Lecture 4

Cezar Ionescu

08/06 DEPARTMENT FOR CONTINUING EDUCATION

UNIVERSITY OF OXFORD

Administrative

- Homework from 25/05/2019 due now!
- Please complete and hand in the declarations of authorship.

Questions?

Unsupervised learning

- we have a sequence of data points that fall into k classes
- example:
  - images of cats and dogs
  - images of possibly malfunctioning engines
  - images of handwritten digits
- we want the learning program to determine the k classes by itself

Formalisation of unsupervised learning

- `Task = Image -> {1, 2, ..., k}`

Formalisation of unsupervised learning

- `Task = Image -> {1, 2, ..., k}`
- `Experience = List(Image)`

Formalisation of unsupervised learning

- `Task = Image -> {1, 2, ..., k}`
- `Experience = List(Image)`
- `perf : (Task, List(Image)) → ℝ`

Formalisation of unsupervised learning

- `Task = Image -> {1, 2, ..., k}`
- `Experience = List(Image)`
- `perf : (Task, List(Image)) → ℝ`
-

  ```
  perf(t, [img₁, ..., imgₙ]) = sum [correct(t, img₁), ..., corr
  ```

  - `correct : (Task, List(Image)) → {1, 2, ..., k}` is “hypothetical”

Exemplars

The data usually comes in the form of tuples of fixed length (the **features**):

$x = (x_1, \ldots, x_m) \in (\Omega_1, \ldots, \Omega_m),\ \Omega_i \subseteq \mathbb{R}$

We usually assume the classification is based on “ideal” elements:

$\xi = (\xi_1, \ldots, \xi_m) \in (\Omega_1, \ldots, \Omega_m),\ \Omega_i \subseteq \mathbb{R}$

The data is a distortion of the ideal exemplars, e.g.

$x = (x_1, \ldots, x_m) = (\xi_1 + \varepsilon_1, \ldots, \xi_m + \varepsilon_m)$

where ε_j is a random variable with mean 0.

Thus, if we only had one exemplar, we could reconstruct it by estimating the mean value of the data.

Expectations

Consider a coin toss experiment:

```
Ω = {H, T}
p : Event → [0, 1]
p(H) = α
```

What is the expected value of the result?

Expectations

Now consider a bet: `h` \$ for heads, `t` \$ for tails. The expected value is

Expectations

Now consider a bet: `h` \$ for heads, `t` \$ for tails. The expected value is

```
h * α + t * (1 - α)
```

Remarks

- `h` and `t` are *not* possible results of the random experiment, which consists of flipping a coin.
- We can construct a "derived" probability space as follows:

Remarks

- `h` and `t` are *not* possible results of the random experiment, which consists of flipping a coin.
- We can construct a “derived” probability space as follows:
- `Ω' = {h, t}`
- `p' : Event' → [0, 1]`
  - `p'(h) = p(H)`
  - `p'(t) = p(T)`

Remarks

- `h` and `t` are *not* possible results of the random experiment, which consists of flipping a coin.
- We can construct a "derived" probability space as follows:
- `Ω' = {h, t}`
- `p' : Event' → [0, 1]`
    - `p'(h) = p(H)`
    - `p'(t) = p(T)`
- Even if we consider `Ω'`, it is still the case that the expected value is *not* a possible result of the experiment.

Random variable

Definition: Given `Ω, Event, p`. A **random variable** is a function `X : Ω → ℝ`.

Probability theory and statistics are largely the study of random variables.

Expected value

Definition: Given `Ω`, `Event`, `p` and a random variable `X`. Assume that `Ω` is finite: `Ω = {ω₁, ω₂, ..., ωₙ}`. The **mean** or **expected value** of `X` is

```
E(X) = X(ω₁) * p(ω₁) + X(ω₂) * p(ω₂) + ... + X(ωₙ) * p(ωₙ)
```

Intuition: center of mass.

Classical statistics versus machine learning

The expected value minimises the guessing error.

This is not always what we want: cf., penalty shots.

Exercise

Consider a card game played with a full deck, in which aces have a value of `11`, jacks `12`, queens `13`, kings `14`, and all other cards a value of `0`. We draw a card at random from the deck. What is its expected value? Define `Ω, Event, p` and the random variable whose expected value you are computing.

Standard deviation

Definition: Consider `Ω = {ω₁, ω₂, ..., ωₙ}, Event, p, X : Ω → ℝ`.
Let `E(X) = μ` The **standard deviation** of `X` is

```
std_dev(X) = √ ((X(ω₁) - μ)² * p(ω₁) + ... (X(ωₙ) - μ)² * p(ωₙ))
```

The **variance** of `X` is `std_dev(X)²`

Exercise

Compute the standard deviation of the random variable defined for the card game exercise.

Notation

In the following, when no ambiguities arise, we use μ to denote `E(X)` and σ for `std_dev(X)`

Chebyshev's theorem

Let `Ω, Event, p` and `X` as above. Let

```
Farₖ = {ω ∈ Ω | ||X(ω) - μ|| > k * σ}
```

Then `p(Farₖ) ≤ 1/k²`.

Interpretation of Chebyshev's theorem

`Ω`, `Event`, `p` and `X` as above. If we draw a random element from `Ω`, then

- it will fall in the interval `[μ - 2 * σ, μ + 2 * σ]` with probability at least `0.75`
- it will fall in the interval `[μ - 3 * σ, μ + 3 * σ]` with probability at least `0.889`
- it will fall in the interval `[μ - 4 * σ, μ + 4 * σ]` with probability at least `0.938`

Example

The mean price of houses in a certain neighbourhood is `400000` $, and the standard deviation is `80000` $. Find the price range in which `75%` of the houses will sell. (Bluman, Chapter 3)

Solution: From Chebyshev's theorem, we know that at least `75%` of the houses are within the interval `[μ - 2 * σ, μ + 2 * σ]`. Therefore, the interval is `[240000, 560000]`.

The normal distribution

```
pdf(x) = (1/√(2 * π * σ²)) * exp(- (x - μ)²/(2 * σ²))
```

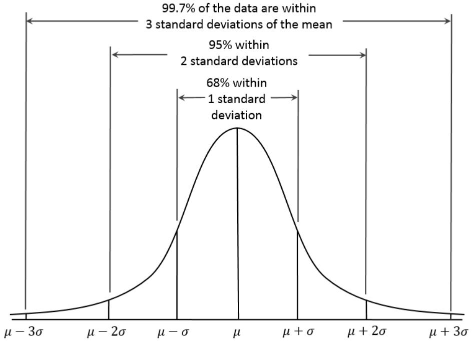

99.7% of the data are within 3 standard deviations of the mean
95% within 2 standard deviations
68% within 1 standard deviation
$\mu - 3\sigma$
$\mu - 2\sigma$
$\mu - \sigma$
μ
$\mu + \sigma$
$\mu + 2\sigma$
$\mu + 3\sigma$

Figure 1: Normal distribution[1]

[1]Figure by Dan Kernler - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=36506025

Example

Consider the neighbourhood from the previous example, where the mean price of houses is `400000` \$, and the standard deviation is `80000` \$. Find the price range in which `75%` of the houses will sell, but now assuming that *the prices are normally distributed*.

Solution: We use the Python function `scipy.stats.normal.interval`:

```
import scipy.stats
scipy.stats.norm.interval(0.75, 400000, 80000)
```

We obtain `[307972, 492028]`. From Chebyshev's theorem, we had the much larger interval `[240000, 560000]`.

The central limit theorem

`Ω, Event, p` and `X` as above. Consider the following experiment: `n` elements `e₁, ..., eₙ` are drawn independently from `Ω` and we compute the mean value of `X` for this sample `μₛ = (X(e₁) + ... + X(eₙ))/n`. Then `μₛ` is a random variable whose probability distribution approaches with increasing `n` the normal distribution with mean `μ` and standard deviation `σ/√(n)`.

Homework

The theorem states that `μₛ` is a random variable. But a random variable is a function defined in the context of a probability space. What is that probability space here? You will need to specify `Ω', Event', p' : Event' → [0, 1]` and define `μₛ : Ω' → ℝ` as a function in terms of in terms of the given `Omega, Event, p`, and `X`.

Quality of approximation

- If the random variable `X` is itself normally distributed, then the approximation in the central limit theorem is good even for small sample sizes `n`.
- If the random variable `X` is not normally distributed, the approximation requires larger `n`. In many practical applications, a value `n ≥ 30` will be adequate.

Example

The Nielsen agency reported that children between 2 and 5 watch an average of 25 hours of TV per week. Assuming a normal distribution with σ = 3 hours. If 20 children are randomly selected, find the probability that the average TV watching time in a week will be μₛ > 26.3 hours. (Bluman, Chapter 6)

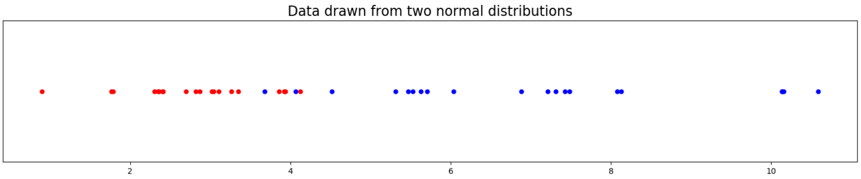
Solution: The sample average μₛ is a random variable that is approximately normally distributed, with mean 25 and standard deviation 3/√(20). We need to find the probability that the value of μₛ is bigger than 26.3. We use the Python function `scipy.stats.norm.cdf` to find the probability that μₛ is smaller or equal to 26.3, and subtract from unity:

```
import scipy.stats
1 - scipy.stats.norm.cdf(26.3, 25, 3/scipy.sqrt(20))
```

We obtain 0.026316151282870237, or approximately 2.6%.

EM algorithm

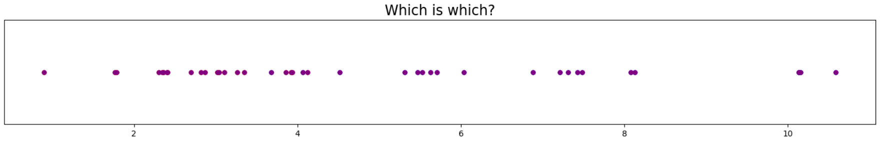
Example: Data generated from `normal(3, 0.8)` (red) and `normal(7, 2)` (blue).



- `20` dots of each colour
- the **sample** means and standard deviations are `2.8, 0.8` red, `6.97, 2.0` for blue.

EM algorithm

What if we do not know which points came from which distribution?



EM algorithm

We are looking for $\mu_1, \sigma_1, \mu_2, \sigma_2$.

- hypothesis space: $H = (\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
- data: $d = (x_1, ..., x_n)$ drawn from the two probability distributions.

We want $h_{map} \in H$

$h_{map} = \text{argmax } p(h \mid d)$

If we have a uniform prior on H, then this is equivalent to h_{ml}:

$h_{ml} = \text{argmax } p(d \mid h)$

EM algorithm

```
   p(d | h)
=
   p(x₁, ..., xₙ | h)
=
   p(x₁ | h) * ... * p(xₙ | h)
```

EM algorithm

```
   ln p(d | h)
=
   ln (p(x₁ | h) * ... * p(xₙ | h))
=
   ln p(x₁ | h) + ... + ln p(xₙ | h)
```

EM algorithm

If we know that `xᵢ` has been drawn from distribution `j`, we can compute

```
p(xᵢ | h) = pdf(xᵢ, μⱼ, σⱼ) = (1/√(2 * π * σⱼ²)) *
                                exp(- (x - μⱼ)²/(2 * σⱼ²))
```

EM algorithm

Introduce the "hidden data":

- data: $d = ((x_1, z_{11}, z_{12}) \ldots, (x_n, z_{n1}, z_{n2}))$, where $z_{ij} \in \{0, 1\}$ is 1, if x_i has been drawn from distribution j, and 0 otherwise.

$$
\begin{aligned}
p(x_i, z_{i1}, z_{i2} \mid h) = {} & (1/\sqrt{(2 * \pi * \sigma_1^2)}) * \\
& \exp(- z_{i1} * (x - \mu_1)^2/(2 * \sigma_1^2)) + \\
& (1/\sqrt{(2 * \pi * \sigma_2^2)}) * \\
& \exp(- z_{i2} * (x - \mu_2)^2/(2 * \sigma_2^2))
\end{aligned}
$$

EM algorithm

The data has become a random variable, hence we cannot maximise `ln p(d | h)`. Instead, we maximise

```
E (ln p(d | h))
```

EM algorithm

$$E(\ln p(d \mid h))$$

$$= E(\ln p(x_1, z_{11}, z_{12} \mid h) + \ldots + \ln p(x_n, z_{n1}, z_{n2} \mid h))$$

$$= E(\ln p(x_1, z_{11}, z_{12} \mid h) + \ldots + E(\ln p(x_n, z_{n1}, z_{n2} \mid h)$$

EM algorithm

$$E\ (\ln p(x_i,\ z_{i1},\ z_{i2}\ |\ h)$$

$$=$$

$$E\ (\ln ((1/\surd(2 * \pi * \sigma_1^2)) * \exp(-\ z_{i1} * (x_i - \mu_1)^2/(2 * \sigma_1^2))$$
$$(1/\surd(2 * \pi * \sigma_2^2)) * \exp(-\ z_{i2} * (x_i - \mu_2)^2/(2 * \sigma_2^2)))$$

$$=$$

$$E\ (\ln 1/\surd(2 * \pi * \sigma_1^2) - z_{i1} * (x_i - \mu_1)^2/(2 * \sigma_1^2)) +$$
$$\ln 1/\surd(2 * \pi * \sigma_2^2) - z_{i2} * (x_i - \mu_2)^2/(2 * \sigma_2^2))$$

$$=$$

$$(\ln 1/\surd(2 * \pi * \sigma_1^2) + \ln 1/\surd(2 * \pi * \sigma_2^2) - E(z_{i1} * (x_i - \mu_1)$$
$$\ln 1/\surd(2 * \pi * \sigma_2^2) - E(z_{i2} * (x_i - \mu_2)^2/(2 * \sigma_2^2))$$

$$=$$

$$(\ln 1/\surd(2 * \pi * \sigma_1^2) + \ln 1/\surd(2 * \pi * \sigma_2^2) - E(z_{i1}) * (x_i - \mu_1$$
$$\ln 1/\surd(2 * \pi * \sigma_2^2) - E(z_{i2}) * (x_i - \mu_2)^2/(2 * \sigma_2^2)$$

EM algorithm

Computing `E(zᵢⱼ)` given `h` is easy:

```
    E(zᵢ₁) = p(xᵢ | μ₁, σ₁) / (p(xᵢ | μ₁, σ₁) + p(xᵢ | μ₂, σ₂)
```

EM algorithm

We now assume that $z_{ij} = E(z_{i1})$. Therefore, we now have "complete" data and can compute h_{ml}.

EM algorithm

0. Pick arbitrary $h \in H$
1. Iterate until convergence: 1.1 Compute $E(z)$ 1.2 Use $z := E(z)$ in order to calculate h_{ml} 1.3. replace h with new h_{ml}

algorithm is to maximise $E(\ln p(d \mid h))$ *iteratively*. We consider an initial h, and then compute the corresponding value of $E(\ln p(d \mid h))$. This will involve finding out $E(z_{ij})$. We then assume that the values of the hidden variables are equal to the expected values. This gives us "complete" data, for which we can maximise the log-likelihood (not just the expected log-likelihood). We thus obtain a new h, and we iterate.