Generative ML II

CS771: Introduction to Machine Learning

Purushottam Kar

Announcements

Doubt clearing session: Sep 14 (Sat), **6:30PM** KD101 (time changed!)

Mid-sem Exam: September 15 (Sunday), 6-8PM, L19, L20

- *Syllabus is till whatever we cover today i.e. Sep 13 (Fri)*
- *Bring your **institute ID card** with you – will lose time if you forget*
- *Bring a **pencil, pen, eraser, sharpener** with you – we wont provide!*
- *Answers to be written on question paper itself. If you write with pen and make a mistake, no extra paper. Final answer **must be in pen***
- ***Auditors cannot appear** for the mid-sem exam*

Seating for mid-sem exam: assigned seating (announced on Piazza)

- *Everyone has been assigned a lecture hall, and a seat number in that hall*
- *All must go to their assigned hall – no extra space if you go to wrong hall*

Recap of Last Lecture

Simple generative models to capture variations in data using a single Gaussian – either standard $\mathcal{N}(\boldsymbol{\mu}, I_d)$, or spherical $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \cdot I_d)$, or general $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$

A more powerful generative model that can fit multiple Gaussians to capture even more intricate and diverse variations in the data

Notion of *latent variables* – used to reason about quantities about which we admit that training data does not directly tell us anything

MLE with latent variables – NP-hard in general. Two heuristics

Alternating Optimization

Expectation Maximization

In the lecture Jupyter notebook, we saw that these powerful methods do indeed capture details about data better and give superior samples

First A Detour – Mixed Regression

An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*

First A Detour – Mixed Regression

An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*

First A Detour – Mixed Regression

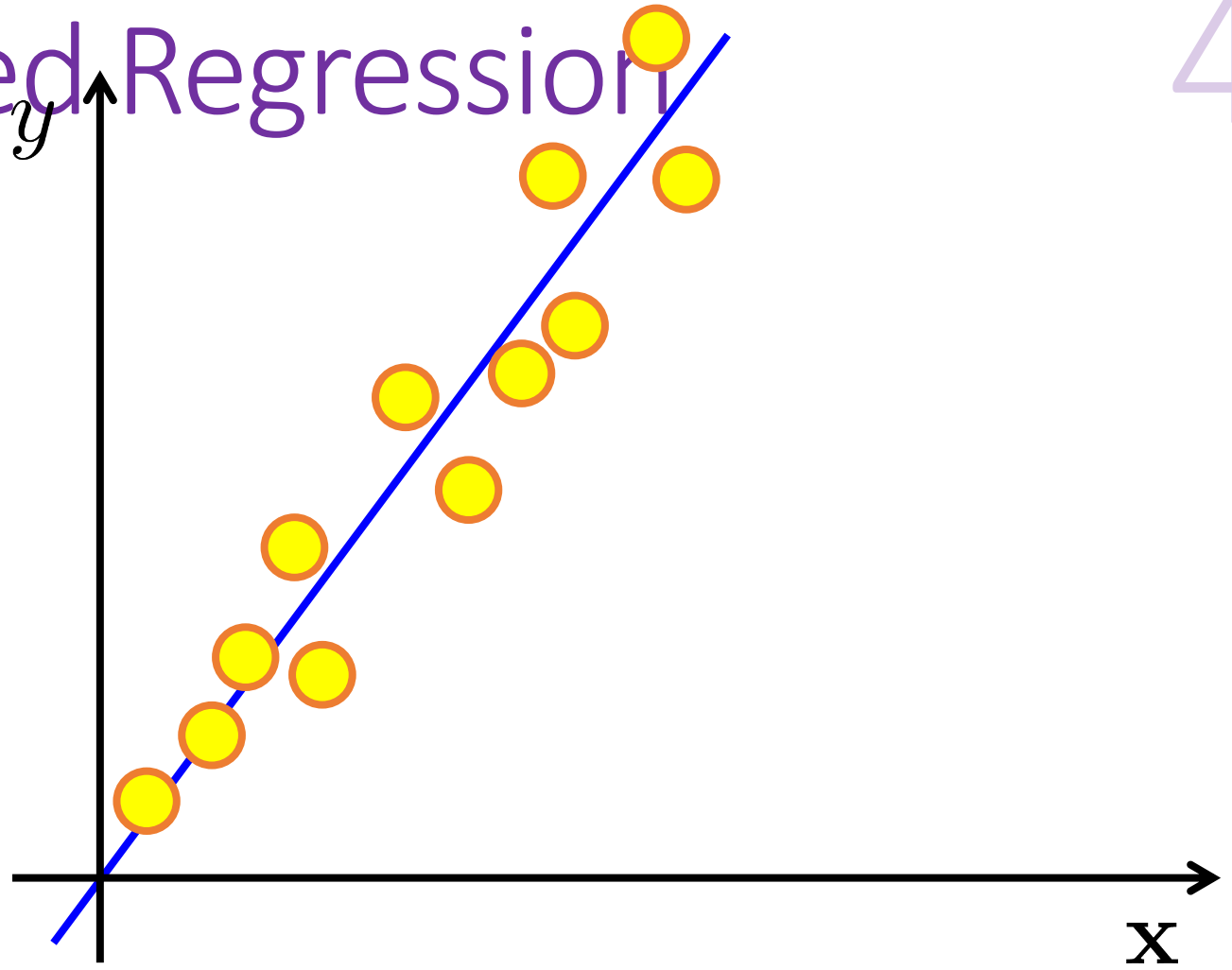
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



First A Detour – Mixed Regression

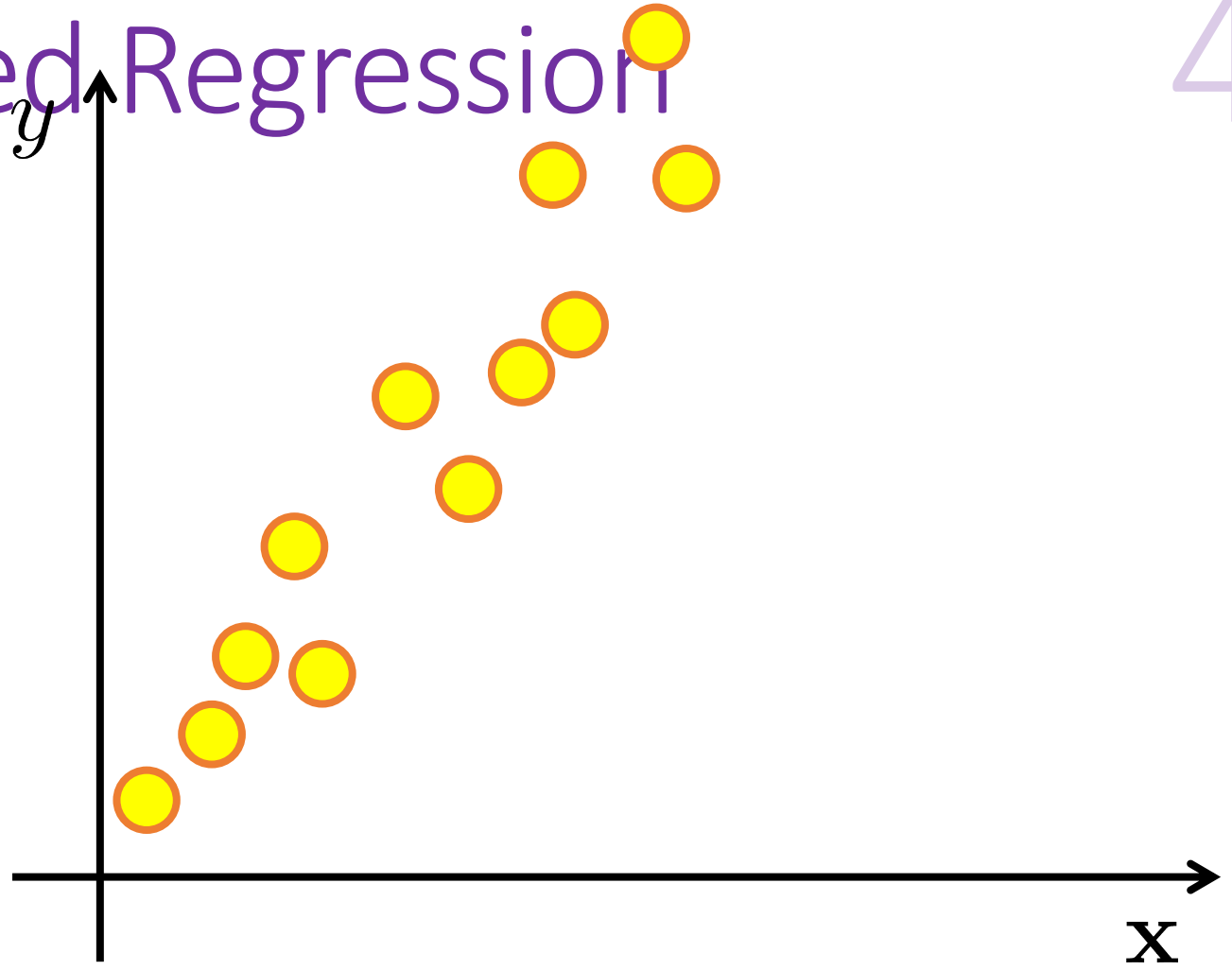
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



First A Detour – Mixed Regression

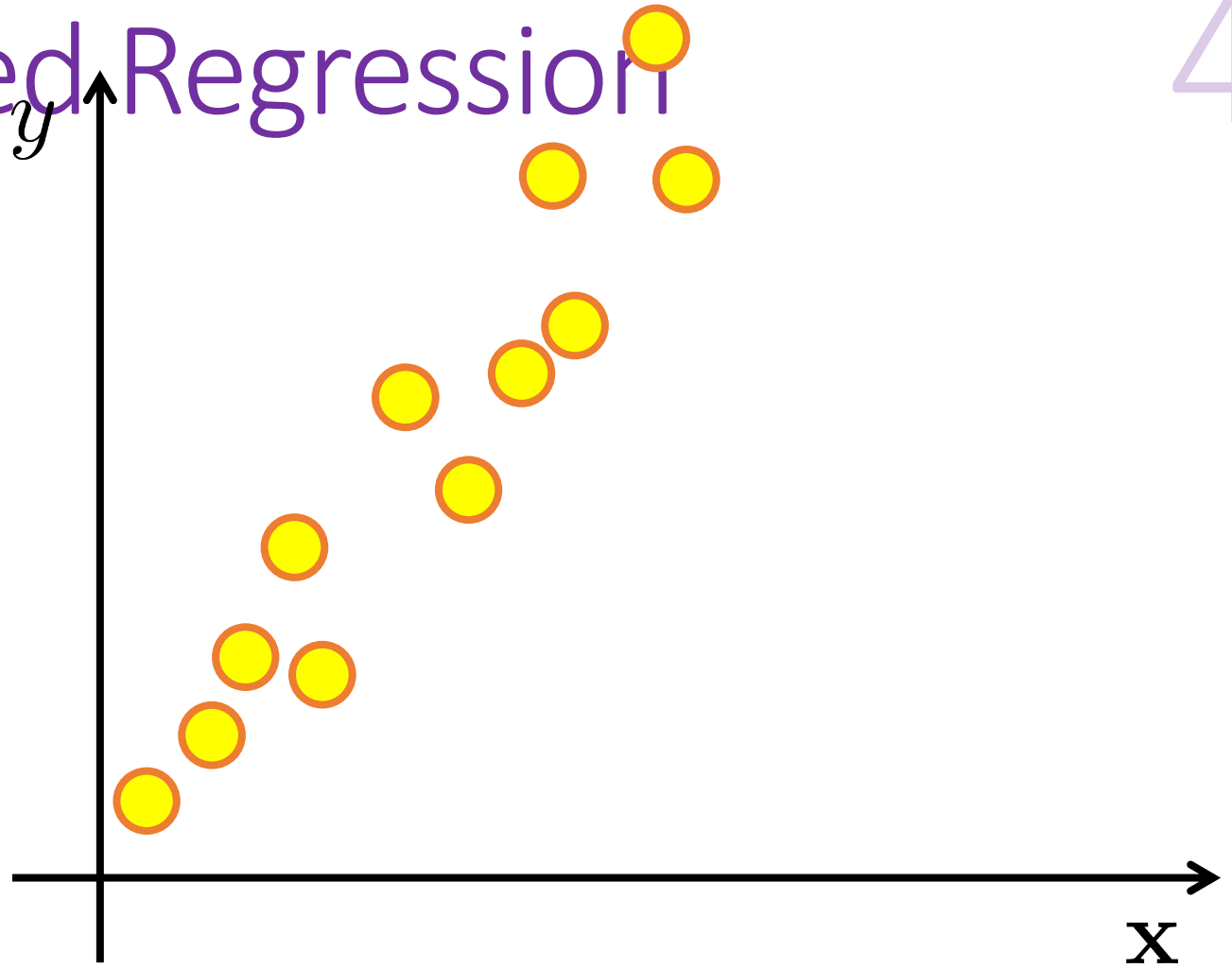
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

First A Detour – Mixed Regression

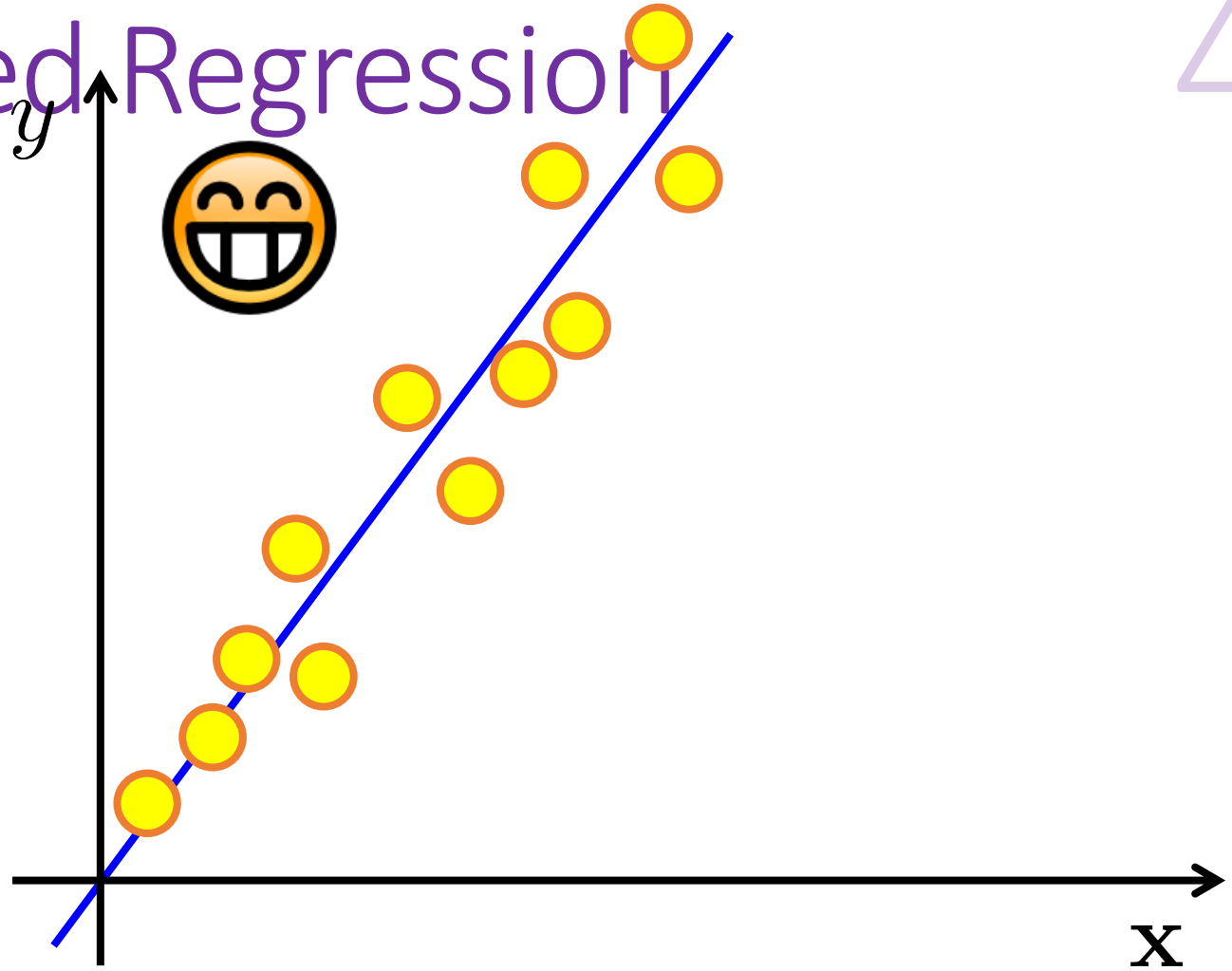
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

First A Detour – Mixed Regression

An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

First A Detour – Mixed Regression

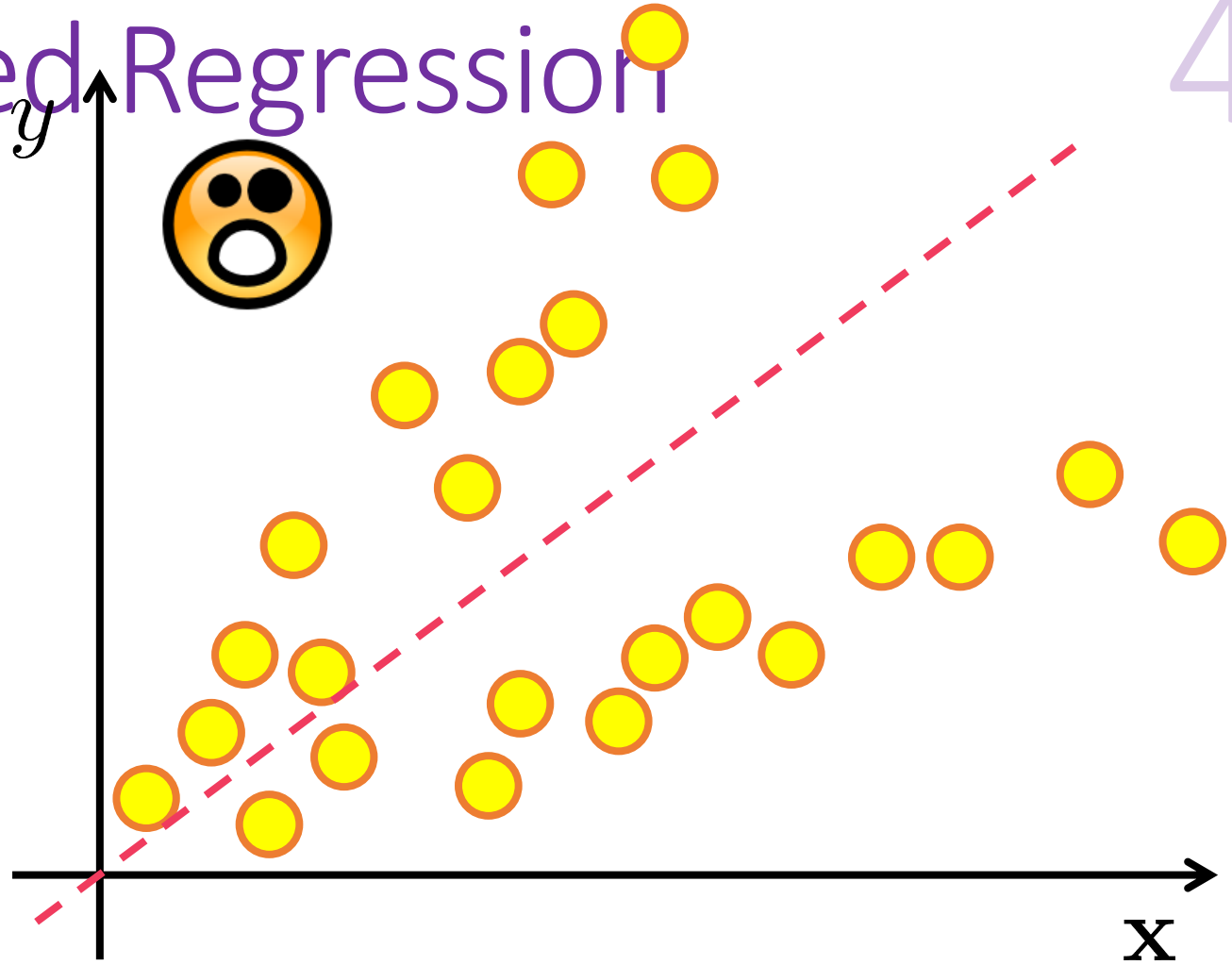
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

First A Detour – Mixed Regression

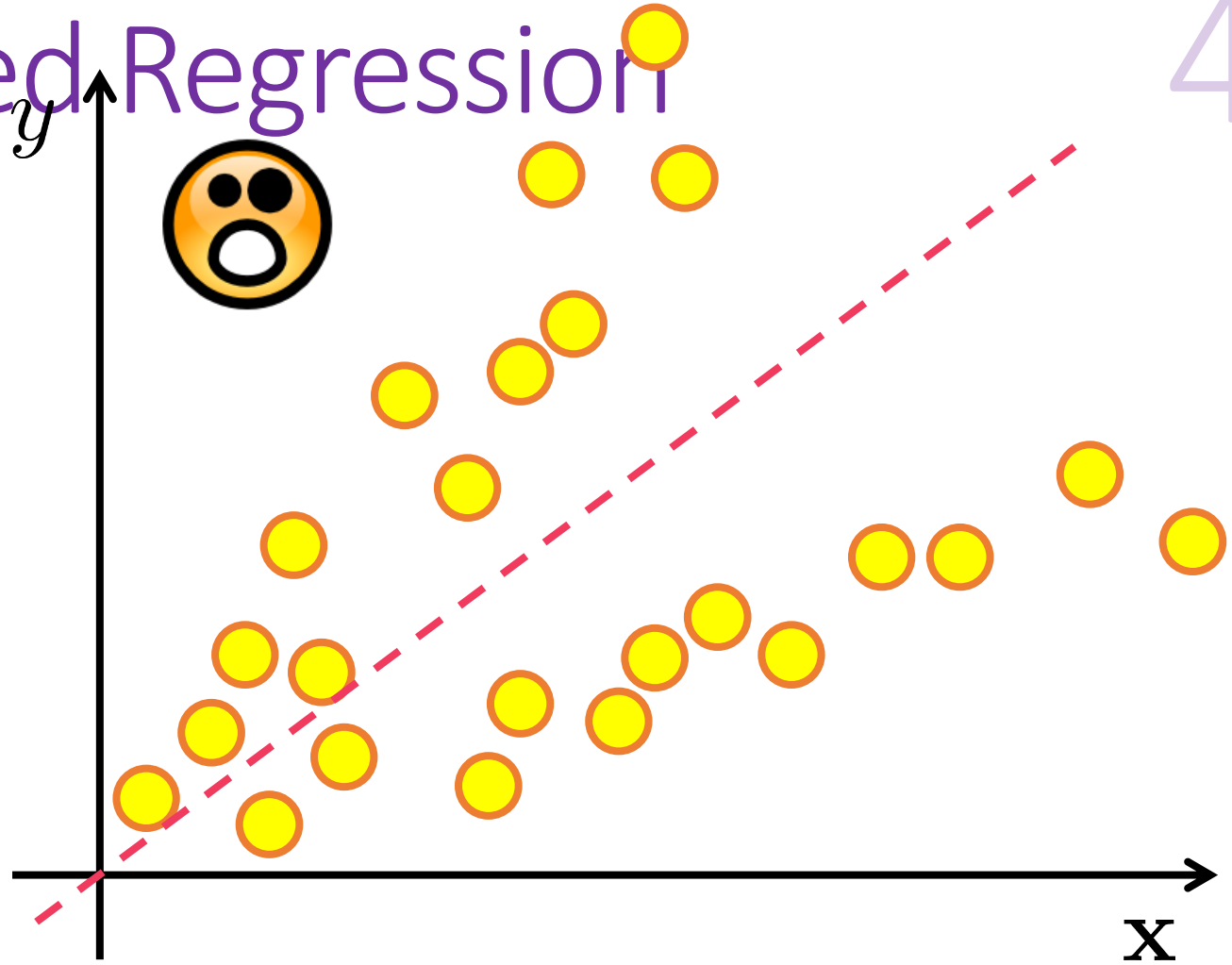
An example of latent variables in a supervised learning task

We have regression train data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ $\mathbf{x}^i \in \mathbb{R}^d, y^i \in \mathbb{R}$

Example: $\mathbf{x}$ denotes age and y denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

First A Detour – Mixed Regression

An example of latent variables

ir

W

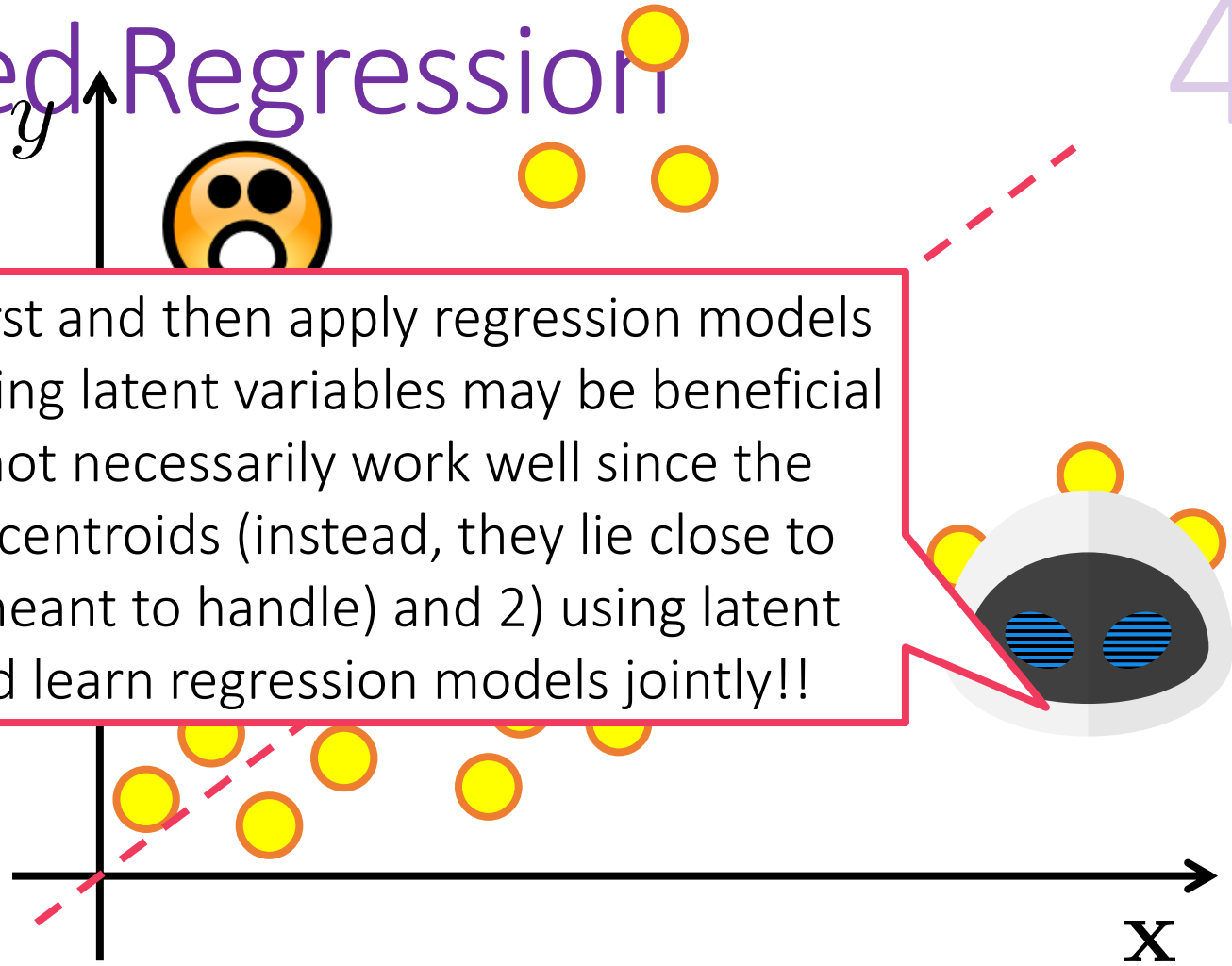
{

E

denotes time spent on website

There are two subpopulations in data (gender) which behave differently even if age is same

An indication that our features may be incomplete/*latent*



Sure, we could try clustering this data first and then apply regression models separately on both clusters. However, using latent variables may be beneficial since 1) clustering e.g. k-means may not necessarily work well since the points here are really not close to two centroids (instead, they lie close to two lines which k-means is really not meant to handle) and 2) using latent variables, we can elegantly cluster and learn regression models jointly!!

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle \right)^2 + \lambda \cdot \|\mathbf{w}\|_2^2$$

$$= (\mathbf{X}^\top \mathbf{X} + \lambda \cdot I)^{-1} \mathbf{X}^\top \mathbf{y}$$

Latent Variables to the Rescue

As before, if we believe that our data is best explained using two linear regression models instead of one, we should work with a *mixed model* (aka mixture of experts)

Will fit two regression models to the data and use a latent variable $z_i \in \{1,2\}$ to keep track of which data point belongs to which model

Let us use Gaussian likelihoods since we are comfortable with it

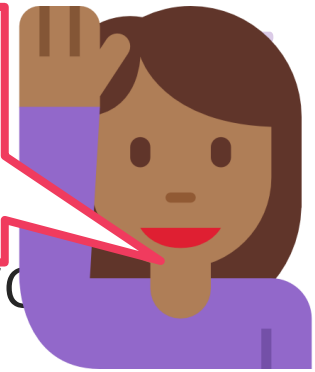
$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = 1, \mathbf{w}^1, \mathbf{w}^2\right] = \mathcal{N}\left(y^i; \langle \mathbf{w}^1, \mathbf{x}^i \rangle, \sigma^2\right)$$
$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = 2, \mathbf{w}^1, \mathbf{w}^2\right] = \mathcal{N}\left(y^i; \langle \mathbf{w}^2, \mathbf{x}^i \rangle, \sigma^2\right)$$

Note: this is not generative learning since we are still learning *discriminative* distributions of the form $\mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w}^c\right]$

Will see soon how to perform generative learning in supervised settings

Latent

We could have had separate σ_1^2 and σ_2^2 for the two components as well which we could also learn. However, this would make things more tedious so for now, let us assume $\sigma_1^2 = \sigma_2^2 = 1$ and also that $\sigma^2 = 1$



As before, if we believe that our data is best explained using two linear regression models instead of one, we should work with a *mixed model* (aka mixture of experts)

Will fit two regression models to the data and use a latent variable $z_i \in \{1,2\}$ to keep track of which data point belongs to which model

Let us use Gaussian likelihoods since we are comfortable with it

$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = 1, \mathbf{w}^1, \mathbf{w}^2\right] = \mathcal{N}\left(y^i; \langle \mathbf{w}^1, \mathbf{x}^i \rangle, \sigma^2\right)$$
$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = 2, \mathbf{w}^1, \mathbf{w}^2\right] = \mathcal{N}\left(y^i; \langle \mathbf{w}^2, \mathbf{x}^i \rangle, \sigma^2\right)$$

Note: this is not generative learning since we are still learning *discriminative* distributions of the form $\mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w}^c\right]$

Will see soon how to perform generative learning in supervised settings

MLE for Mixed Regression

$\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d} \sum_{i=1}^n \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w}^1, \mathbf{w}^2\right]$ which, upon introducing latent variables gives $\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d} \sum_{i=1}^n \ln\left(\sum_{c\in\{1,2\}} \mathbb{P}\left[y^i, z_i = c \mid \mathbf{x}^i, \mathbf{w}^1, \mathbf{w}^2\right]\right)$

Method 1: Alternating Optimization

$$\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d, z_i\in\{1,2\}} \sum_{i=1}^n \ln \mathbb{P}\left[y^i, z_i \mid \mathbf{x}^i, \mathbf{w}^1, \mathbf{w}^2\right]$$

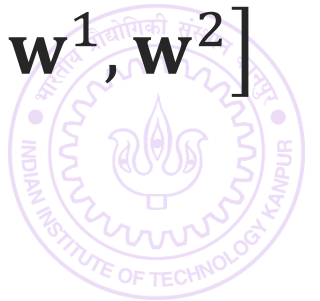
As before, assume $\mathbb{P}\left[z_i \mid \mathbf{x}^i, \mathbf{w}^1, \mathbf{w}^2\right] =$ *constant for sake of simplicity to get*

$$\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d, z_i\in\{1,2\}} \sum_{i=1}^n \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i, \mathbf{w}^1, \mathbf{w}^2\right]$$

Step 1*: Fix* $\mathbf{w}^1, \mathbf{w}^2$ *and update all* $z_i = \arg\max_{c\in\{1,2\}} \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = c, \mathbf{w}^1, \mathbf{w}^2\right]$

Step 2*: Fix all* z_i *and update the models*

$\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d} \sum_{i=1}^n \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i, \mathbf{w}^1, \mathbf{w}^2\right]$



Alternating Optimization for MR

As before, we assumed the likelihood distributions as

$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = c, \mathbf{w}^1, \mathbf{w}^2\right] = \mathcal{N}\left(y^i; \langle \mathbf{w}^c, \mathbf{x}^i \rangle, 1\right) \text{ for } c \in \{1,2\}$$

Step 1 becomes

$$\arg\max_{c \in \{1,2\}} \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = c, \mathbf{w}^1, \mathbf{w}^2\right] = \arg\min_{c \in \{1,2\}} \left| y^i - \langle \mathbf{w}^c, \mathbf{x}^i \rangle \right|$$

i.e. assign every data point to its "closest" line or the line which fits it better

Step 2 becomes

$$\arg\max_{\mathbf{w}^1, \mathbf{w}^2 \in \mathbb{R}^d} \sum_{i=1}^n \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i, \mathbf{w}^1, \mathbf{w}^2\right]$$

$$= \arg\min_{\mathbf{w}^1 \in \mathbb{R}^d} \sum_{i:z_i=1} \left(y^i - \langle \mathbf{w}^1, \mathbf{x}^i \rangle\right)^2 + \arg\min_{\mathbf{w}^2 \in \mathbb{R}^d} \sum_{i:z_i=2} \left(y^i - \langle \mathbf{w}^2, \mathbf{x}^i \rangle\right)^2$$

i.e. perform least squares on the data points assigned to each component

May incorporate a prior as well to add a regularizer (ridge regression)

Repeat!

Alternating Optimization for MR

As before, $\mathbb{P}[y^i \mid \mathbf{x}$

Step 1 be

$\arg \max_{c \in \{1,}$

i.e. assi

Step 2 be

$\arg \max_{\mathbf{w}^1, \mathbf{w}}$

$= \arg \min_{\mathbf{w}^1 \in \mathbb{R}^d}$

$\langle \mathbf{w}^c, \mathbf{x}^i \rangle|$

ter

$\langle \mathbf{w}^2, \mathbf{x}^i \rangle)^2$

i.e. perform least squares on the data points assigned to each component

May incorporate a prior as well to add a regularizer (ridge regression)

Repeat!

AltOpt for MR

1. Initialize models $\{\mathbf{w}^c\}_{c \in \{1,2\}}$
2. For $i \in [n]$, update $\{z_i\}$ using $\{\mathbf{w}^c\}$
   1. Let $z_i = \arg\min_c \left| y^i - \langle \mathbf{w}^c, \mathbf{x}^i \rangle \right|$
3. Update $\{\mathbf{w}^c\}$ using $\{z_i\}$
   1. Let $\mathbf{w}^c = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i:z_i=c} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle \right)^2$
4. Repeat until convergence

EM for Mixed Regression

Original Prob: $\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d} \sum_{i=1}^{n} \ln\left(\sum_{c\in\{1,2\}} \mathbb{P}\left[y^i, z_i = c \mid \mathbf{x}^i, \mathbf{w}^1, \mathbf{w}^2\right]\right)$

Step 1 (**E Step**) Consists of two sub-steps

Step 1.1 *Assume our current model estimates are* $\mathbf{w}^1 = \mathbf{p}, \mathbf{w}^2 = \mathbf{q}$

Use the current models to ascertain how likely are different values of z_i for the i-th data point i.e. compute $q_c^i = \mathbb{P}\left[z_i = c \mid y^i, \mathbf{x}^i, \mathbf{p}, \mathbf{q}\right]$ for both $c \in \{1,2\}$

Step 1.2 *Use weights* q_c^i *to set up a new objective function*

As before, assume $\mathbb{P}\left[z_i \mid \mathbf{x}^i, \mathbf{p}, \mathbf{q}\right]$ = constant for sake of simplicity

$\sum_{i=1}^{n} \sum_{c\in\{1,2\}} q_c^i \cdot \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = c, \mathbf{w}^1, \mathbf{w}^2\right]$

Step 2 (M Step) Maximize the new obj. fn. to get new models

$$\arg\max_{\mathbf{w}^1,\mathbf{w}^2\in\mathbb{R}^d} \sum_{i=1}^{n} \sum_{c\in\{1,2\}} q_c^i \cdot \ln \mathbb{P}\left[y^i \mid \mathbf{x}^i, z_i = c, \mathbf{w}^1, \mathbf{w}^2\right]$$

Repeat!

EM for Mixed Regression

EM for MR

1. Initialize means $\{\mathbf{w}^c\}_{c=1\ldots C}$ (for C components)
2. For $i \in [n]$, update q_c^i using $\{\mathbf{w}^c\}$
   1. Let $p_c^i = \exp\left(-\frac{\left(y^i - \langle \mathbf{w}^c, \mathbf{x}^i \rangle\right)_2^2}{2}\right)$
   2. Let $q_c^i = \frac{p_c^i}{\sum_{c=1}^C p_c^i}$ (normalize)
3. Update $\mathbf{w}^c = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n q_c^i \cdot \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)_2^2$
   $= \arg\min_{\mathbf{w} \in \mathbb{R}^d} (X\mathbf{w} - \mathbf{y})^\top Q_c (X\mathbf{w} - \mathbf{y})$ where $Q_c = \mathrm{diag}(q_c^1, \ldots, q_c^n)$
   $= (X^\top Q_c X)^{-1} (X^\top Q_c \mathbf{y})$ (apply first order optimality)
4. Repeat until convergence

Generative Supervised Learning

Core idea behind generative learning for classification with C classes

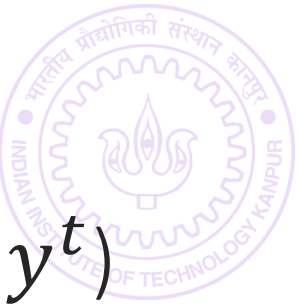
> *For each class $c \in [C]$ learn what do data points of that class look like (using a distribution). For a test data point, ask each of these C distributions to vote based on how much they think the data point belongs to their class*

Interpreting the above in language of probability, for a test point $\mathbf{x}^t$, predict the label $\hat{y}^t$ based on the following rule ($\boldsymbol{\theta}$ is the model)

$$\hat{y}^t = \arg\max_{c \in [C]} \mathbb{P}[y^t = c \mid \mathbf{x}^t, \boldsymbol{\theta}]$$

$$= \arg\max_{c \in [C]} \frac{\mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}] \cdot \mathbb{P}[y^t = c \mid \boldsymbol{\theta}]}{\mathbb{P}[\mathbf{x}^t \mid \boldsymbol{\theta}]}$$ (apply Bayes rule)

$$= \arg\max_{c \in [C]} \mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}] \cdot \mathbb{P}[y^t = c \mid \boldsymbol{\theta}]$$ (ignore terms w/o y^t)

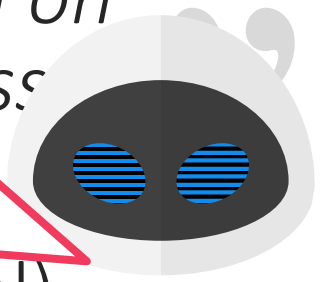


Generative Supervised Learning

Core idea behind generative learning for classification with C classes

> *For each class $c \in [C]$ learn what do data points of that class look like (using a distribution). For a test data point, ask each of t… d on how much they think t… ss*

Every generative learning algo takes a decision on how to learn these probability distributions from data. Let us see one example of how this can be done



Interpreting the above in lang…
predict the label $\hat{y}^t$ based on …el)

$$\hat{y}^t = \arg\max_{c \in [C]} \mathbb{P}[y^t = c \mid \mathbf{x}^t, \boldsymbol{\theta}]$$

$$= \arg\max_{c \in [C]} \frac{\mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}] \cdot \mathbb{P}[y^t = c \mid \boldsymbol{\theta}]}{\mathbb{P}[\mathbf{x}^t \mid \boldsymbol{\theta}]}$$ (apply Bayes rule)

$$= \arg\max_{c \in [C]} \mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}] \; \mathbb{P}[y^t = c \mid \boldsymbol{\theta}]$$ (ignore terms w/o y^t)

A simple generative model

For sake of simplicity, assume $C = 2$ (binary classification)

Choose $\mathbb{P}[y^t = 1 \mid \boldsymbol{\theta}] = p$ and consequently, $\mathbb{P}[y^t = -1 \mid \boldsymbol{\theta}] = 1 - p$

The quantity $\mathbb{P}[y^t = 1 \mid \boldsymbol{\theta}]$ *tells us the* prior *class probability (aka class marginal prob) i.e. how frequently do we see elements of the class 1 overall*

In contrast, $\mathbb{P}[y^t = 1 \mid \mathbf{x}^t, \boldsymbol{\theta}]$ *tells us the* posterior *class probability i.e. how likely is the class 1 the correct class for this particular data point*

Example*:* $\mathbb{P}[y^t = 1 \mid \boldsymbol{\theta}]$ *is similar to the general probability of Nadal winning if playing against Federer (approx. 60%)*

Example*:* $\mathbb{P}[y^t = 1 \mid \mathbf{x}^t, \boldsymbol{\theta}]$ *is similar to the probability of Nadal winning if playing against Federer on a* $\mathbf{x}^t$ *= grass court (approx. 25%)*

Thus, we are modelling $\mathbb{P}[y^t = c \mid \boldsymbol{\theta}]$ as a Rademacher distribution

Had it been a multiclass problem, we would have used a multinoulli (aka categorical distribution) to model this!

A simple generative model

We now take care of $\mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}]$ – this is nothing but a generative model for feature vectors of class c – aka class conditional dist.

We have already seen ways to model generative distributions (as a single or a mixture or Gaussians)

Let us choose to model each class using a single Gaussian for simplicity

Could have used a mixture too – more powerful modelling

$\mathbb{P}[\mathbf{x}^t \mid y^t = 1, \boldsymbol{\theta}] = \mathcal{N}(\boldsymbol{\mu}^+, \Sigma^+)$ and $\mathbb{P}[\mathbf{x}^t \mid y^t = -1, \boldsymbol{\theta}] = \mathcal{N}(\boldsymbol{\mu}^-, \Sigma^-)$

Thus, our model should be $\boldsymbol{\theta} = \{\{\boldsymbol{\mu}^+, \Sigma^+\}, \{\boldsymbol{\mu}^-, \Sigma^-\}, p\}$

All that is left is to estimate these parameters ☺

Will use the MLE route to do so – however, can have priors over all/some of these parameters $\boldsymbol{\mu}^+, \Sigma^+, \boldsymbol{\mu}^-, \Sigma^-, p$ and do MAP/Bayesian inference too – generative Bayesian learning!

MLE for generative classification

Since both $\mathbf{x}$ and y are getting modelled here, the likelihood function now looks at the joint probability of $\mathbb{P}[\mathbf{x}, y \mid \boldsymbol{\theta}]$

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathbb{P}[\mathbf{x}^1, y^1, \ldots, \mathbf{x}^n, y^n \mid \boldsymbol{\theta}]$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i, y^i \mid \boldsymbol{\theta}\right]$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\theta}\right] \cdot \mathbb{P}\left[y^i \mid \boldsymbol{\theta}\right]$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^+, \Sigma^+, \boldsymbol{\mu}^-, \Sigma^-\right] \cdot \mathbb{P}\left[y^i \mid p\right]$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{y^i}, \Sigma^{y^i}\right] \cdot \mathbb{P}\left[y^i \mid p\right]$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ln \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{y^i}, \Sigma^{y^i}\right] + \ln \mathbb{P}\left[y^i \mid p\right]$$

MLE for generative classification

Since both $\mathbf{x}$ and y are getting modelled here, the likelihood function now looks at the joint probability of $\mathbb{P}[\mathbf{x}, y \mid \boldsymbol{\theta}]$

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathbb{P}[\mathbf{x}^1, y^1, \ldots, \mathbf{x}^n, y^n \mid \boldsymbol{\theta}]$$

Independence

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i, y^i \mid \boldsymbol{\theta}\right]$$

Chain Rule

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\theta}\right] \cdot \mathbb{P}\left[y^i \mid \boldsymbol{\theta}\right]$$

Get rid of conditioning that does not matter

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^+, \Sigma^+, \boldsymbol{\mu}^-, \Sigma^-\right] \cdot \mathbb{P}\left[y^i \mid p\right]$$

Get rid of conditioning that does not matter

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{y^i}, \Sigma^{y^i}\right] \cdot \mathbb{P}\left[y^i \mid p\right]$$

Take logs

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ln \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{y^i}, \Sigma^{y^i}\right] + \ln \mathbb{P}\left[y^i \mid p\right]$$

MLE for generative classification

Thus, we have

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ln \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{y^i}, \Sigma^{y^i}\right] + \ln \mathbb{P}\left[y^i \mid p\right]$$

This neatly breaks up into three optimization problems

$$\hat{p}_{\text{MLE}} = \arg\max_{p} \sum_{i=1}^{n} \ln \mathbb{P}\left[y^i \mid p\right]$$

$$\left\{\hat{\boldsymbol{\mu}}^{+}_{\text{MLE}}, \hat{\Sigma}^{+}_{\text{MLE}}\right\} = \arg\max_{\boldsymbol{\mu}^{+}, \Sigma^{+}} \sum_{i:y^i=1} \ln \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{+}, \Sigma^{+}\right]$$

$$\left\{\hat{\boldsymbol{\mu}}^{-}_{\text{MLE}}, \hat{\Sigma}^{-}_{\text{MLE}}\right\} = \arg\max_{\boldsymbol{\mu}^{-}, \Sigma^{-}} \sum_{i:y^i=-1} \ln \mathbb{P}\left[\mathbf{x}^i \mid y^i, \boldsymbol{\mu}^{-}, \Sigma^{-}\right]$$

We have seen how to solve each one of these problems!!

MLE for generative classification

$$\hat{p}_{\text{MLE}} = \arg\max_{p} \sum_{i=1}^{n} \ln \mathbb{P}\left[y^i \mid p\right]$$

Let n_+ (resp. n_-) be number of training data points with label 1 (resp. -1)

$$= \arg\max_{p} n_+ \cdot \ln p + n_- \cdot \ln(1-p)$$

Applying first order optimality gives us $\hat{p}_{\text{MLE}} = \frac{n_+}{n_+ + n_-} = \frac{n_+}{n}$

The other two problems we solved in the last class

$$\hat{\boldsymbol{\mu}}^{+}_{\text{MLE}} = \frac{1}{n_+} \sum_{i:y^i=1} \mathbf{x}^i \text{ and } \hat{\boldsymbol{\mu}}^{-}_{\text{MLE}} = \frac{1}{n_-} \sum_{i:y^i=-1} \mathbf{x}^i$$

$$\hat{\Sigma}^{+}_{\text{MLE}} = \frac{1}{n_+} \sum_{i:y^i=1} \left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{+}_{\text{MLE}}\right)\left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{+}_{\text{MLE}}\right)^{\top} \text{ and}$$

$$\hat{\Sigma}^{-}_{\text{MLE}} = \frac{1}{n_-} \sum_{i:y^i=-1} \left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{-}_{\text{MLE}}\right)\left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{-}_{\text{MLE}}\right)^{\top}$$

Special Cases – I

Recall that in generative algorithms, we make predictions using $\arg\max_{c \in [C]} \mathbb{P}[\mathbf{x}^t \mid y^t = c, \boldsymbol{\theta}] \cdot \mathbb{P}[y^t = c \mid \boldsymbol{\theta}]$

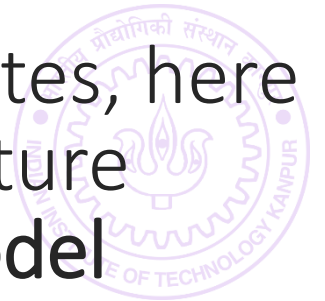
Let us look at the special case when we fix $\Sigma^+ = \Sigma^- = I_d$

In this case, we will predict $\hat{y}^t = 1$ only if

$\exp(-\|\mathbf{x}^t - \hat{\boldsymbol{\mu}}^+\|_2^2/2) \cdot \frac{n_+}{n} \geq \exp(-\|\mathbf{x}^t - \hat{\boldsymbol{\mu}}^-\|_2^2/2) \cdot \frac{n_-}{n}$ which happens exactly when $\mathbf{w}^\top \mathbf{x}^t + b \geq 0$ (i.e. linear classifier!!) where

$\mathbf{w} = 2(\hat{\boldsymbol{\mu}}^+ - \hat{\boldsymbol{\mu}}^-)$ and $b = \|\hat{\boldsymbol{\mu}}^-\|_2^2 - \|\hat{\boldsymbol{\mu}}^+\|_2^2 + 2\ln\left(\frac{n_+}{n_-}\right)$

Note that since a standard Gaussian has independent coordinates, here we implicitly assumed that the different coordinates of the feature vector $\mathbf{x}^t$ were independent – this is called the **Naïve Bayes model**



Special Cases – II

Next case: let us fix $\Sigma^+ = \Sigma^- = \Sigma$ but not necessarily I_d

In this case we have (similar calculations as before)

$\hat{\boldsymbol{\mu}}^+_{\text{MLE}} = \frac{1}{n_+}\sum_{i:y^i=1} \mathbf{x}^i$ and $\hat{\boldsymbol{\mu}}^-_{\text{MLE}} = \frac{1}{n_-}\sum_{i:y^i=-1} \mathbf{x}^i$

$$\hat{\Sigma}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{y^i}_{\text{MLE}}\right)\left(\mathbf{x}^i - \hat{\boldsymbol{\mu}}^{y^i}_{\text{MLE}}\right)^{\top}$$

However, even in this case, decision boundary remains linear with

$\mathbf{w} = 2\hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}^+ - \hat{\boldsymbol{\mu}}^-)$ and $b = \langle \hat{\boldsymbol{\mu}}^-, \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}^- \rangle - \langle \hat{\boldsymbol{\mu}}^+, \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}^+ \rangle + 2\ln\left(\frac{n_+}{n_-}\right)$

This special case is often called **Linear Discriminant Analysis** or else **Fisher's linear discriminant** (named after Sir Ronald Fisher)

Special Cases – III

Let us fix $p = 0.5$ as well as fix $\Sigma^+ = \Sigma^- = I_d$

In that case we predict $\hat{y}^t = 1$ only if $\|\mathbf{x}^t - \hat{\boldsymbol{\mu}}^+\|_2^2 \leq \|\mathbf{x}^t - \hat{\boldsymbol{\mu}}^-\|_2^2$

Note that this is exactly LwP since even LwP learnt $\hat{\boldsymbol{\mu}}^+, \hat{\boldsymbol{\mu}}^-$ using averages of the data points with those labels ☺

If we fix $p = 0.5$ as well as $\Sigma^+ = \Sigma^- = \Sigma$ but not necessarily I_d, then also we get LwP but with a Mahalanobis distance instead!

If we had modelled class conditional distributions using a mixture of Gaussians, we would have obtained a classifier that looks similar to what we got LwP with multiple prototypes per class

General Case

In the most general case where each class gets its own separate Gaussian (not necessarily standard or spherical), the decision boundary is a quadratic function

Let $\widehat{\Lambda}^{+} \triangleq \left(\widehat{\Sigma}^{+}\right)^{-1}$ and $\widehat{\Lambda}^{-} \triangleq \left(\widehat{\Sigma}^{-}\right)^{-1}$ for notational simplicity

$\mathbf{x}^{\top} A\mathbf{x} + \mathbf{x}^{\top}\mathbf{b} + c \geq 0$ where $A = \widehat{\Lambda}^{-} - \widehat{\Lambda}^{+}, \mathbf{b} = 2\left(\widehat{\Lambda}^{+}\widehat{\boldsymbol{\mu}}^{+} - \widehat{\Lambda}^{-}\widehat{\boldsymbol{\mu}}^{-}\right)$

and $c = (\widehat{\boldsymbol{\mu}}^{-})^{\top}\widehat{\Lambda}^{-}\widehat{\boldsymbol{\mu}}^{-} - (\widehat{\boldsymbol{\mu}}^{+})^{\top}\widehat{\Lambda}^{+}\widehat{\boldsymbol{\mu}}^{+} + 2\ln\left(\frac{n_{+}}{n_{-}}\right) + \ln\frac{|\widehat{\Lambda}^{+}|}{|\widehat{\Lambda}^{-}|} \geq 0$

Historical note: although Fisher's linear discriminant is the name given to a special case (that assumes $\Sigma^{+} = \Sigma^{-} = \Sigma$), Fisher's original article (1936) did consider the general case (ref. Wikipedia)

Generative Learning with Missing Data

Suppose a test data point comes that has certain features missing

Assume for sake of simplicity that only the last k coordinates go missing i.e. $\mathbf{x}^t = [\mathbf{x}_o^t, \mathbf{x}_m^t] \in \mathbb{R}^d, \mathbf{x}_o^t \in \mathbb{R}^{d-k}, \mathbf{x}_m^t \in \mathbb{R}^k$ *(o = observed, m = missing)*

In such cases, generative models can help reconstruct the data point as well as classify it correctly – just use marginal probability!!

$\mathbb{P}[\mathbf{x}_o^t \mid y^t = c, \boldsymbol{\theta}] = \int_{\mathbb{R}^k} \mathbb{P}[[\mathbf{x}_o^t, \mathbf{v}] \mid y^t = c, \boldsymbol{\theta}]\, d\mathbf{v}$ *(law of total probability)*

All marginals of Gaussians are Gaussian. If $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ *then* $\mathbf{x}_o \sim \mathcal{N}(\boldsymbol{\mu}_o, \Sigma_{oo})$

Can use this to classify $\mathbf{x}_o^t$ *into some class, say* $\hat{y}^t$

Afterward, using the predicted class $\hat{y}^t$*, we can even fill in missing features*

This step is called **feature imputation** *in machine learning*

$$\hat{\mathbf{x}}_m^t = \arg\max_{\mathbf{v}} \mathbb{P}[\mathbf{v} \mid \mathbf{x}_o^t, \hat{y}^t, \boldsymbol{\theta}] = \boldsymbol{\mu}_m^{\hat{y}^t} + \Sigma_{mo}^{\hat{y}^t} \left(\Sigma_{oo}^{\hat{y}^t}\right)^{-1} \left(\mathbf{x}_o^t - \boldsymbol{\mu}_o^{\hat{y}^t}\right)$$

Turns out, conditionals of Gaussian are Gaussian too ☺

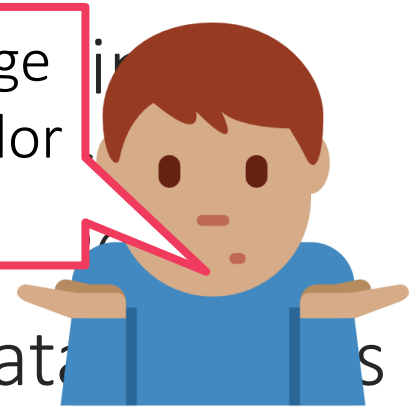
Generative Learning with Missing Data

Supp

A

$\mathbf{x}$

What if I don't know which coordinates are missing (e.g. a pixel in an image is white – is it because it was supposed to be white or is it because the color information for that pixel is missing). What can we do then?

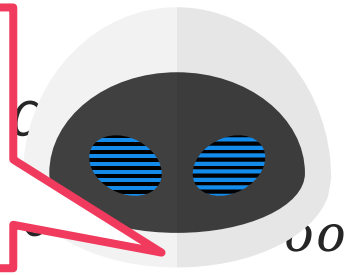


In such cases, generative models can help reconstruct the data as well as classify it correctly – just use marginal probability!

$\mathbb{P}[\mathbf{x}_o^t \mid y^t =$

All marginals

All these questions can be dealt with (to some extent) but require more sophisticated methods. Entire sub-areas of ML e.g. adversarial learning, robust learning are devoted to these



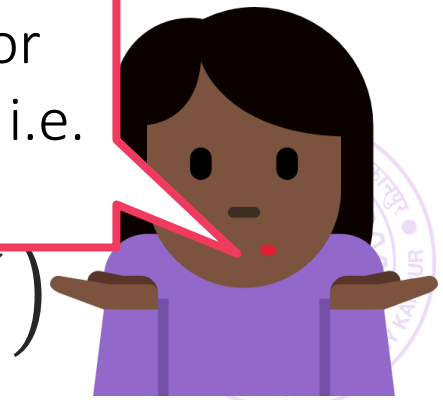
$\mathbf{x}_{oo})$

Can use this to classify $\mathbf{x}^t$ *into some class, say* $\hat{y}^t$

A

T

What if my training data is itself incomplete, e.g. has missing features or even completely wrong labels. What if these corruptions are adversarial i.e. the result of mischief designed to make the ML algo fail?



$$\hat{\mathbf{x}}_m^t = \arg\max_{\mathbf{v}} \mathbb{P}[\mathbf{v} \mid \mathbf{x}_o^t, \hat{y}^t, \boldsymbol{\theta}] = \boldsymbol{\mu}_m^{y^t} + \Sigma_{mo}^{y^t}\left(\Sigma_{oo}^{y^t}\right) \quad \left(\mathbf{x}_o^t - \boldsymbol{\mu}_o^{y^t}\right)$$

Turns out, conditionals of Gaussian are Gaussian too ☺