Exam 3 Review Day

Contents

- expectation maximization, k-means
- probability bounds
- KL divergence and cross entropy
- source coding
- logistic regression
- neural networks
- autoencoders, VAEs

Generative and Discriminative Models

posterior $\propto$ likelihood $\times$ prior

$$\widehat{y} = \arg\max_y p(y|\boldsymbol{x}) \qquad \text{MAP estimate}$$

$$\widehat{y} = \arg\max_y p(\boldsymbol{x}|y)\cancel{p(y)} \qquad \text{ML (maximum likelihood) estimate}$$

- we don't have $p(\boldsymbol{x}, y)$, instead we have $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
  - *generative* models
    - we imagine some distribution $p(\boldsymbol{x}|y)$ that *generates* our data

      example: imagine that $p(\boldsymbol{x}|y) \sim \mathcal{N}(\boldsymbol{u}_i, \Sigma_i)$
  - *discriminative* models
    - discriminative classifiers directly find $p(y|\boldsymbol{x})$ to *discriminate* between classes.

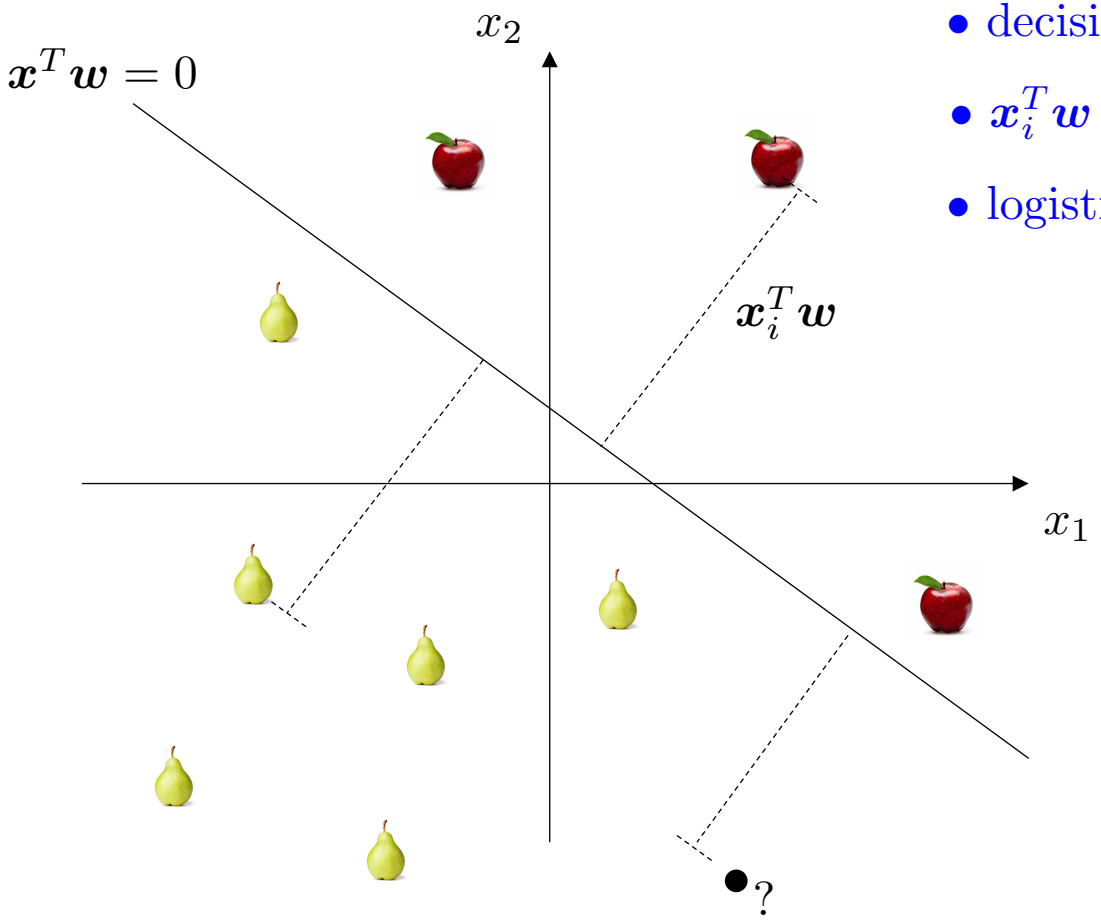
      directly fit $p(y|\boldsymbol{x})$ to data by treating $\boldsymbol{x}$ as non-random
    - we don't learn anything about the distribution of x

Two-Class Logistic Regression

- logistic regression is a *discriminative* classifier

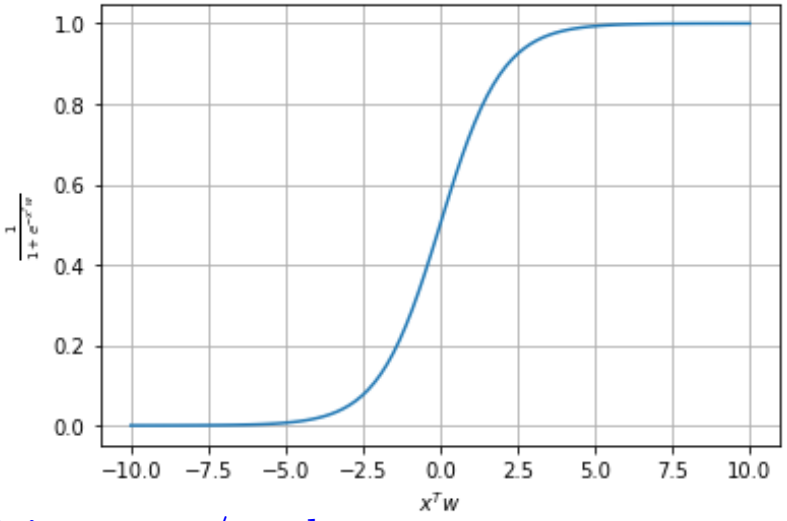
  directly fit $p(y|\boldsymbol{x})$ to data by treating $\boldsymbol{x}$ as non-random

$$p(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}^T\boldsymbol{w}}}$$



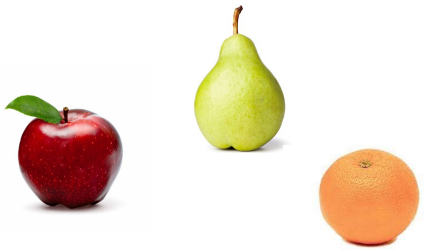
- decision boundary is $\boldsymbol{x}^T\boldsymbol{w}=0$
- $\boldsymbol{x}_i^T\boldsymbol{w}$ is the signed distance from the decision boundary
- logistic function takes $\boldsymbol{x}_i^T\boldsymbol{w}$ and squishes it to make it a probability

$$\frac{1}{1+e^{-\boldsymbol{x}_i^T\boldsymbol{w}}}$$



- probability that ? is a pear/apple depends distance from boundary

Multi-Class Logistic Regression



- logistic regression is a *discriminative* classifier

$$y_i = \begin{cases} 1 & \text{with probability } \theta_i \\ -1 & \text{with probability } 1-\theta_i \end{cases} \qquad \theta_i = \frac{1}{1+e^{-\boldsymbol{x}_i^T \boldsymbol{w}}} = \frac{e^{\boldsymbol{x}_i^T \boldsymbol{w}}}{e^{\boldsymbol{x}_i^T \boldsymbol{w}}+1}$$

- multiclass (multinomial) logistic regression:

$$y_i = \begin{cases} 1 & \text{with probability } \theta_{i,1}(\boldsymbol{x}_i) \\ 2 & \text{with probability } \theta_{i,2}(\boldsymbol{x}_i) \\ & \vdots \\ c & \text{with probability } \theta_{i,C}(\boldsymbol{x}_i) \end{cases} \qquad \theta_{i,c} = p(y=c|\boldsymbol{x}_i) = \frac{e^{\boldsymbol{x}_i^T \boldsymbol{w}_c}}{\sum_{c'=1}^{C} e^{\boldsymbol{x}_i^T w_{c'}}}$$

- softmax:

$$[\boldsymbol{\sigma}(\boldsymbol{z})]_i = \frac{e_i^z}{\sum_j e^{z_j}} \qquad \boldsymbol{\sigma}(\boldsymbol{z}) = \frac{e^{\boldsymbol{z}}}{\sum_j e^{z_j}}$$

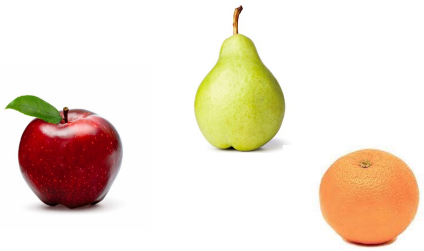
- softmax example:

```
[[-18.77687567]
 [39.26910416]
 [50.56209096]
 [47.72731802]
 [34.95432764]]
```

→

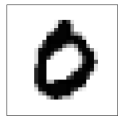
```
[[0.    ]
 [0.    ]
 [0.9445]
 [0.0555]
 [0.    ]]
```

Multi-Class Logistic Regression, Single Layer

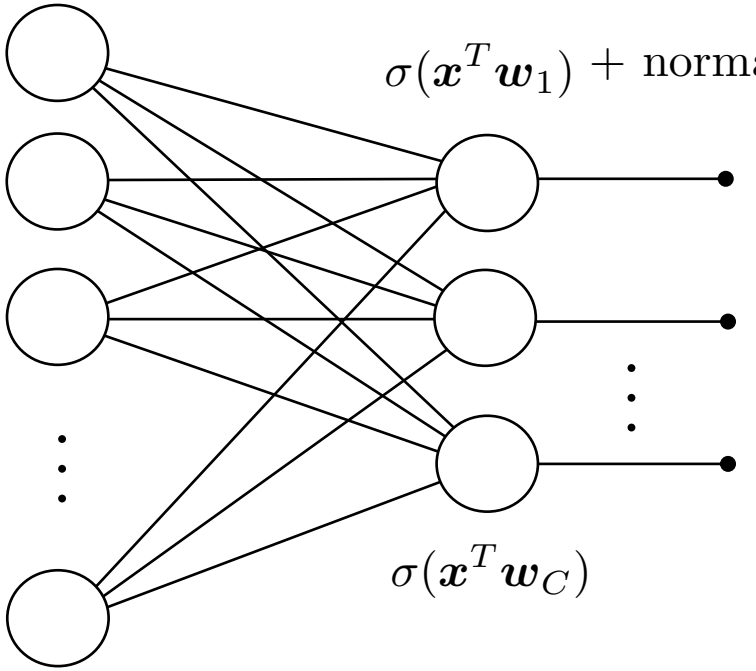


- logistic regression is a single *single layer of artificial neural network*

multiple neurons, single layer: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}^T \boldsymbol{x})$

$\boldsymbol{x}_i =$ 

$$\boldsymbol{y}_i = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$



$$\widehat{\boldsymbol{y}}_i = \begin{bmatrix} 0.92 \\ 0.01 \\ \vdots \\ 0.03 \end{bmatrix}$$

- loss

$$\ell(\widehat{\boldsymbol{y}}_i, \boldsymbol{y}_i) = D(\boldsymbol{y}_i || \widehat{\boldsymbol{y}}_i)$$

- training: gradient descent/Newton's method
- objective is convex

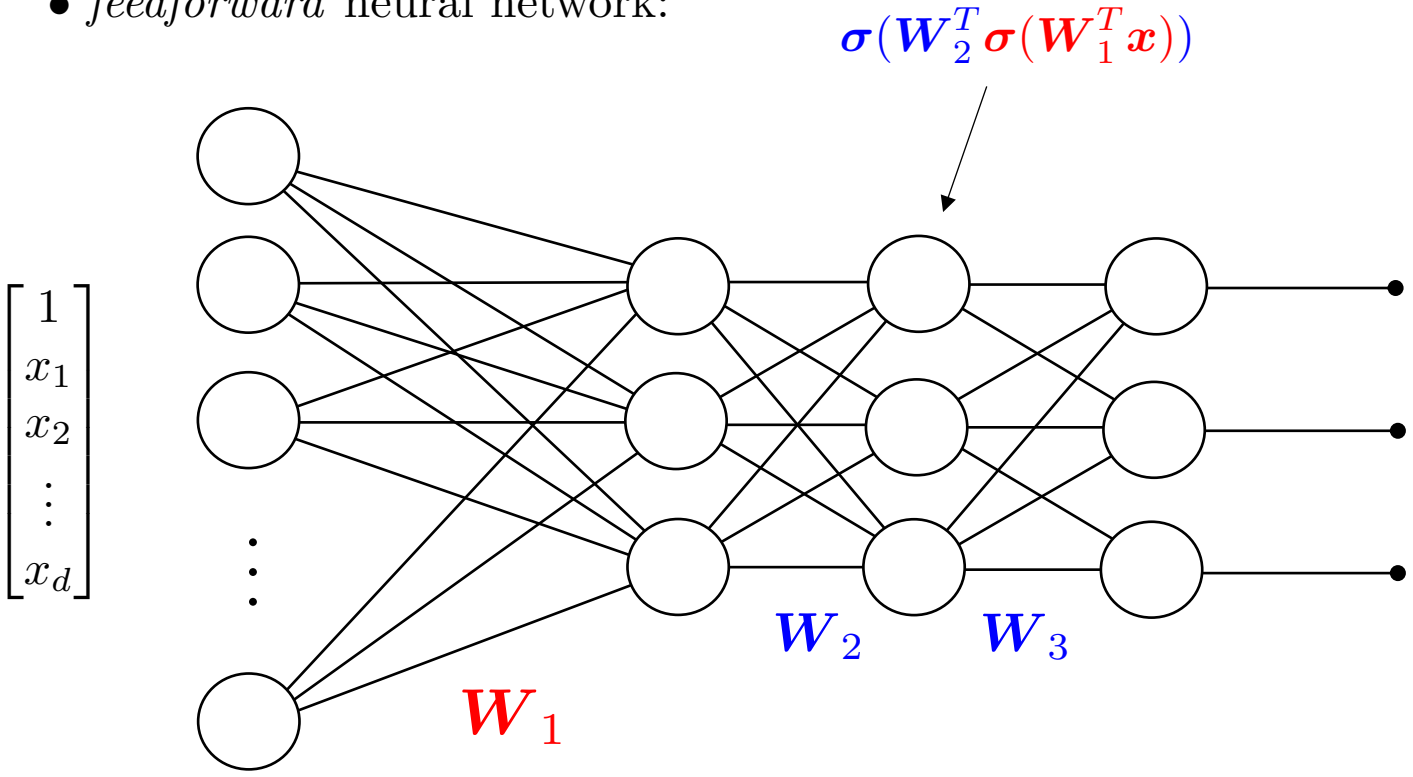
Artificial Neural Networks and Logistic regression

- binary logistic regression:
  - single *artificial neuron*
- multiclass logistic regression:
  - single layer *artificial neural network*
- *feedforward* neural network:

single layer: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}^T \boldsymbol{x})$

two layers: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}_2^T \boldsymbol{\sigma}(\boldsymbol{W}_1^T \boldsymbol{x}))$

ℓ-layers: $\widehat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{W}_\ell^T \dots \boldsymbol{\sigma}(\boldsymbol{W}_2^T \boldsymbol{\sigma}(\boldsymbol{W}_1^T \boldsymbol{x})) \dots)$



- softmax:

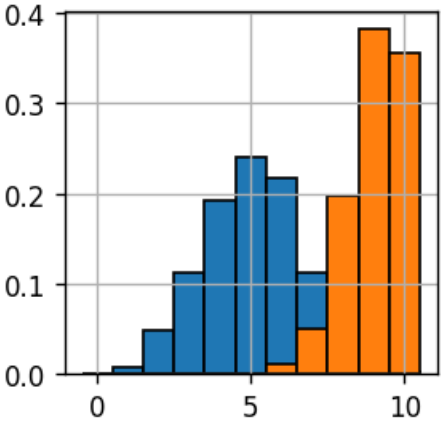
$$[\boldsymbol{\sigma}(\boldsymbol{z})]_i = \frac{e^{z}_i}{\sum_j e^{z_j}}$$

$$\boldsymbol{\sigma}(\boldsymbol{z}) = \frac{e^{\boldsymbol{z}}}{\sum_j e^{z_j}}$$

- logistic (unnormalized):

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

KL Divergence, Entropy, Cross-Entropy



- entropy:

$$H(\boldsymbol{p}) = \sum_i p_i \log \frac{1}{p_i}$$

- number of bits required to encode a sequences of i.i.d. random variables

- cross-entropy:

$$H(\boldsymbol{p}, \boldsymbol{q}) = \sum_i p_i \log \frac{1}{q_i}$$

- number of bits required to encode a sequences of i.i.d. random variables from $\boldsymbol{p}$ using a code optimized for $\boldsymbol{q}$.
- $H(\boldsymbol{p}, \boldsymbol{q})$ is big if $q_i \ll p_i$ for some i

- Kullback-Leibler Divergence:

$$D\left(\boldsymbol{p}||\boldsymbol{q}\right) = \sum_i p_i \log \frac{p_i}{q_i}$$

- distance between $\boldsymbol{p}$ and $\boldsymbol{q}$
- important in binary hypothesis testing
- KL divergence is non-negative: $D(p||q) \geq 0$ and $D(p||q) = 0$ if and only if $p(x) = q(x)$ for all x

- relationship:

$$H(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}) + D(\boldsymbol{p}||\boldsymbol{q})$$

Source Coding

- Shannon's *source coding* theorem is fundamental to quantifying information
- applies to discrete data, i.e, discrete X
- interpretation:

> n i.i.d. random variables each with entropy $H(X)$ can be compressed into $nH(X)$ bits with negligible risk of loss of information as n grows. Conversely, if compressed into fewer than $nH(X)$ bits, it is almost certain that information will be lost
>
> [adapted from Cover and Thomas, 2006]

Probability Bounds

- imagine $X_i \overset{i.i.d.}{\sim} \mathcal{N}(\mu, \sigma^2)$
- we're interested in the sample mean $\widehat{\mu} = \frac{1}{n}\sum_i X_i$

$$\widehat{\mu} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

- how good is our estimate μ?

$$\mathbb{P}(|\widehat{\mu} - \mu| \geq t) = 2\int_t^\infty \frac{1}{\sqrt{2\pi\frac{\sigma^2}{n}}} e^{-\frac{x^2}{2\sigma^2/n}} dx$$

$$\leq 2e^{-t^2 n/\sigma^2}$$

- *Hoeffding's Inequality.* Let $X_1, X_2, \ldots, X_n$ be i.i.d. bounded random variables with $X_i \in [a, b]$ and define $\widehat{\mu} = \frac{1}{n}\sum X_i$, $\mu = E[X_i]$. Then

$$\mathbb{P}(|\widehat{\mu} - \mu| \geq t) \leq 2e^{-\frac{2t^2 n}{(b-a)^2}}$$

- Hoeffding gives a tail bound that is basically like knowing the distributions are normal

Source Coding

$X_i \overset{i.i.d.}{\sim} \text{bernoulli}(\theta)$ $\quad\quad$ $\boldsymbol{x} \sim \text{bernoulli}(\theta)$

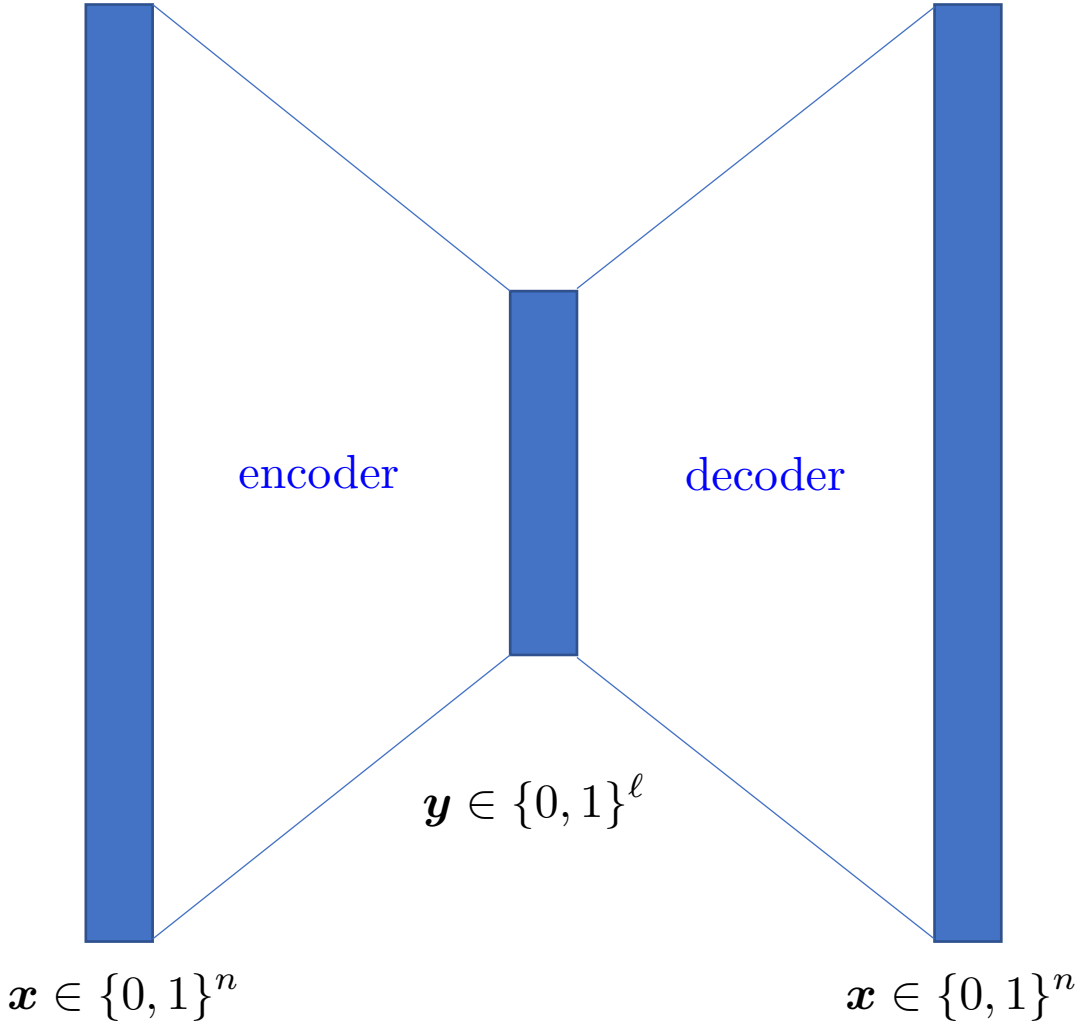
$$\mathbb{P}(|\widehat{\theta} - \theta| \geq \delta) \leq 2e^{-2\delta^2 n}$$

$$\mathbb{P}(|k - n\theta| \geq m) \leq 2e^{-2m^2/n}$$

- only need to encode vectors that have between $n\theta - m$ and $n\theta + m$ ones.

$$\sum_{k=\theta n - m}^{\theta n + m} \binom{n}{k} \approx 2m \binom{n}{\theta n} \approx 2^{\log_2(2m)} \times 2^{nH(X)}$$

- compression ratio $\approx H(X)$
- crux of argument was showing $\binom{n}{\theta n} \approx 2^{nH(X)}$



Unsupervised Learning

- *supervised* machine learning is about **learning functions from data**

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

- *unsupervised* machine learning is about finding interesting patterns in data

$$\mathcal{D} = \{(x_i)\}_{i=1}^{n}$$

  - *knowlege discovery*
  - under-specified

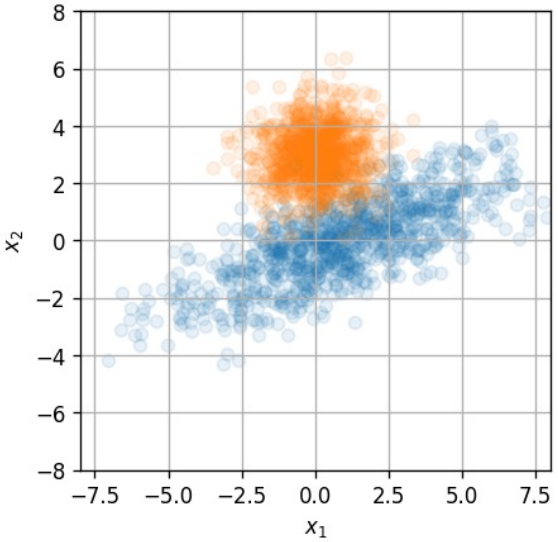
  - learn $p(\boldsymbol{x})$ instead of $p(\boldsymbol{x}, y)$ or $p(y|\boldsymbol{x})$
  - learn $p_\theta(\boldsymbol{x})$

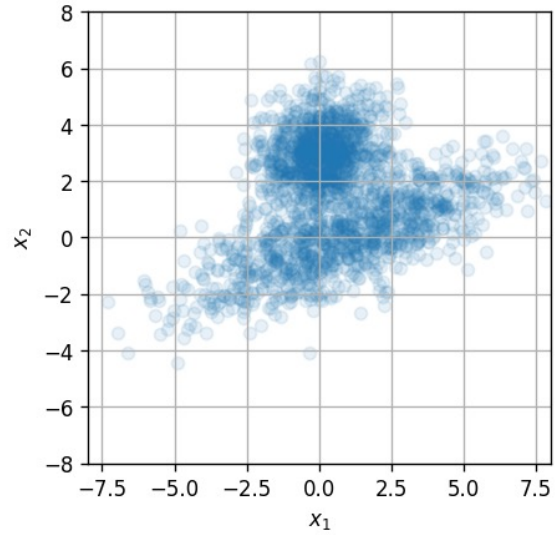
- examples:

  clustering

  density estimation

  community detection in graphs





Latent Variable Models

- goal: learn $p(\boldsymbol{x})$ from $\mathcal{D} = \{(\boldsymbol{x}_i)\}_{i=1}^n$

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, z) dz \qquad p(\boldsymbol{x}) = \sum_z p(\boldsymbol{x}, z)$$

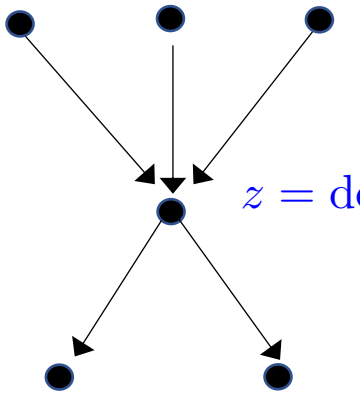
$$p(\boldsymbol{x}) = \int p(z) p(\boldsymbol{x}|z) dz \qquad p(\boldsymbol{x}) = \sum_z p(z) p(\boldsymbol{x}|z)$$

  - z is a *latent* variable

- *might* be mathematically convenient
- maybe $p(\boldsymbol{x}|z)$ has a simple form
- z might be meaningful, or maybe just convenient
- MNIST: z = digit, pen width, slant, etc.
- *latent* means hidden or concealed

  - if z is discrete, then *mixture* model

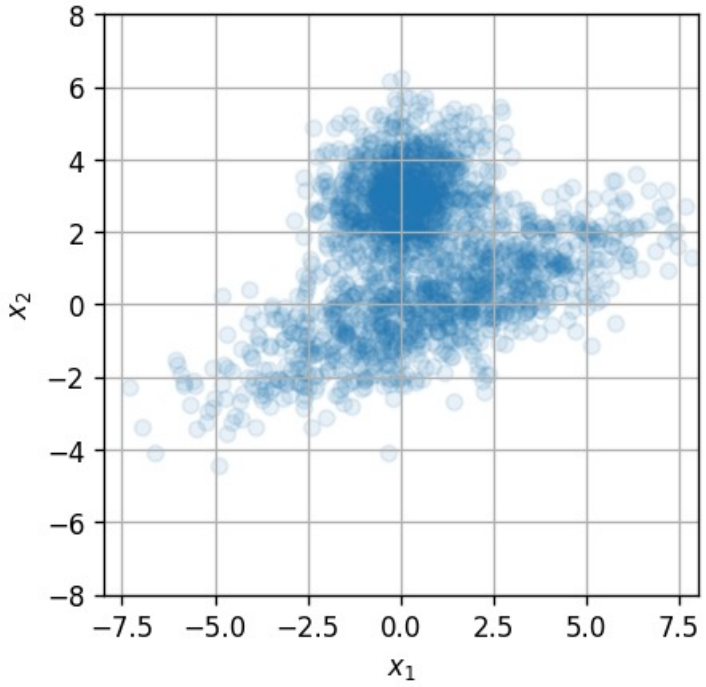
M/F, age, income, etc.



z = democrat or republican

how they vote in two elections

Expectation Maximization (EM) for GMMs



initialize means: $\boldsymbol{\mu}_k,\ j = 1, \ldots, K$

initialize covariances: $\boldsymbol{\Sigma}_k = \boldsymbol{I}$

initialize priors: $\pi_k = \frac{1}{K}$

E-step: 1. compute the responsibility of each point to each cluster (*soft* assign each point):

$$p(z = k|\boldsymbol{x}_i) = r_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(\boldsymbol{x}_i|\mu_{k'}, \Sigma_{k'})}$$

M-step: 2. estimate prior, mean and covariance for each mixture component

$$\boldsymbol{\mu}_k = \frac{1}{\sum_i r_{ik}} \sum_i r_{ik} \boldsymbol{x}_i \qquad \boldsymbol{\Sigma}_k = \frac{1}{\sum_i r_{ik}} \sum_i r_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T$$

$$\pi_k = \frac{1}{N} \sum_i r_{ik}$$

3. repeat 1, 2

Variational Inference

- goal: learn $p(\boldsymbol{z}|\boldsymbol{x})$

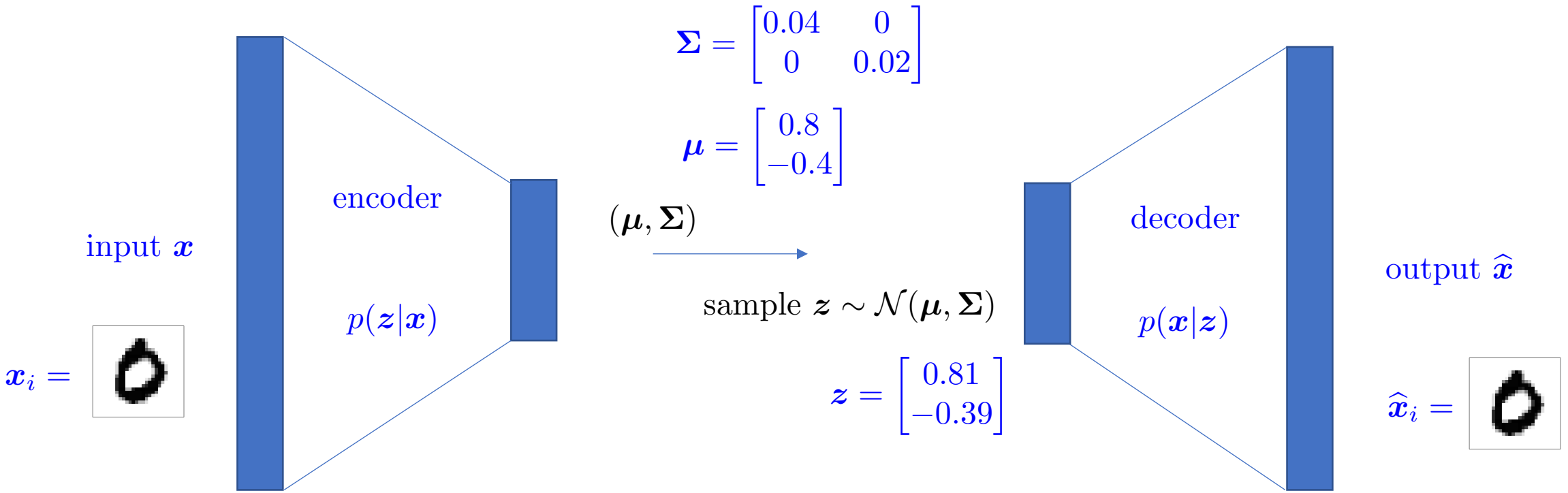
$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}$$

- $p(\boldsymbol{z}|\boldsymbol{x})$ is the posterior over the hidden variable $\boldsymbol{z}$

- variational inference: estimate $p(\boldsymbol{z}|\boldsymbol{x})$ using some other distribution $q(\boldsymbol{z}|\boldsymbol{x})$

- restrict $q(\boldsymbol{z}|\boldsymbol{x})$ to be from some tractable family, i.e, $\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x}))$

Variational Autoencoders



$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = ||\hat{\boldsymbol{x}} - \boldsymbol{x}||^2 + D(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(0, \boldsymbol{I}))$$

- ensure that $\boldsymbol{x} \approx \hat{\boldsymbol{x}}$
- try to force $p(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

Teaching Evaluations

https://heliocampusac.wisc.edu/