \text{Y})$

$P(\text{p34=M} \mid \text{N}) \times P(\text{p61=M} \mid \text{N}) \times P(\text{BMI=H} \mid \text{N}) \times P(\text{cancer} = \text{N})$

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$P(\text{p34=M} \mid \text{Y}) \times \boldsymbol{P(\text{p61=M} \mid \text{Y})} \times P(\text{BMI=H} \mid \text{Y}) \times P(\text{cancer} = \text{Y})$

$P(\text{p34=M} \mid \text{N}) \times P(\text{p61=M} \mid \text{N}) \times P(\text{BMI=H} \mid \text{N}) \times P(\text{cancer} = \text{N})$

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$$P(\text{p34=M} \mid \text{Y}) \times P(\text{p61=M} \mid \text{Y}) \times \boldsymbol{P(\text{BMI=H} \mid \text{Y})} \times P(\text{cancer} = \text{Y})$$
$$P(\text{p34=M} \mid \text{N}) \times P(\text{p61=M} \mid \text{N}) \times P(\text{BMI=H} \mid \text{N}) \times P(\text{cancer} = \text{N})$$

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$$P(\text{p34=M} \mid \text{Y}) \times P(\text{p61=M} \mid \text{Y}) \times P(\text{BMI=H} \mid \text{Y}) \times \boldsymbol{P(\text{cancer} = \text{Y})}$$

$$P(\text{p34=M} \mid \text{N}) \times P(\text{p61=M} \mid \text{N}) \times P(\text{BMI=H} \mid \text{N}) \times P(\text{cancer} = \text{N})$$

Another Example for Naïve Bayes

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	**Y**
Medium	Low	Medium	**Y**
Low	Low	High	**Y**
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	**Y**
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	**Y**

$0.4 \times 0 \times 0.4 \times 0.5 = 0$

$0.2 \times 0.4 \times 0.2 \times 0.5 = 0.008$

In practice, we finesse the zeroes and use logs:
(note: $\log(A \times B \times C \times D \times \ldots) = \log(A)+\log(B)+ \ldots$)

New patient:
P34=M, P61=M, BMI = H

Best guess at cancer field ?

which of these gives the highest value?

P34 level	P61 level	BMI	Prostate cancer
High	Low	Medium	Y
Medium	Low	Medium	Y
Low	Low	High	Y
Low	High	Low	N
Low	Low	Low	N
Medium	Medium	Low	N
High	Low	Medium	Y
High	Medium	Low	N
Low	Low	High	N
Medium	High	High	Y

$\log(0.4) + \log(0.001) + \log(0.4) + \log(0.5) = -4.09$

$\log(0.2) + \log(0.4) + \log(0.2) + \log(0.5) = -2.09$

Numerical Stability

$\log(a \times b \times c \times \ldots) = \log(a) + \log(b) + \log(c) + \ldots$

This helps us to avoid/reduce the **underflow** errors, which we would otherwise get with when multiplying many probabilities, e.g.

0.003 × 0.000296 × 0.001 × …[**100** fields] × 0.042 …

Avoiding the Zero-Probability

- One conditional prob. = zero → predicted prob. = zero
  - Not desired
- Use **Laplacian smoothing**
  - E.g., a dataset with 1000 samples, income=low (0), income= medium (990), and income = high (10)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "smoothed" prob. estimates are close to their "unsmoothed" counterparts

Summary on Naïve Bayes

- Assumption: features independent conditioned on class label
- Advantages
  - Easy to implement and fast to compute
  - Good results obtained in many high-dim cases
- Disadvantages
  - Often dependencies exist among variables, e.g., pixels in image, …
  - **Question**: *what method* can help reduce the dependency?
- Verdict: A good baseline to start with

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- Naïve Bayes Classifier

Review

- Bayesian Regression
- PCA
- Autoencoder

Factors for deciding review topics

- Poll
- Mock quiz 2 results
- Discussion board Q&A
- Q&A in Lab

Question 1: Multiple Answer **Avera**

Which part do you want me to review/explain in more detail in Lecture 8?

Correct	Answers	Percent Correct
☑	Bayesian regression	54.385%
☑	Pytorch & Deep learning general	45.614%
☑	PCA	56.14%
☑	K-means clustering	42.105%
☑	Autoencoder	47.368%
☑	Convolutional neural network	40.35%
☑	How to run lab 6 and 7 notebooks	19.298%

Machine Learning Ingredients

- **Data**: +pre-processing (& visualisation), e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure ~ Architecture ← expert knowledge
    - Must **specify** before ML, can optimise via cross validation (CV)
  - **Hyper-parameter**, e.g., prior, #degree, layer ← knowledge
    - Must **specify** (choices) and can optimise via CV (***tuning***)
  - Parameters (theta)
    - Compute/learn parameter, e.g., **weights**, bias ← optimisation alg.
- Evaluation metric (what's best): loss/error function
- Optimisation: (how to find the best) learnable parameters

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- Naïve Bayes Classifier

Review

- **Bayesian Regression**
- PCA
- Autoencoder

Bayesian Regression vs Non-Bayes

$$p(\theta \mid y) \propto p(y \mid \theta)\, p(\theta)$$

posterior $\propto$ likelihood · prior

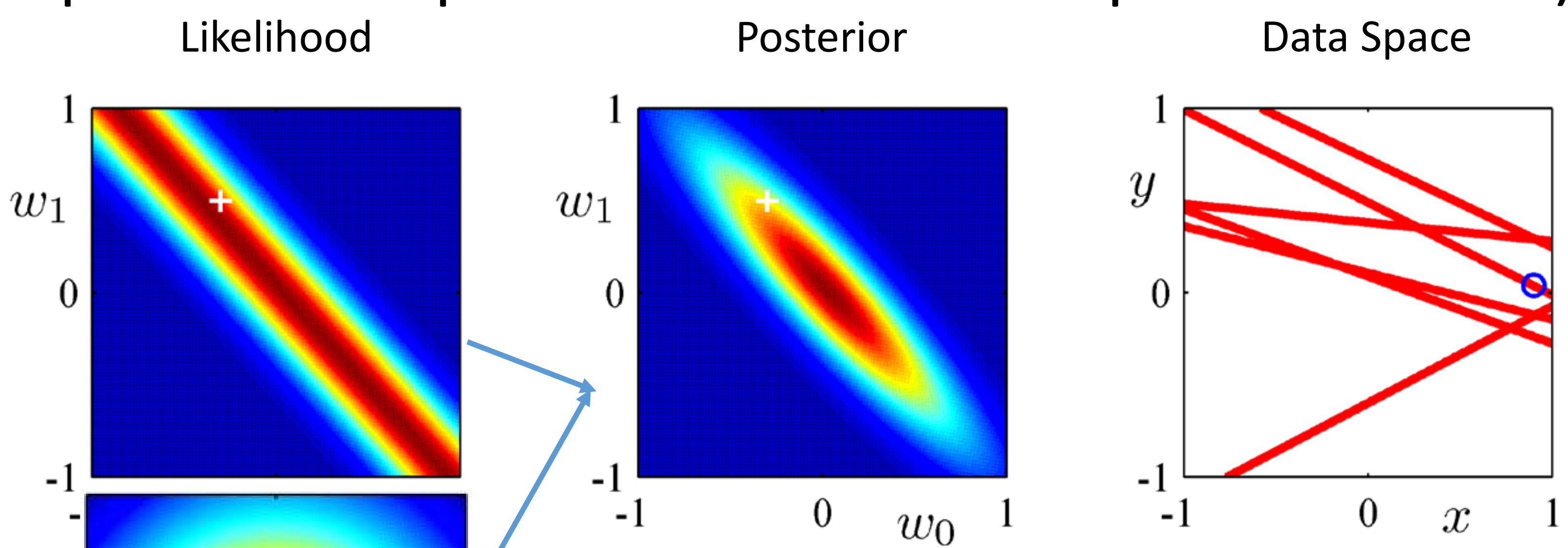
- Bayesian regression
  - Model structure: assuming given in this module
  - Place/specify prior on model parameters (weights)
    - Parameter for prior → Hyper-parameters (given or CV)
  - Compute likelihood after observing data
  - Update posterior (→new prior) and iterate
  - Metric: MSE, Max A Posterior (MAP); optim: closed/SGD
- Non-Bayesian
  - Model structure: assuming given
  - Compute likelihood after observing data
  - Metric: MSE, Max Likelihood estimation (MLE); optim: …

Bayesian Regression Ingredients

- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: basis function chosen, e.g., poly
  - **Hyper-parameter**: basis function (e.g., degree) & prior hyper
  - Parameters (theta): weights and bias
- Evaluation metric (what's best): MSE
- Optimisation: (how to find the best): closed form for Gaussian distributions, SGD etc. otherwise

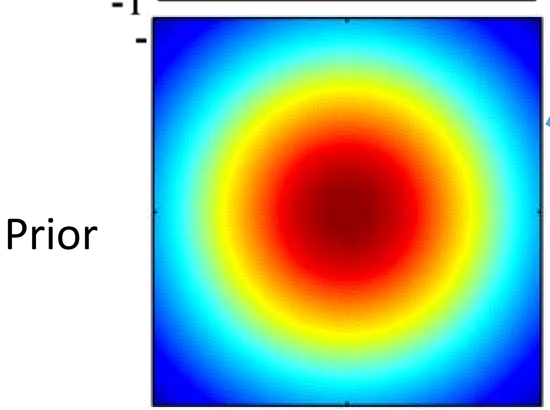
Bayesian Regression (S14 of L6)

1 data point observed → soft constraint. This posterior → prior for the next data point observed)



$y=w_1x + w_0$

(0.9, 0.1)

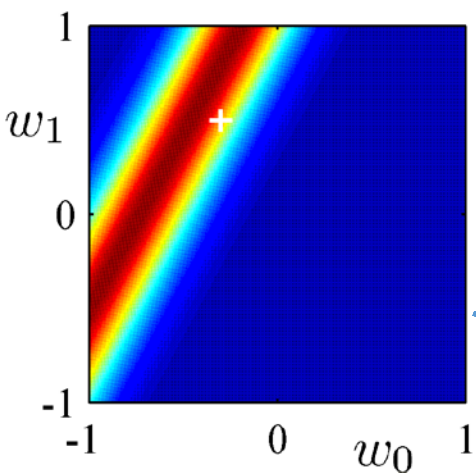


Each observed data point defines a likelihood like the top left

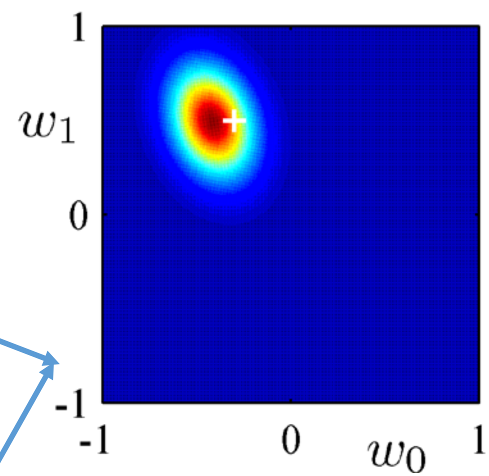
Bayesian Regression (S15 of L6)

Another data point observed

Likelihood

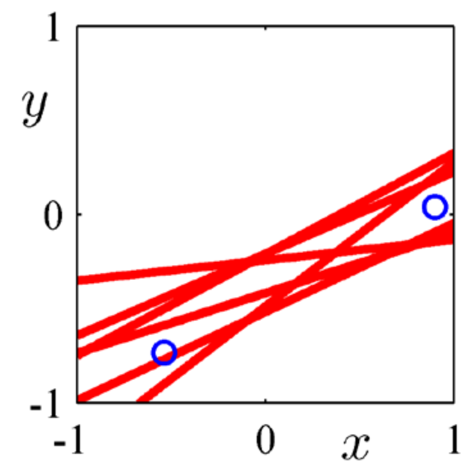


Posterior



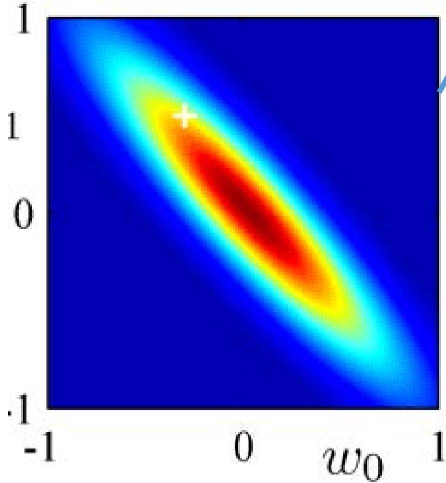
y=w₁x + w₀

Data Space



(-0.7, -0.8)

Current Prior = Previous posterior

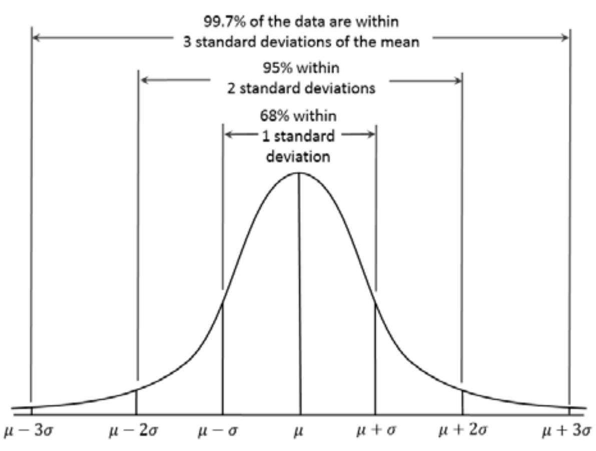


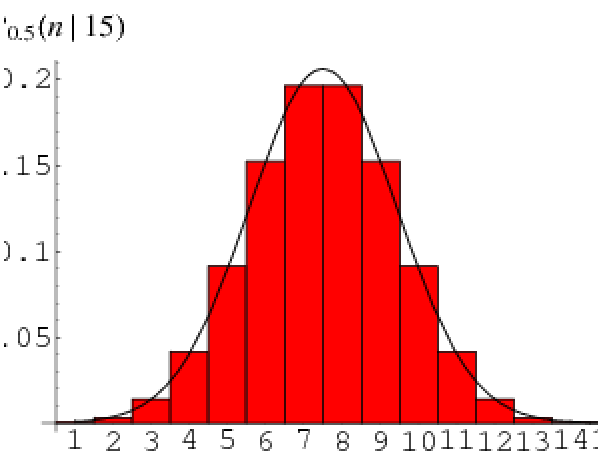
Question:

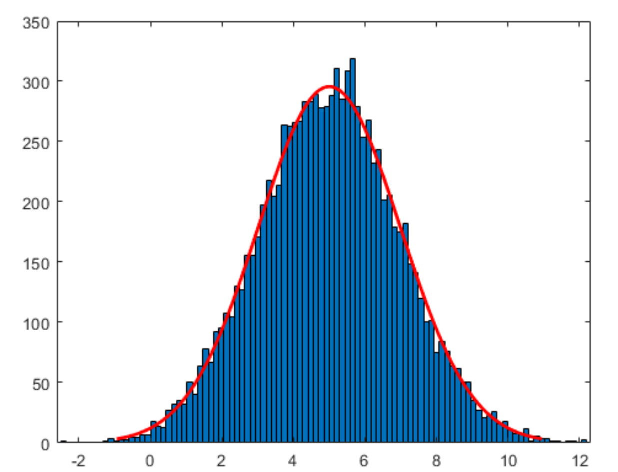
What will be the MLE solution at this point?

Bayesian Regression: Computation

- Gaussian/Normal distribution
  - Knowing the **mean and (co)variance** (std) is sufficient to specify the distribution (*sufficient statistics*)
    - Closed form solution often feasible
  - Solution
    - Density estimation: estimating mean and (co)variance
    - Optimisation: take the mode (max) → mean







Lab 6 – Main Trick

$$y = mx + c + \epsilon$$

$$c \sim \mathcal{N}(0, \alpha_1)$$

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

$$p(c) = \frac{1}{\sqrt{2\pi\alpha_1}} \exp\left(-\frac{1}{2\alpha_1}c^2\right)$$

$$p(\mathbf{y}|\mathbf{x}, c, m, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - mx_i - c)^2\right)$$

$$p(c|\mathbf{y}, \mathbf{x}, m, \sigma^2) = \frac{p(\mathbf{y}|\mathbf{x}, c, m, \sigma^2)p(c)}{p(\mathbf{y}|\mathbf{x}, m, \sigma^2)} = \frac{p(\mathbf{y}|\mathbf{x}, c, m, \sigma^2)p(c)}{\int p(\mathbf{y}|\mathbf{x}, c, m, \sigma^2)p(c)\mathrm{d}c}$$

$$= \frac{1}{\sqrt{(2\pi\tau^2)}} \exp\left(-\frac{1}{2\tau^2}(c-\mu)^2\right)$$

$$p(c|\mathbf{y}, \mathbf{x}, m, \sigma^2) \propto p(\mathbf{y}|\mathbf{x}, c, m, \sigma^2)p(c)$$

$$\log p(c|\mathbf{y}, \mathbf{x}, m, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - c - mx_i)^2 - \frac{1}{2\alpha_1}c^2 + \text{const}$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - mx_i)^2 - \left(\frac{n}{2\sigma^2} + \frac{1}{2\alpha_1}\right)c^2 + c\frac{\sum_{i=1}^{n}(y_i - mx_i)}{\sigma^2},$$

$$\log p(c|\mathbf{y}, \mathbf{x}, m, \sigma^2) = -\frac{1}{2\tau^2}(c-\mu)^2 + \text{const},$$

$$\tau^2 = \left(n\sigma^{-2} + \alpha_1^{-1}\right)^{-1} \qquad \mu = \frac{\tau^2}{\sigma^2}\sum_{i=1}^{n}(y_i - mx_i)$$

Bayesian Regression: Benefits

- Key benefits
  - Enable taking prior knowledge into account
  - Provide uncertainty estimation, predicting an output distribution with mean and **variance**
- Limitation
  - If prior is wrong, …
  - Complexity
- More depth (derivations for multivariate, etc.)? https://www.youtube.com/watch?v=dtkGq9tdYcI

ML 10.1-10.7 from
https://www.youtube.com/watch?v=yDLKJtOVx5c&list=PLD0F06AA0D2E8FFBA

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- Naïve Bayes Classifier

Review

- Bayesian Regression
- **PCA**
- Autoencoder

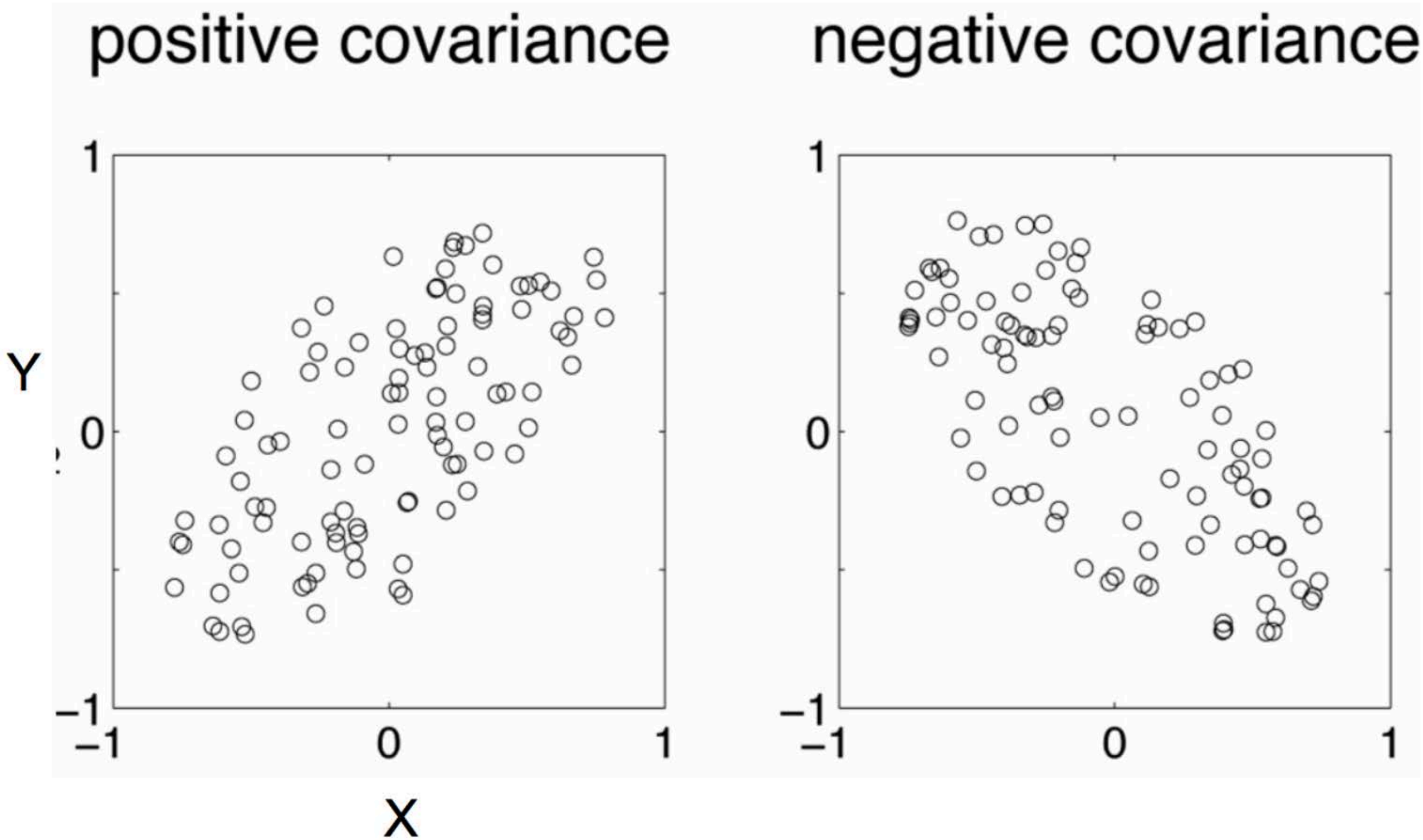
PCA Ingredients

- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: linear projection $\mathbf{y}=\mathbf{U}^T\mathbf{x}$
  - **Hyper-parameter**: lower dimension k
  - Parameters (theta): $\mathbf{U}$
- Evaluation metric (what's best): max variance
- Optimisation: (how to find the best) eigen-decomposition

Covariance

- Variance and Covariance:
  - Measure of the "spread" of a set of points around their *center of mass* (mean) (→ similar measurement as *k*-means)
- Variance (**scalar**):
  - Measure of the deviation from the mean for points in **one dimension**
- Covariance (**matrix**):
  - Measure of how much each of the dimensions vary from the mean with **respect to each other (~ correlations)**

- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between the two dimensions**
- **Covariance between one dimension is the variance**



Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

- Used to find relationships between dimensions in high dimensional data sets
- Scatter matrix: sample-based estimation of covariance matrix

$$q_{jk} = \frac{1}{N} \sum_{i=1}^{N} \left(X_{ij} - E(X_j)\right)\left(X_{ik} - E(X_k)\right)$$

The Sample mean

- Uncorrelated variables → covariance = 0
- Diagonal cov mat → all variables are uncorrelated

Form correlated from original by rotating the data space using rotation matrix $\mathbf{R}$.

$$p(\mathbf{y}) = \frac{1}{|2\pi\mathbf{D}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top\mathbf{D}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)$$

Multiply the data by a rotation matrix $\mathbf{R}$

$$p(\mathbf{y}) = \frac{1}{|2\pi\mathbf{D}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\mathbf{R}^\top\mathbf{y}-\mathbf{R}^\top\boldsymbol{\mu})^\top\mathbf{D}^{-1}(\mathbf{R}^\top\mathbf{y}-\mathbf{R}^\top\boldsymbol{\mu})\right)$$

Collect $\mathbf{R}$ to the left and right of $\mathbf{D}$

$$p(\mathbf{y}) = \frac{1}{|2\pi\mathbf{D}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top\mathbf{R}\mathbf{D}^{-1}\mathbf{R}^\top(\mathbf{y}-\boldsymbol{\mu})\right)$$

Let

$$\mathbf{C}^{-1} = \mathbf{R}\mathbf{D}^{-1}\mathbf{R}^\top$$

Rewritten in typical Guassian form

$$p(\mathbf{y}) = \frac{1}{|2\pi\mathbf{C}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top\mathbf{C}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)$$

This gives a **covariance matrix** for correlated variables (the general case):

$$\mathbf{C} = \mathbf{R}\mathbf{D}\mathbf{R}^\top$$

Note $|\mathbf{C}| = |\mathbf{D}|$, see Determinant of Matrix Product

PCA

Input: $\mathbf{x} \in \mathbb{R}^D$: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Set of basis vectors: $\mathbf{u}_1, \ldots, \mathbf{u}_K$

Summarize a *D* dimensional vector *X* with *K* dimensional feature vector *h(x)*

$$h(\mathbf{x}) = \begin{bmatrix} \mathbf{u}_1 \cdot \mathbf{x} \\ \mathbf{u}_2 \cdot \mathbf{x} \\ \cdots \\ \mathbf{u}_K \cdot \mathbf{x} \end{bmatrix}$$

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

Basis vectors are orthonormal

$$\mathbf{u}_i^T \mathbf{u}_j = 0$$

$$||\mathbf{u}_j|| = 1$$

Lagrangian for PCA Solution

- Scatter mat for I/P $$\mathbf{S} = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu} \right) \left(\mathbf{x}^{(i)} - \boldsymbol{\mu} \right)^{\top}$$
- **Question**: what is the scatter mat for the projections?
- Find the first direction via (unit-norm) constrained optimisation, using Lagrange multipliers:

$$L\left(\mathbf{u}_1, \lambda_1\right) = \mathbf{u}_1^{\top} \mathbf{S} \mathbf{u}_1 + \lambda_1 \left(1 - \mathbf{u}_1^{\top} \mathbf{u}_1\right)$$

- Gradient w.r.t. $\mathbf{u}_1$: $$\frac{\mathrm{d}L\left(\mathbf{u}_1, \lambda_1\right)}{\mathrm{d}\mathbf{u}_1} = 2\mathbf{S}\mathbf{u}_1 - 2\lambda_1 \mathbf{u}_1$$
- Set to 0 and rearrange: $\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$
  - Eigenvalue problem, physical meanings of eigenvalues

Week 8 Contents / Objectives

Part A

- Probabilistic Classification
- Naïve Bayes Classifier

Review

- Bayesian Regression
- PCA
- **Autoencoder**

Convolutional Autoencoder

```
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            # 1 input image channel, 16 output channel, 3x3 square convolution
            nn.Conv2d(1, 16, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(16, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, 7)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 7),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()  #to range [0, 1]
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Autoencoder Ingredients

- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: layers defined in nn.module
  - **Hyper-parameter**: layer specs, e.g., #channels, kernel size
  - Parameters (theta): layer weights and biases
- Evaluation metric (what's best): MSE or other
- Optimisation: (how to find the best) backprop

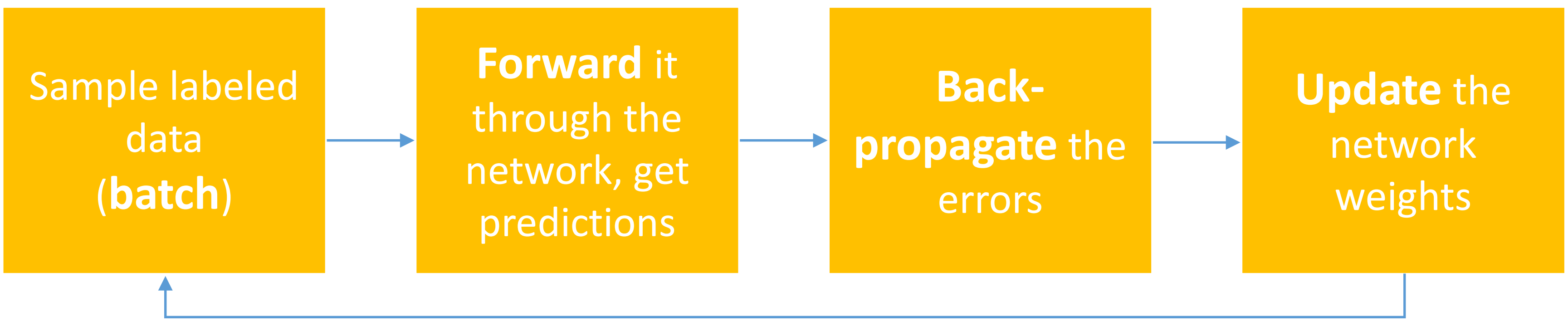
Autoencoder Training

```
#Hyperparameters for training
batch_size=64
learning_rate=1e-3
max_epochs = 20

#Set the random seed for reproducibility
torch.manual_seed(509)
#Choose mean square error loss
criterion = nn.MSELoss()
#Choose the Adam optimiser
optimizer = torch.optim.Adam(myAE.parameters(), lr=learning_rate, weight_decay=1e-5)
#Specify how the data will be loaded in batches (with random shffling)
train_loader = torch.utils.data.DataLoader(mnist_data, batch_size=batch_size, shuffle=True)
#Storage
outputs = []

#Start training
for epoch in range(max_epochs):
    for data in train_loader:
        img, label = data
        optimizer.zero_grad()
        recon = myAE(img)
        loss = criterion(recon, img)
        loss.backward()
        optimizer.step()
    if (epoch % 3) == 0:
        print('Epoch:{}, Loss:{:.4f}'.format(epoch+1, float(loss)))
    outputs.append((epoch, img, recon),)
```

Training



Data → Model → Metric → Optimisation

Machine Learning Ingredients

- **Data**: +pre-processing (& visualisation), e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure ~ Architecture ← expert knowledge
    - Must **specify** before ML, can optimise via cross validation (CV)
  - **Hyper-parameter**, e.g., prior, #degree, layer ← knowledge
    - Must **specify** (choices) and can optimise via CV (***tuning***)
  - Parameters (theta)
    - Compute/learn parameter, e.g., **weights**, bias ← optimisation alg.
- Evaluation metric (what's best): loss/error function
- Optimisation: (how to find the best) learnable parameters

Acknowledgement

- Part A used materials from: Jonathan Huang, David Corne, Tom Mitchell, Jiawei Han, Micheline Kamber, and Jian Pei

- Review used additional materials (addition to Lecture 6/7 materials) from: Fereshteh Sadeghi