Alin Tomescu, http://people.csail.mit.edu/~alinush

Lecture 15

Project details:

- Write-up for project: 4, 6, 8 pages (for 1, 2 and 3 people respectively)
- Can be related to research but not collaborated with people outside the class
- A way to express who did what on the project

Generative modelling (continued)

Last time we talked about supervised learning where we had data: $\{(x_1, y_1), i = 1, \dots, n\}, y \in \{-1,1\}$ (even though labels did not have to be binary).

We need to find some constraint-limited way to find what the underlying distribution might be:

$$p(x, y; \theta)$$

Limiting the alternatives you are exploring while learning is critical.

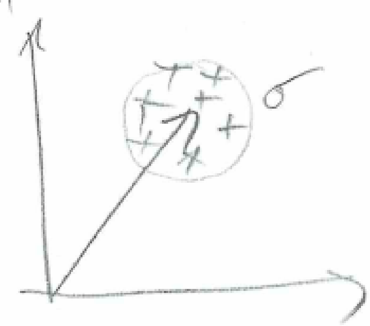
$$p(x, y; \theta) = p(x|y; \theta)P(y; \theta)$$

We have to determine the two $\mathcal{N}(x; \mu_y, \sigma^2 I)$ and P_y (for $y = \pm 1$)

Today, we assume σ is fixed.

In this model, theta is $\theta = \{\mu_1, \mu_{-1}, P_1, P_{-1}\}$. Note that we can compute $P_y = \frac{|\{Y=y\}|}{|Y|}$.

FIG 1



Each training example is sampled iid (most certainly incorrect in practice) from a normal distribution $\mathcal{N}\left(x^{(i)}; \mu_{y^{(i)}}, \sigma^2 I\right)$

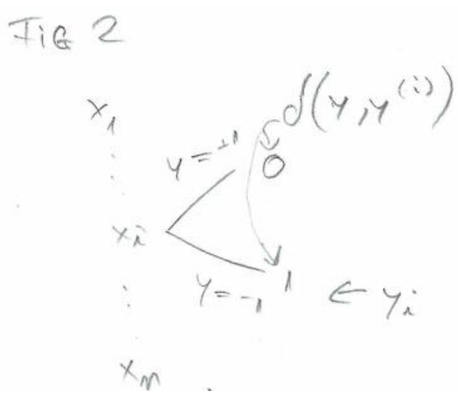
We can write down the log-likelihood for $p(x, y; \theta)$ of the data we have, where D is our training set:

$$l(\theta; D) = \sum_{i=1}^{n} \log\left[P\left(x^{(i)} | y^{(i)}; \theta\right) P\left(y^{(i)}; \theta\right)\right] = \sum_{i=1}^{n} \log\left[\mathcal{N}\left(x^{(i)}; \mu_{y^{(i)}}, \sigma^2 I\right) P_{y^{(i)}}\right]$$

$$= \sum_{i=1}^{n} \sum_{y \in Y} \delta\left(y, y^{(i)}\right) \log\left[\mathcal{N}\left(x^{(i)}; \mu_{y^{(i)}}, \sigma^2 I\right) P_{y^{(i)}}\right]$$

$$\delta\left(y, y^{(i)}\right) = \begin{cases} 1, y = y^{(i)} \\ 0, y \neq y^{(i)} \end{cases}$$

The ML estimates (if you compute them) are:



$$\frac{\partial}{\partial P_y}\left(\sum_{i=1}^{n} \sum_{y \in Y} \delta\left(y, y^{(i)}\right) \log\left[\mathcal{N}\left(x^{(i)}; \mu_y, \sigma^2 I\right) P_y\right] + \lambda\left(\sum_{y \in Y} P_y - 1\right)\right)$$

$$= \frac{\partial}{\partial P_y}\left(\sum_{i=1}^{n} \sum_{y \in Y} \delta\left(y, y^{(i)}\right) \log \mathcal{N}\left(x^{(i)}; \mu_y, \sigma^2 I\right) + \sum_{i=1}^{n} \sum_{y \in Y} \delta\left(y, y^{(i)}\right) \log P_y\right) + \lambda = \sum_{i=1}^{n} \frac{\delta\left(y, y^{(i)}\right)}{P_y} + \lambda = 0$$

$$\Rightarrow$$

$$P_y = \frac{\sum_{i=1}^{n} \delta\left(y, y^{(i)}\right)}{-\lambda}$$

If we take the sum over all the y values, we get:

$$\sum_{y \in Y} P_y = \sum_{y \in Y}\left(\frac{\sum_{i=1}^{n} \delta\left(y, y^{(i)}\right)}{-\lambda}\right) \Rightarrow 1 = \frac{\sum_{y \in Y} \sum_{i=1}^{n} \delta\left(y, y^{(i)}\right)}{-\lambda} \Rightarrow \lambda = -\sum_{i=1}^{n} \sum_{y \in Y} \delta\left(y, y^{(i)}\right) = -\sum_{i=1}^{n} 1 = -n \Rightarrow$$

$$\hat{p}_y = \frac{\sum_{i=1}^{n} \delta\left(y, y^{(i)}\right)}{n}$$

$$\hat{\mu}_y = \frac{1}{\sum_{i=1}^{n} \delta\left(y, y^{i}\right)} \sum_{i=1}^{n} \delta\left(y, y^{(i)}\right) x^{(i)}, y = \pm 1$$

So now we have classifier, it tells us exactly how the input examples are related to the labels.

Given a new example x, my predicted label is:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(x, y; \hat{\theta}) = \underset{y}{\operatorname{argmax}} P(x|y; \hat{\theta}) P(y; \hat{\theta}) = \underset{y}{\operatorname{argmax}} P(x|y; \hat{\theta}) P_y$$

Another way is to write a **discriminant function** that is positive when predicted label is positive and negative when predicted label is negative:

$$f(x;\theta) = \log\left[\frac{P(x, y=1;\theta)}{P(x, y=-1;\theta)}\right] = \log\left[\frac{\mathcal{N}(x^{(i)}; \mu_1, \sigma^2 I)\hat{P}_{y=1}}{\mathcal{N}(x^{(i)}; \mu_{-1}, \sigma^2 I)\hat{P}_{y=-1}}\right] = -\frac{1}{2\sigma^2}\|x - \hat{\mu}_1\|^2 + -\frac{1}{2\sigma^2}\|x - \hat{\mu}_{-1}\|^2 + \log\frac{\hat{P}_{y=1}}{\hat{P}_{y=-1}}$$

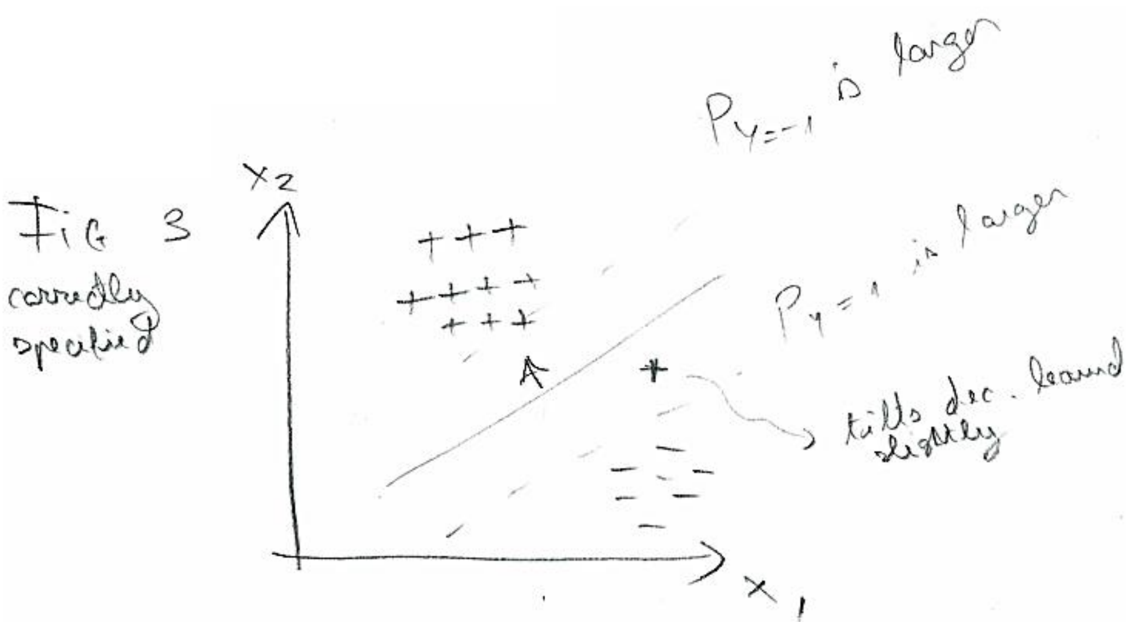
If I make the class 1 more likely apriori (make $\hat{P}_{y=1}$ higher) then I make $f(x;\theta)$ more positive.

$f(x;\theta)$ is a **quadratic discriminant function in general**. In this special case, where the variances are equal, this discriminant function is **actually linear** because if you expand it (applying $\|x - \hat{\mu}_1\|^2 = (x - \hat{\mu}_1)^T(x - \hat{\mu}_1)$) we get:

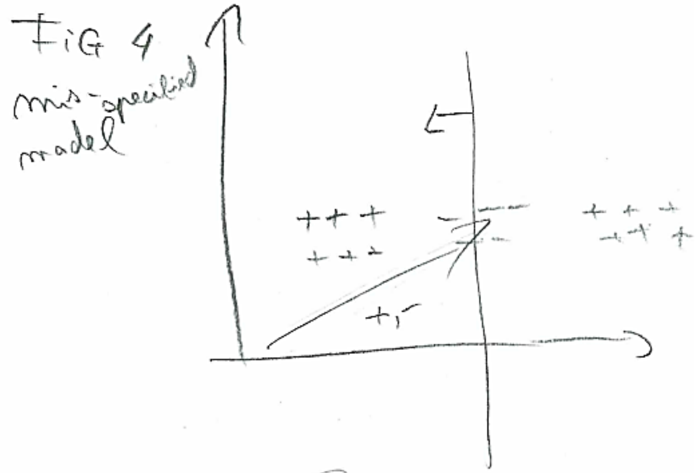
$$f(x;\theta) = -\frac{1}{2\sigma^2}(\hat{\mu}_1 + \hat{\mu}_{-1}) \cdot x - \frac{1}{2\sigma^2}\|\hat{\mu}_1\|^2 - \frac{1}{2\sigma^2}\|\hat{\mu}_{-1}\|^2 + \log\frac{\hat{P}_{y=1}}{\hat{P}_{y=-1}} = w \cdot x + w_0$$

Note: Again this only holds when $\Sigma_1 = \Sigma_{-1}$.

Example 1: When our model is correctly specified (that means we were right in picking a Gaussian model for the two clusters of + and – points)



Example 2: When our model is mis-specified (as in, we picked a Gaussian but the x values don't look like a Gaussian)

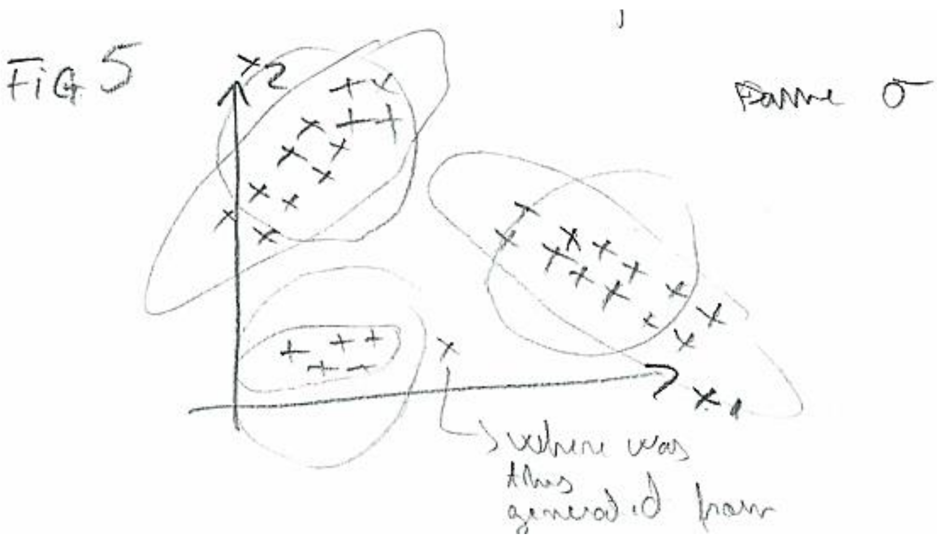


Mixture models

Let's expand this model a little bit. Let's try to estimate more complicated models.

Definition: Mixture models mix distributions together (they assume data is a mixture of multiple distributions).

Mixture models can be used in both supervised (labels are given) and unsupervised (labels are not given) learning.



Example: Our exam scores will be clustered in different probability distributions based on our backgrounds (math, programming, literature)

We still try to reconstruct $P(x|y)P(y)$, $y = 1, \ldots, k$, where k is also a parameter we have to estimate from the data. But we actually **fix** k to make the problem easier.

We are no longer doing binary classification. $D = \{x_1, \ldots, x_n\}$, we are trying to uncover the types of data points.

We need to parameterize our distributions:

$P(x|y;\theta) = N(x; \mu_y, \sigma^2 I)$, where σ is fixed for all clusters. The reason we fix σ is because it makes the problem easier.

$$P(y;\theta) = P_y$$

Assumptions about training set generation

The process that our model assumes the data was generated from is described below:

For each $i = 1, \dots, n$ we would sample $y^{(i)} \sim \text{Multinomial}(P_1, \dots, P_k)$ and once I have it I would generate a data point from a corresponding Gaussian distribution:

$$x^{(i)} = N\left(x^{(i)}; \mu_{y^{(i)}}, \sigma^2 I\right)$$

Note: there are $|y^{(i)}| = k$ such $N(\mu_y, \sigma^2 I)$ distributions that the $x^{(i)}$'s can be drawn from. In the particular case above. $k = 3$. So, you decided which one you pick from, based on what the label $y^{(i)}$ of $x^{(i)}$ was chosen as. If $y^{(i)}$ was let's say 2 (for our case with $k = 3$), then we pick $x^{(i)}$ from the 2nd distribution.

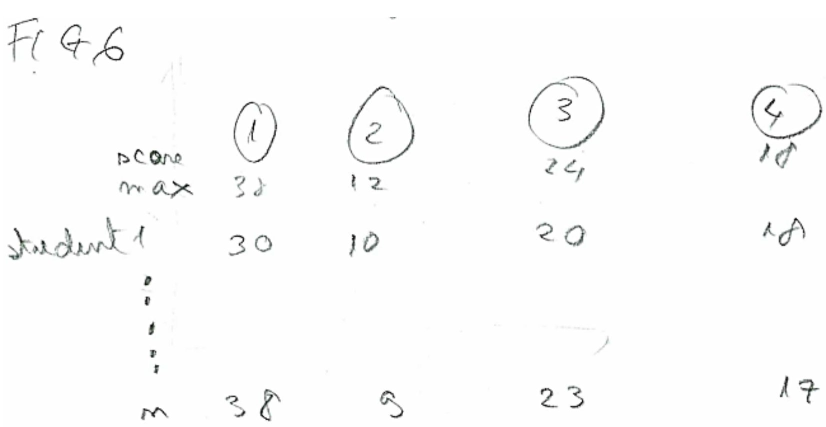
Now, given this way of generating the data, except we don't get the labels, but we get the k, how can we figure out the clusters?

Given data $D = \{x_1, \dots, x_n\}$ what is the log-likelihood of generating that data?

$$l(\theta; D) = \sum_i^n \log\left[\sum_y^k N\left(x^{(i)}; \mu_y, \sigma^2 I\right) P_y\right]$$

This is **difficult to maximize?** This is called *incomplete log-likelihood*.

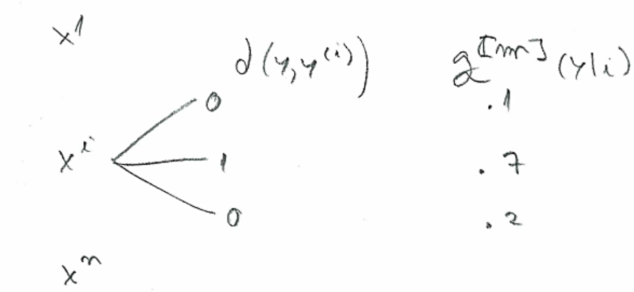
Example: Suppose student took a 4 question exam with max. grades 38, 12, 24 and 18 for questions 1, 2, 3 and 4 respectively. We might want to cluster together students based on how well they did on certain questions. Maybe it turns out there are 4 types of students, where each type does extremely well on question i and very poorly on the others.



The EM (expectation-maximization) algorithm

Estimation step (E-step): Figures out what the labels are (see figure 7)

Fig 7



Maximization step (M-step): use the label assignments to do ML estimation

$$l(\theta; x, y) = \sum_{i=1}^{n} \sum_{y=1}^{k} \delta(y, y^{(i)}) \log\left[N(x^{(i)}; \mu_y, \sigma^2 I) P_y\right]$$

$$\hat{p}_y = \frac{\sum_i \delta(y, y^{(i)})}{n}$$

$$\hat{\mu}_y = \cdots \text{(as before)}$$

This would be nice, but we don't have the labels. How do we figure out what the labels are? I could pick them randomly. You can do a clustering algorithm. You could specify some parameters, like mean and σ and then you have a model, and you can use it to predict the labels by predicting that model is the truth, then reestimate the model and then refine the assignments.

$$E_{y^{(i)}|x^{(i)};\theta^{[m]}}\{\delta(y, y^{(i)})\} = E\{\delta(y, y^{(i)}) \mid x^{(i)}, \theta^{[m]}\}$$

Step 1: $\theta^{[0]}$ is chosen at random just to get started.

Step E: Estimation step becomes: $q^{[m]}(y|i) = E\{\delta(y, y^{(i)}) \mid x^{(i)}, \theta^{[m]}\} = P(y|x^{(i)}; \theta^{[m]})$

(intuition: compute new assignments based on μ_y)

Step M: Maximization step becomes:

We want to increase:

$$E\left\{l(\theta; x, y) = \sum_{i=1}^{n} \sum_{y=1}^{k} q^{[m]}(y|i) \log\left[N(x^{(i)}; \mu_y, \sigma^2 I) P_y\right]\right\}$$

$$\hat{p}_y^{[m+1]} = \frac{\sum_{i=1}^{n} q^{[m]}(y|i)}{n}, i = 1 \ldots k$$

$$\hat{\mu}_y^{[m+1]} = \frac{1}{\sum_{i=1}^{n} q^{[m]}(y \mid i)} \sum_{i=1}^{n} q^{[m]}(y \mid i) \cdot x^{(i)}, i = 1 \ldots k$$

(intuition: recompute μ_y based on labels, $\forall y$)

You can show that each iteration of this algorithm increases that log-likelihood. At some point, the mean and $\hat{p}_y$ will not change anymore at which point we would have converged.

Notes from office hours:

Note that the EM algorithm really maximizes for:

$$\underset{\theta}{\operatorname{argmax}} \sum_{x \in S_n} \log p\left(x^{(i)}; \theta\right)$$

Where the training set S_n is given **without** the labels $y^{(i)}$.

Since, in general $P(A) = \sum_{B_i \in \mathcal{B}} P(A \cap B_i)$, this becomes:

$$\underset{\theta}{\operatorname{argmax}} \sum_{x \in S_n} \log p(x; \theta) = \underset{\theta}{\operatorname{argmax}} \sum_{x \in S_n} \log \sum_{y=1}^{k} p(x, y; \theta)$$

Now, since it's not mathematically convenient to compute the $\log$ of a sum, and since a Gaussian is a concave function it can be shown that:

$$\underset{\theta}{\operatorname{argmax}} \sum_{x \in S_n} \log \sum_{y=1}^{k} p(x, y; \theta) \geq \underset{\theta}{\operatorname{argmax}} \sum_{x \in S_n} \sum_{y=1}^{k} \log p(x, y; \theta)$$

This means that maximizing the right-side will also maximize the left-side (since the left side is a lower bound for the right side).

Again, k is assumed to be known, so as to make the problem easier.