

CX4240 Spring 2026

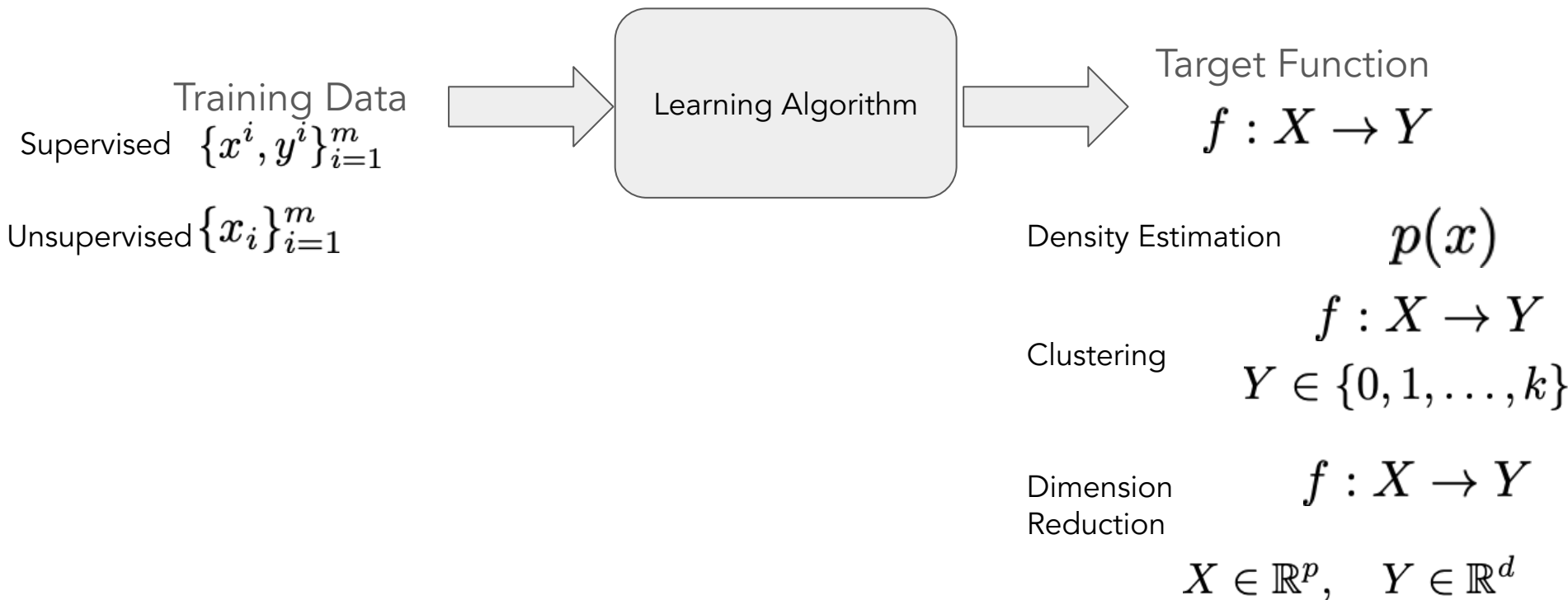
Dimension Reduction

Bo Dai
School of CSE, Georgia Tech
bodai@cc.gatech.edu

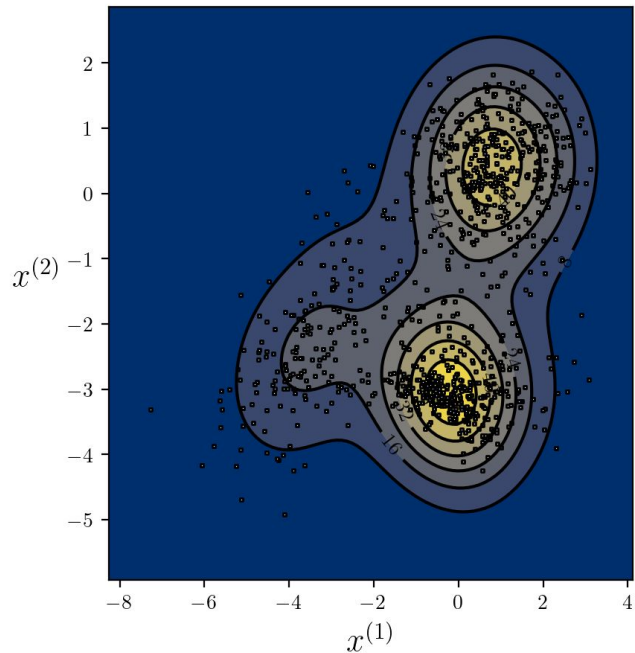
Logistics

- The [practice exam](#) has been released on **March 6th** on the course website.
- One exam will be held on **March 18th** in lieu of the regular class.
- One exam for students, who have registered with the ODS and approved, will be held on **March 18th** in **CODA C1315**
- We will have a makeup exam on **March 17th in CODA C1315**
 - You need an approval of absence to register this makeup exam
 - Please register the makeup exam by **March 16th (sending email to instructors and TAs)**.
- For students attending exam in CODA, you may need the building access. If so, please gather at **2nd floor in CODA on 3:15pm** and TA will lead you to the room.

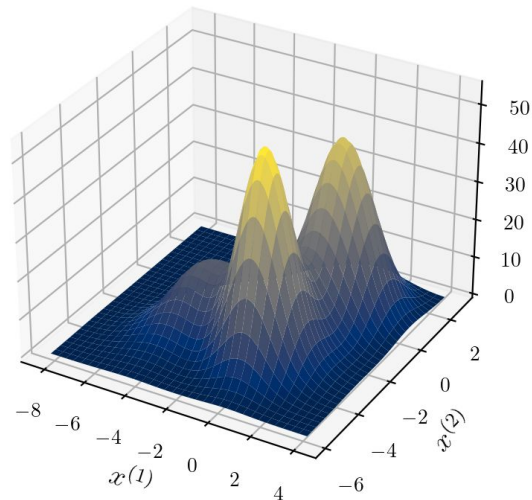
Supervised Learning vs. Unsupervised Learning



Density Estimation



$$\{x_i\}_{i=1}^m$$

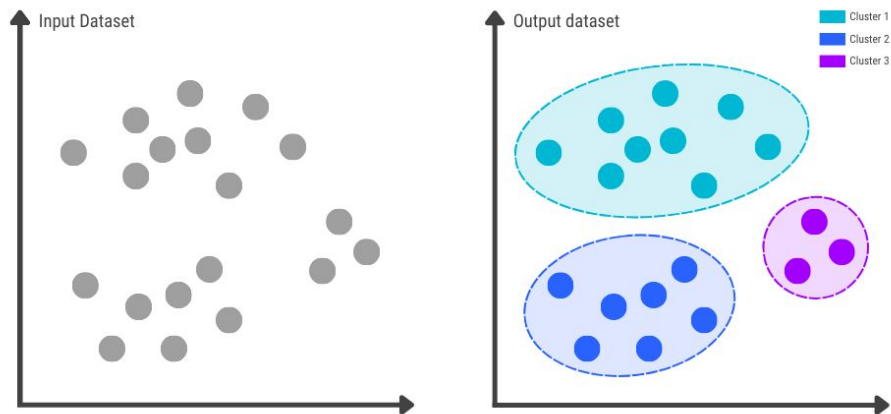


$$p(x)$$

Generative Models

$$x \sim p(x)$$

Clustering

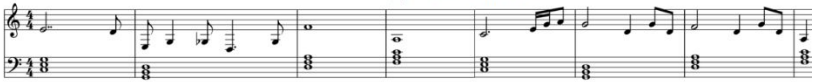


Density Estimation: Generative Models

$$x \sim p(x)$$



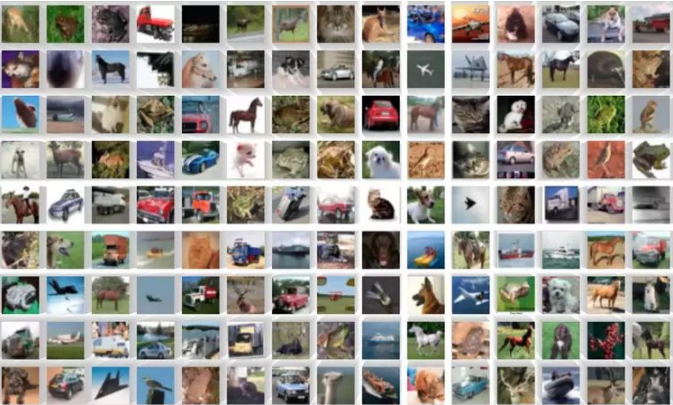
(a) MidiNet model 1



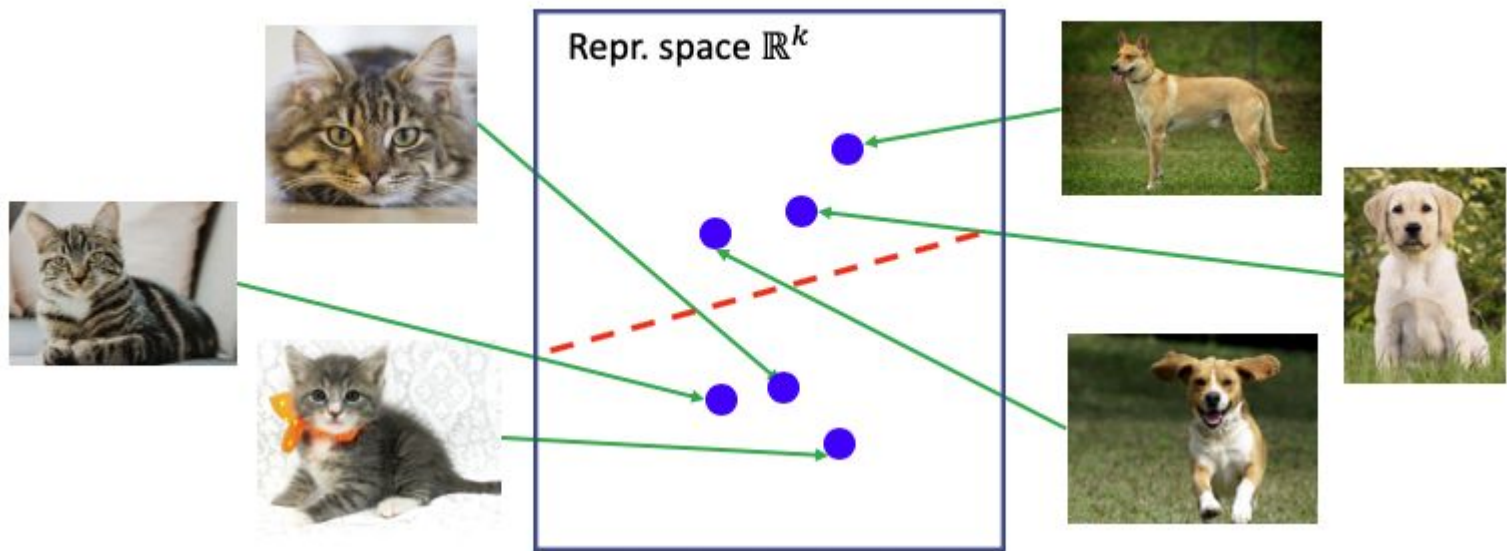
(b) MidiNet model 2



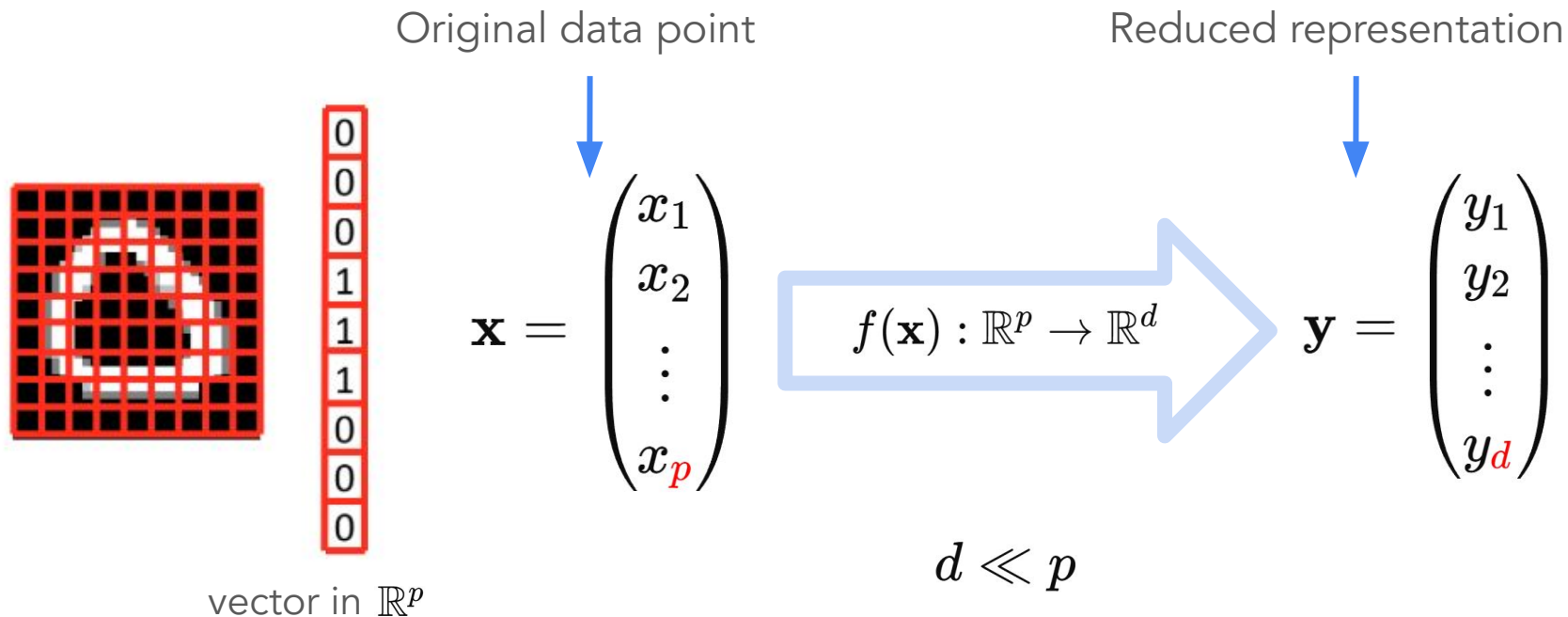
(c) MidiNet model 3



Dimension Reduction/Representation Learning

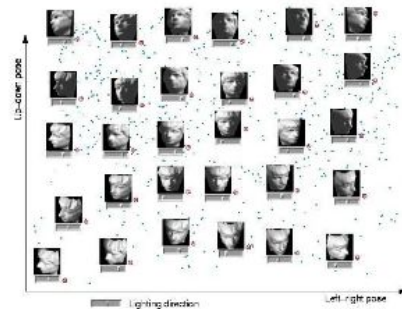
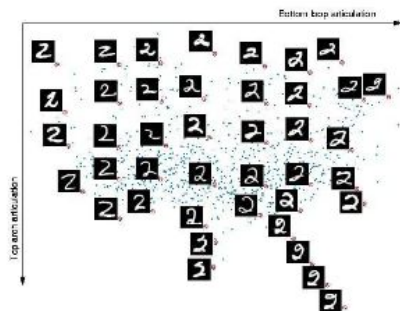


Dimension Reduction/Representation Learning



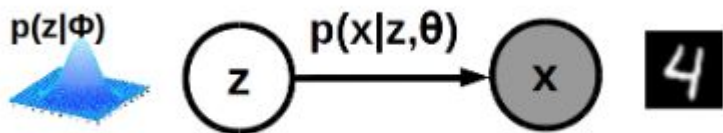
Why Dimension Reduction

- To compress data by reducing dimensionality. E.g., representing each image in a large collection as a linear combination of a small set of “template” images
- Visualization (e.g., by projecting high-dim data to 2D or 3D)



- To exploit the massive unlabelled data
- To make learning algorithms run faster
- To reduce overfitting problem caused by high-dimensional data

Revisit Latent Variable Models



$$p(x) = \int p(x|z)p(z)dz$$



Figure 5: 1024×1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

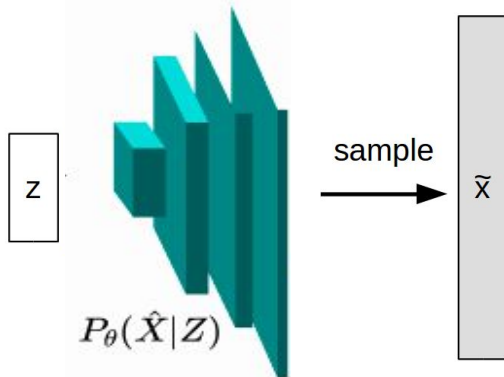
Deep Gaussian Distribution

Gaussian Distribution

$$\begin{aligned} p(\mathbf{x} | \mathbf{z}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_z)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_z)^T \boldsymbol{\Sigma}_z^{-1}(\mathbf{x} - \boldsymbol{\mu}_z)\right) \end{aligned}$$

Deep Gaussian Distribution

$$\begin{aligned} p(\mathbf{x} | \mathbf{z}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}(z), \boldsymbol{\Sigma}(z)) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}(z))}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}(z))^T \boldsymbol{\Sigma}(z)^{-1}(\mathbf{x} - \boldsymbol{\mu}(z))\right) \end{aligned}$$



deep neural
network

$$\mu_{W_\mu}(z), \sigma_{W_\sigma}(z)$$

Deep Latent Variable Models: Deep Gaussian LVM



$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \sigma I)$$

$$\begin{aligned} p(\mathbf{x} | \mathbf{z}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\Sigma}(\mathbf{z})) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}(\mathbf{z}))}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}(\mathbf{z}))^T \boldsymbol{\Sigma}(\mathbf{z})^{-1}(\mathbf{x} - \boldsymbol{\mu}(\mathbf{z}))\right) \end{aligned}$$

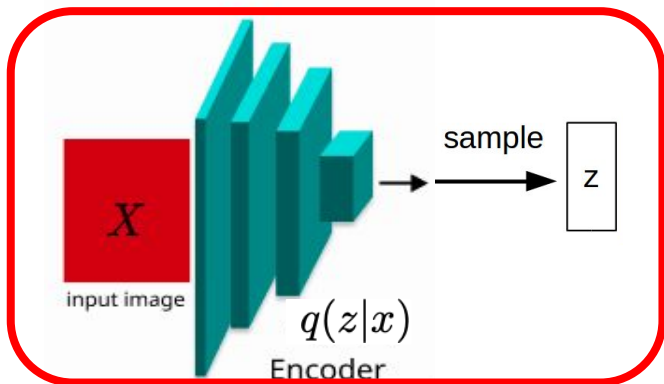
Model Parameters

$$\boldsymbol{\mu}_{W_\mu}(\mathbf{z}), \boldsymbol{\sigma}_{W_\sigma}(\mathbf{z}) \quad \boldsymbol{\sigma}$$

Evidence Lower Bound

$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^m \int q(z^i | x^i) \left(\log p(x^i | z^i) p(z^i) \right) dz^i$$

$$- \underbrace{\int q(z^i | x^i) \log q(z^i | x^i) dz_i}_{H(q(z|x))}$$



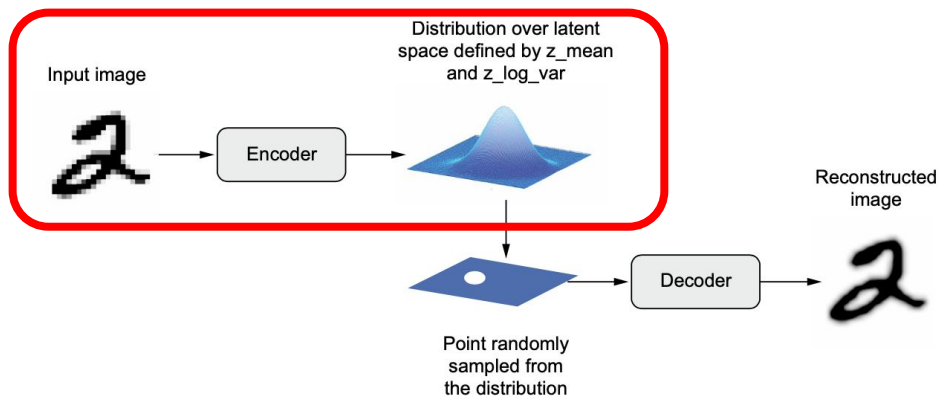
$$q(z|x) = \mathcal{N}(z | \mu_z(x), \text{diag}(\sigma_z^2(x)))$$

deep neural
network

$$\mu_{W_z}(x), \quad \sigma_{W_z}(x)$$

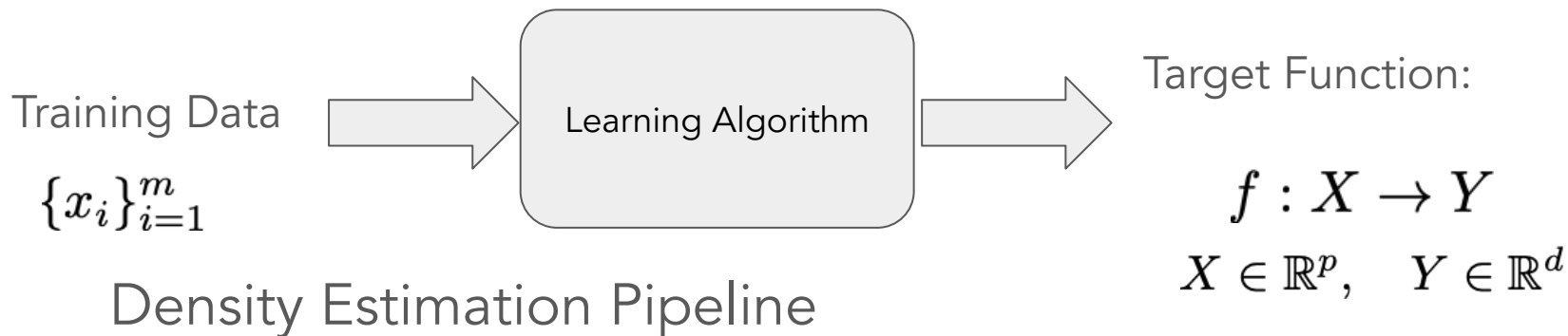
Encoder as Dimension Reduction

$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^m \int q(z^i | x^i) \left(\log p(x^i | z^i) p(z^i) \right) dz^i$$



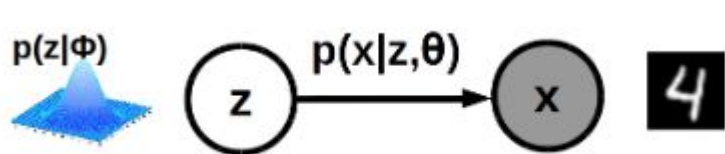
$$\underbrace{- \int q(z^i | x^i) \log q(z^i | x^i) dz^i}_{H(q(z|x))}$$

Dimension Reduction: Probabilistic Principal Component Analysis as LVM



1. Build probabilistic models
 Gaussian Latent Variable Model
2. Derive loss function (by MLE or MAP....)
3. Select optimizer

Gaussian LVM for Dimension Reduction



$$p(x) = \int p(x|z)p(z)dz$$

$$p(z) = \mathcal{N}(0, \sigma I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

Gaussian LVM for Dimension Reduction



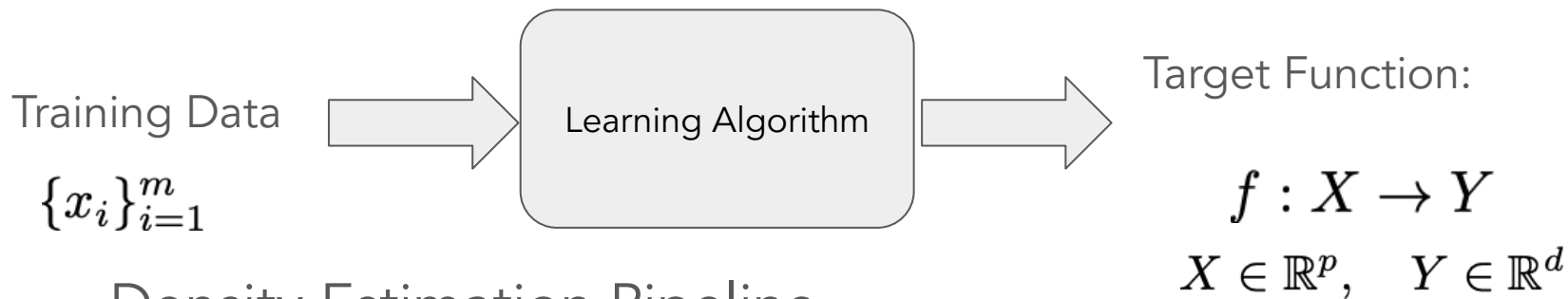
$$p(x) = \int p(x|z)p(z)dz$$

$$p(z) = \mathcal{N}(0, \sigma I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

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

Probabilistic Principal Component Analysis as LVM



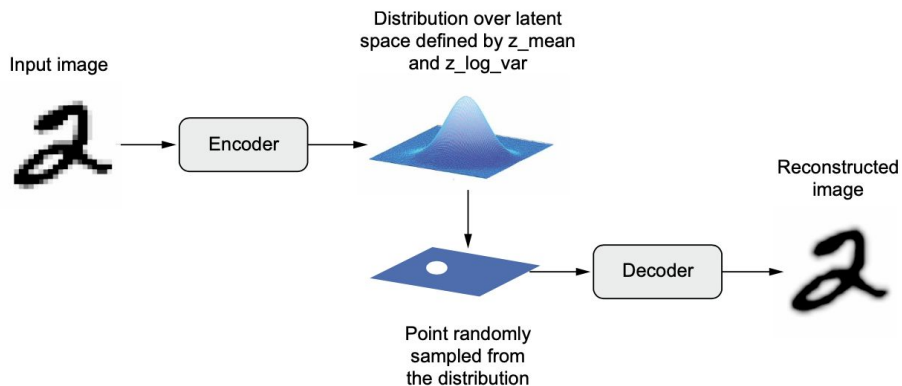
Density Estimation Pipeline

1. Build probabilistic models
Gaussian Latent Variable Model
2. Derive loss function (by MLE or MAP....)
ELBO approximation of MLE
3. Select optimizer

Revisit Evidence Lower Bound

$$\max_{q(z|x)} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^m \int q(z^i|x^i) \left(\log p(x^i|z^i) p(z^i) \right) dz^i$$

$$\underbrace{- \int q(z^i|x^i) \log q(z^i|x^i) dz^i}_{H(q(z|x))}$$



Gaussian LVM for Dimension Reduction



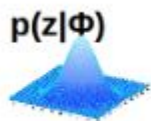
$$p(x) = \int p(x|z)p(z)dz$$

$$p(z) = \mathcal{N}(0, \sigma I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

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

Gaussian LVM for Dimension Reduction



$$p(z) = \mathcal{N}(0, \sigma I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

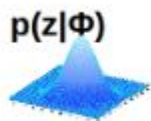
$$q(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

$$p(x) = \int p(x|z)p(z)dz$$

$$E[x] = E[Wz + \mu + \epsilon] = \mu$$

$$\begin{aligned} \text{Cov}[x] &= E[(Wz + \epsilon)(Wz + \epsilon)^T] = E[Wzz^T W^T] + \text{Cov}[\epsilon] \\ &= WE[zz^T]W^T + \text{Cov}[\epsilon] = WW^T + \sigma^2 I_D \end{aligned}$$

Gaussian LVM for Dimension Reduction



$$p(z) = \mathcal{N}(0, \sigma^2 I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

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

$$p(x) = \int p(x|z)p(z)dz$$

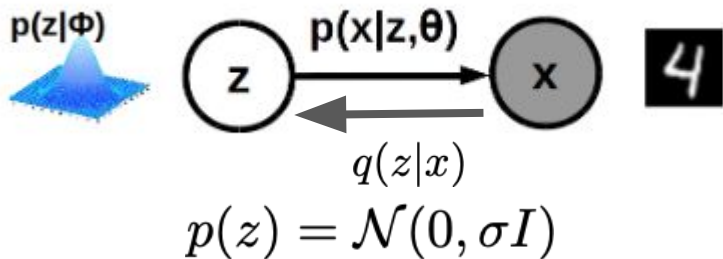
$$p(x) = \mathcal{N}(\mu, WW^T + \sigma^2 I)$$

$$E[x] = E[Wz + \mu + \epsilon] = \mu$$

$$\text{Cov}[x] = E[(Wz + \epsilon)(Wz + \epsilon)^T] = E[Wzz^T W^T] + \text{Cov}[\epsilon]$$

$$= WE[zz^T]W^T + \text{Cov}[\epsilon] = WW^T + \sigma^2 I_D$$

Gaussian LVM for Dimension Reduction



$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

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

$$p(x) = \int p(x|z)p(z)dz$$

$$p(x) = \mathcal{N}(\mu, WW^T + \sigma^2 I)$$

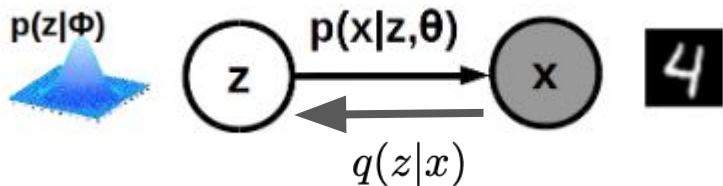
- The posterior mean is given by (see the tutorial)

$$E[z|x] = (W^T W + \sigma^2 I)^{-1} W^T (x - \mu)$$

- Posterior variance:

$$\text{Cov}[z|x] = \sigma^2 (W^T W + \sigma^2 I)^{-1}$$

Gaussian LVM for Dimension Reduction



$$p(x) = \int p(x|z)p(z)dz$$

$$p(z) = \mathcal{N}(0, \sigma I)$$

$$p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I)$$

$$q(z|x) = \frac{p(x|z)p(z)}{p(x)} = \mathcal{N}(MW^\top(x - \mu), \sigma^2 M) \quad M = (W^\top W + \sigma^2 I)^{-1}$$

Revisit MLE of LVM

$$p(x) = \int p(x|z)p(z)dz$$

$$\max_{\sigma, W_\mu, W_\sigma} \sum_{i=1}^m \log p(x^i) = \sum_{i=1}^m \log \int p(z)p(x^i|z)dz$$

MLE of Gaussian LVM

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \mathcal{N}(\mu, WW^\top + \sigma^2I)$$

MLE of Gaussian LVM

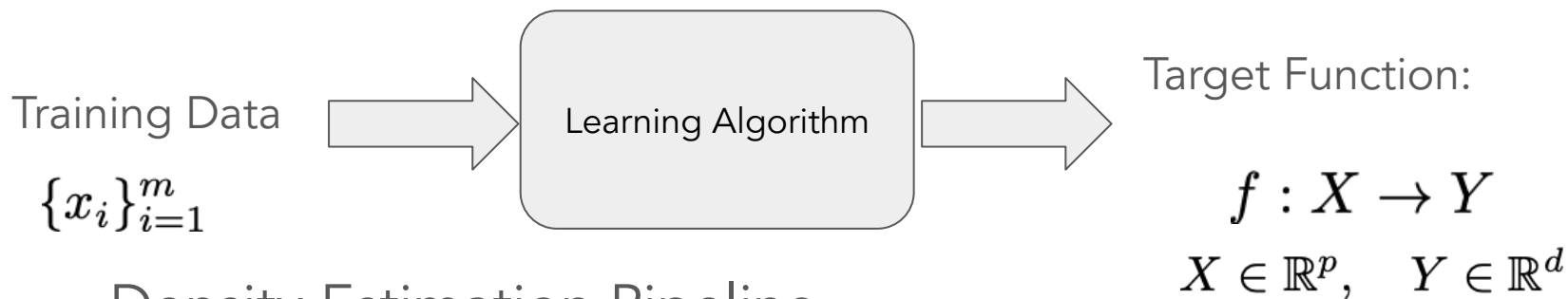
$$p(x) = \int p(x|z)p(z)dz = \mathcal{N}(\mu, WW^\top + \sigma^2I)$$

$$\max_{\sigma, W, \mu} \sum_{i=1}^m \log p(x^i)$$

$$M = (W^\top W + \sigma^2I)^{-1}$$

$$-\frac{mp}{2} \log(2\pi) - \frac{m}{2} \log \det(M^{-1}) - \frac{1}{2} \sum_{i=1}^m (x^i - \mu)^\top M (x^i - \mu)$$

Probabilistic Principal Component Analysis as LVM



Density Estimation Pipeline

1. Build probabilistic models
Gaussian Latent Variable Model
2. Derive loss function (by MLE or MAP....)
MLE
3. Select optimizer
Necessary Condition

$$\sum_{n=1}^N M(x_n - \mu) = 0 \quad \longrightarrow \quad \mu = \frac{1}{N} \sum_{n=1}^N x_n$$

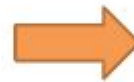
- The optimal parameters for the maximal log-likelihood are

Denote

$$S = S_{true} + S_{noise}$$

$$U \Lambda U^T = U_M \Lambda_M U_M^T + U_n \Lambda_n U_n^T$$

Covariance of Gaussian was $\Sigma_M = \text{diag}(\underbrace{\sigma^2, \dots, \sigma^2}_M, \underbrace{0, \dots, 0}_{D-M})$



$$\sigma^2 = \frac{1}{D - M} \sum_{j=M+1}^D \lambda_j$$

$$W W^T + \sigma^2 I = W W^T + \Sigma_M^2 + \Sigma_n^2$$

$$W = U_M (\Lambda_M - \sigma^2 I)^{1/2}$$

Covariance should match

$$U_M \Lambda_M U_M^T = W W^T + \Sigma_M^2$$

Probabilistic PCA

1, Compute $\mu = \frac{1}{N} \sum_{n=1}^N x_n$ $S = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)(x_n - \mu)^T$

2, Compute Eigen-decomposition

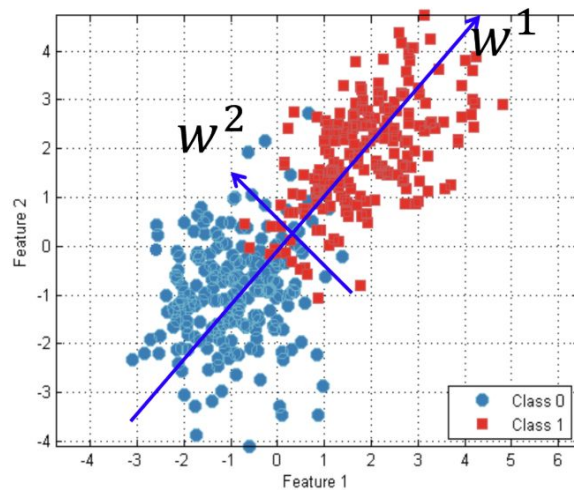
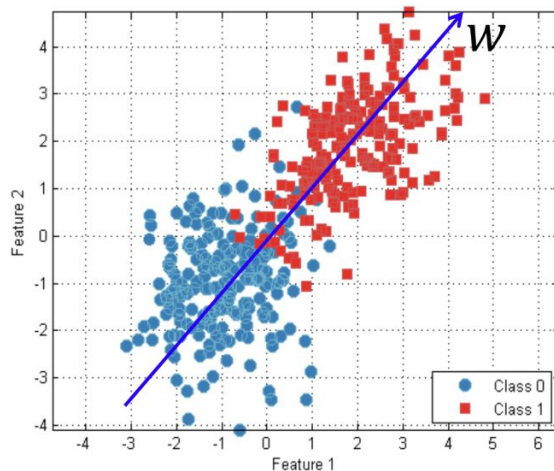
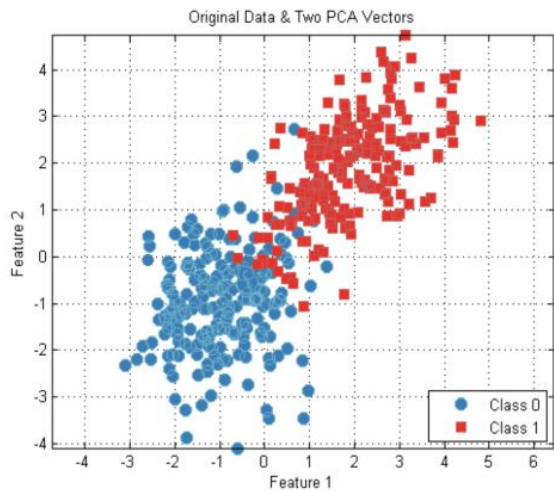
$$S = U_M \Lambda_M U^T + U_n \Lambda_n U_n^T$$

$$\sigma^2 = \frac{1}{D - M} \sum_{j=M+1}^D \lambda_j$$

3, Compute Projection

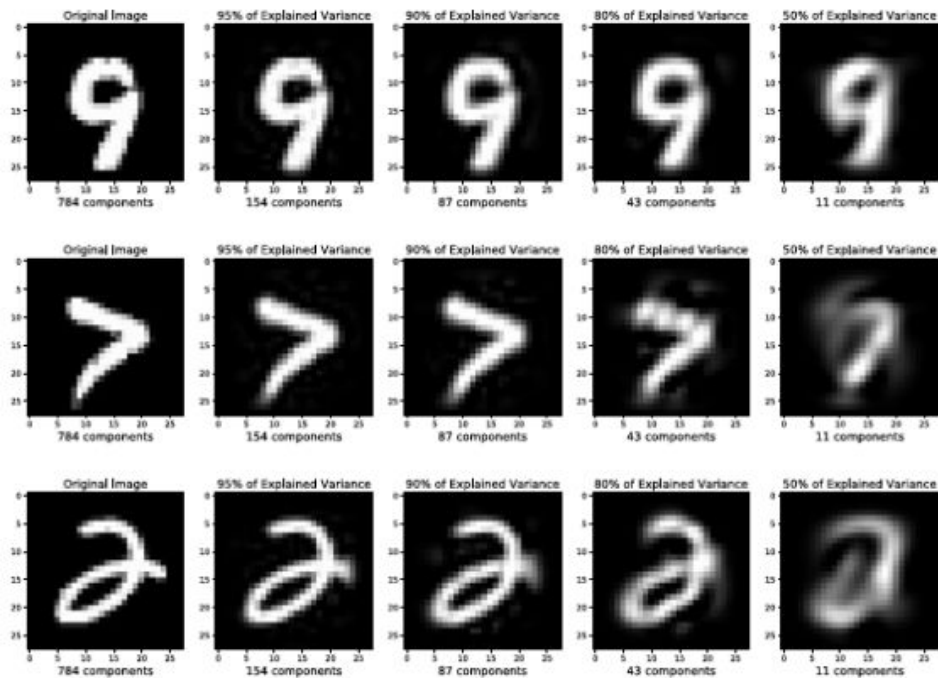
$$W = U_M (\Lambda_M - \sigma^2 I)^{1/2}$$

Illustration of PCA



Example: MNIST digits

- 28x28 images = 784 PCA vectors
- Project to K dimensional space and then project back up



Eigenfaces (Sirovich & Kirby 1987, Turk & Pentland 1991)

The first 9 PCA components



Original



PCA-reconstructed

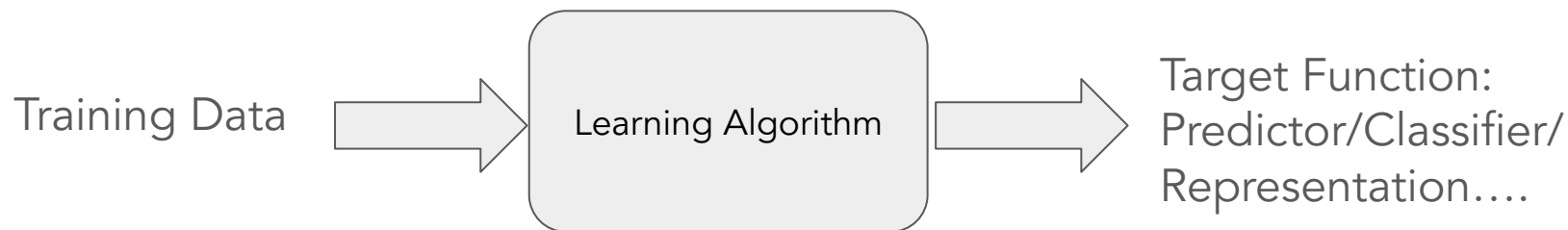


PCA: Summary

- Three views
 - Max variance, min reconstruction error, probabilistic
- Applications
 - Dimensionality reduction
 - 2D/3D visualization
 - Compression
 - Whitening (de-correlating features)
 - (not mentioned) De-noising: discard the smallest variance features = the noise components (hopefully!)
- Limitation
 - Only linear transformations

CX4240 Spring 2026 Review

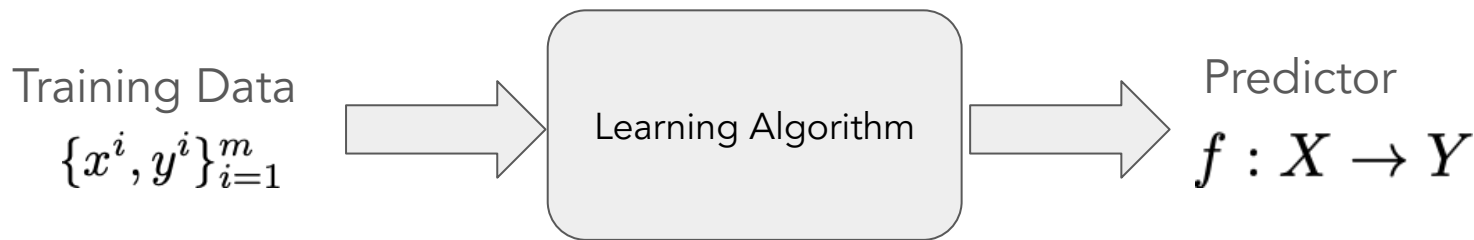
ML Algorithm Pipeline



General ML Algorithm Pipeline

1. Build probabilistic models
2. Derive loss function (by MLE or MAP....)
3. Select optimizer

Regression algorithms



General ML Algorithm Pipeline

1. Build probabilistic models:
Gaussian noise + linear model/polynomial model
2. Derive loss function:
MLE vs. MAP
3. Select optimizer
Necessary Condition vs. (Stochastic) GD

Probabilistic Model: Gaussian Likelihood

- Assume y is a linear in x plus noise ϵ

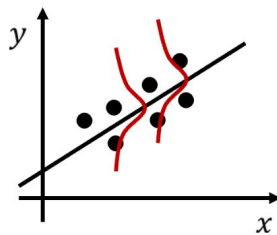
$$y = \theta^\top x + \epsilon$$

- Assume ϵ follows a Gaussian $N(0, \sigma)$ $\epsilon \sim \mathcal{N}(0, \sigma)$

$$\mathbb{E}[y] = \theta^\top x + \mathbb{E}[\epsilon] = \theta^\top x$$

$$y = \theta^\top x + \epsilon \sim \mathcal{N}(\theta^\top x, \sigma)$$

$$p(y^i | x^i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - \theta^\top x^i)^2}{2\sigma^2}\right)$$



Maximum log-Likelihood Estimation (MLE)

$$L(\theta) = \prod_i^m p(y^i|x^i; \theta) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m \exp\left(-\frac{\sum_i^m (y^i - \theta^\top x^i)^2}{2\sigma^2}\right)$$

$$\max_{\theta} \log L(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^m (y^i - \theta^\top x^i)^2 - m \log(\sqrt{2\pi}\sigma)$$

Maximum a Posteriori (MAP)

$$p(\theta) \propto \exp(-\lambda \|\theta\|_2^2)$$

Gaussian Prior

$$\max_{\theta} \log p(\theta | \{x^i, y^i\}_{i=1}^m) = \log L(\theta) + \log p(\theta)$$

Ridge Regression

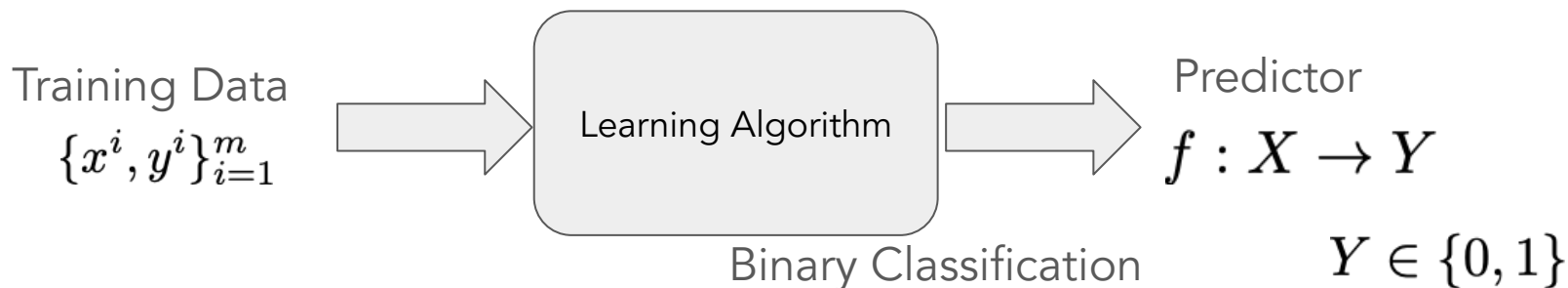
$$\propto -\frac{1}{m} \sum_{i=1}^m (y^i - \theta^\top x^i)^2 - \lambda \|\theta\|_2^2$$

Gradient Calculation

$$\min_{\theta} -\log L(\theta) \propto \frac{1}{m} \sum_{i=1}^m (y^i - \theta^\top x^i)^2$$

$$\frac{\partial \log L(\theta)}{\partial \theta} = -\frac{2}{m} \sum_{i=1}^m (y^i - \theta^\top x^i) x^{i\top};$$

Binary Classification Algorithms



Logistic Regression Pipeline

1. Build probabilistic models:
Bernoulli Distribution
2. Derive loss function:
MLE and MAP
3. Select optimizer:
(Stochastic) Gradient Descent

Probabilistic Model in Classification: Bernoulli Likelihood

- Logistic regression model

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

- Note that

$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

MLE

- Logistic regression model

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

- Note that

$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

- Plug in

$$\begin{aligned} l(\theta) &:= \log \prod_{i=1}^n p(y^i|x^i, \theta) && \text{(Bernoulli)} \\ &= \sum_{i=1}^n \log \left(\frac{\exp(-\theta^\top x^i)}{1 + \exp(-\theta^\top x^i)} \right) \underbrace{I(y^i = 0)}_{1-y^i} + \log \left(\frac{1}{1 + \exp(-\theta^\top x^i)} \right) \underbrace{I(y^i = 1)}_{y^i} \\ &= \sum_{i=1}^n (y^i - 1)\theta^\top x^i - \log(1 + \exp(-\theta^\top x^i)) \end{aligned}$$

MAP

Logistic regression model

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

Note that

$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

$$p(\theta) \propto \exp(-\lambda \|\theta\|_2^2)$$

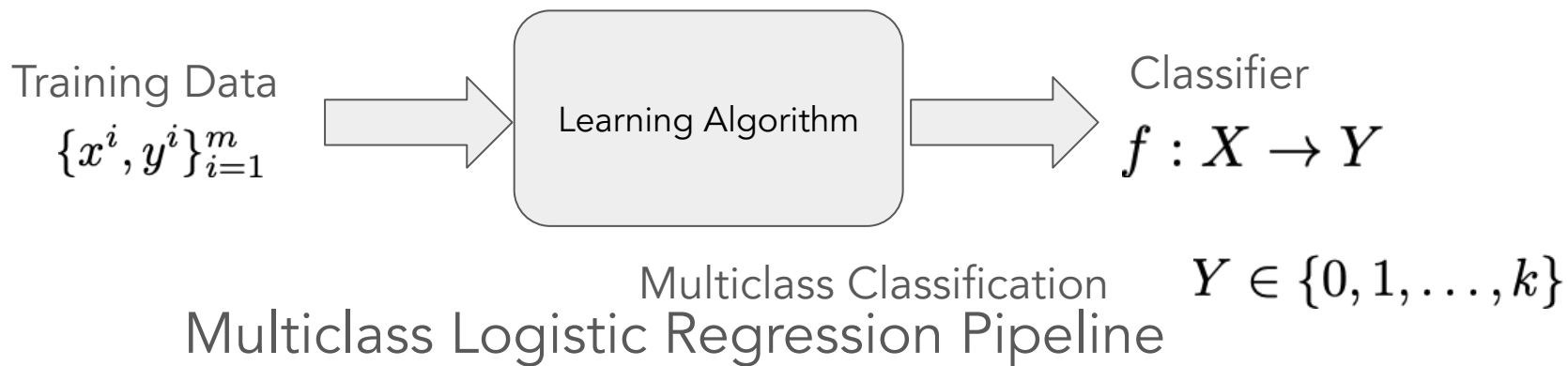
$$\begin{aligned} \max_{\theta} \log p(\theta | \{x^i, y^i\}_{i=1}^m) &= \log L(\theta) + \log p(\theta) \\ &= \sum_i (y^i - 1) \theta^\top x^i - \log(1 + \exp(-\theta^\top x^i)) - \lambda \|\theta\|_2^2 \end{aligned}$$

Gradient Calculation of MLE

$$\max_{\theta} \log L(\theta) = \sum_i (y^i - 1) \theta^\top x^i - \log(1 + \exp(-\theta^\top x^i))$$

$$\frac{\partial \log L(\theta)}{\partial \theta} = \sum_i (y^i - 1) x^i + \frac{\exp(-\theta^\top x^i) x^i}{1 + \exp(-\theta^\top x^i)}$$

Multiclass Logistic Regression Algorithms



1. Build probabilistic models:
Categorical Distribution + Linear Model
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Softmax Parametrization

$$p(y_i = 1 | \theta_i^\top x) \in (0, 1), \quad \sum_{i=1}^k p(y_i = 1 | \theta_i^\top x) = 1$$

Positivity $p(y_i = 1 | \theta_i^\top x) \propto \exp(\theta_i^\top x)$

Normalization $p(y_i = 1 | \theta_i^\top x) = \frac{\exp(\theta_i^\top x)}{\sum_{i=1}^k \exp(\theta_i^\top x)}$

MLE

- Given all input data $\{x^i, y^i\}_{i=1}^m$

$$p(y^i | \theta^\top x^i) = \prod_{j=1}^k p(y_j^i = 1 | \theta^\top x^i)^{y_j^i}$$

- Log-likelihood

$$\ell(\theta) = \sum_{i=1}^m \sum_{j=1}^k y_j^i \log p(y_j^i = 1 | \theta^\top x^i) \quad \text{cross-entropy}$$

$$= \sum_{i=1}^m \sum_{j=1}^k y_j^i \log \frac{\exp(\theta_j^\top x^i)}{\sum_{c=1}^k \exp(\theta_c^\top x^i)}$$

$$= \sum_{i=1}^m \sum_{j=1}^k y_j^i (\theta_j^\top x^i) - \sum_{i=1}^m \log \sum_{c=1}^k \exp(\theta_c^\top x^i)$$

MAP

- Likelihood

$$p(y = j|x, \theta) = \frac{\exp(\theta_j^\top x)}{\sum_{c=1}^k \exp(\theta_c^\top x)}$$

- Prior

$$p(\theta) \propto \exp(-\lambda \|\theta\|_2^2)$$

$$\begin{aligned} \max_{\theta} \log p(\theta | \{x^i, y^i\}_{i=1}^m) &= \log L(\theta) + \log p(\theta) \\ &= \sum_{i=1}^m \sum_{j=1}^k y_j^i \theta_j^\top x^i - \sum_{i=1}^m \log \sum_{c=1}^k \exp(\theta_c^\top x^i) - \lambda \|\theta\|_2^2 \end{aligned}$$

Gradient Calculation of MLE

$$\max_{\theta} \log L(\theta) = \sum_{i=1}^m \sum_{j=1}^k y_j^i \theta_j^\top x^i - \sum_{i=1}^m \log \sum_{c=1}^k \exp(\theta_c^\top x^i)$$

$$\frac{\partial \log L(\theta)}{\partial \theta} = \sum_{i=1}^m \sum_{j=1}^k (y_j^i - p(y_j^i = 1 | x^i, \theta)) x^i$$

$$p(y_j^i = 1 | x^i, \theta) = \frac{\exp(\theta_j^\top x^i)}{\sum_{c=1}^k \exp(\theta_c^\top x^i)}$$

Vector Formation

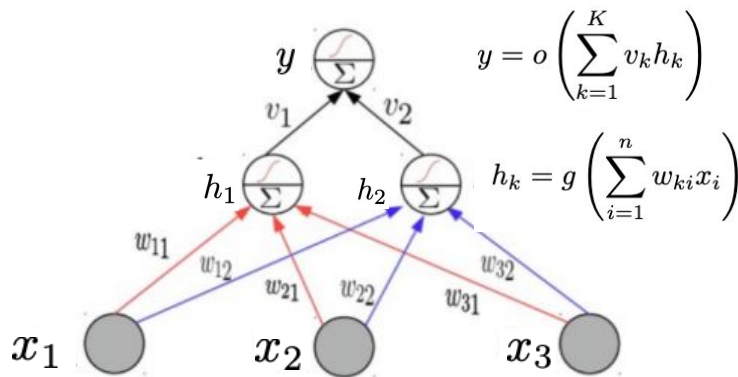
$$y = o(\mathbf{V}g(\mathbf{W}\mathbf{x}))$$

$$\mathbf{V} = [v_1, v_2]$$

$$\mathbf{h} = [h_1, h_2]^\top = g(\mathbf{W}\mathbf{x})$$

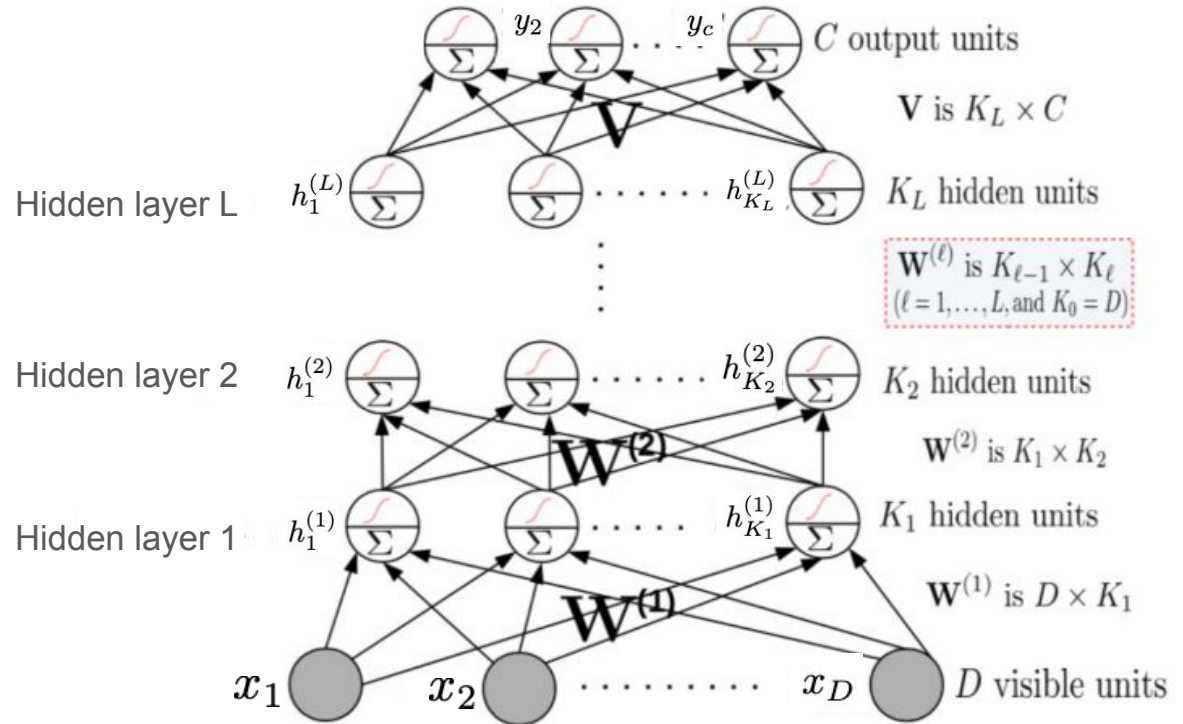
$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$$

$$\mathbf{x} = [x_1, x_2, x_3]^\top$$

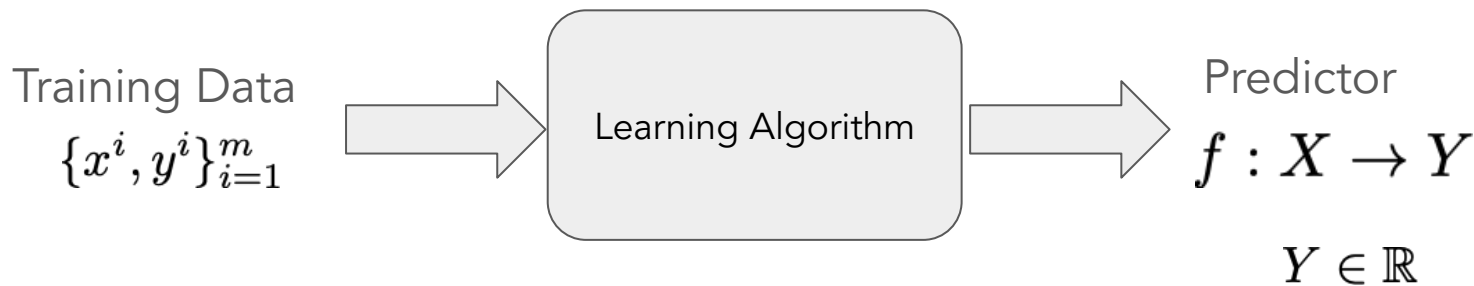


Multi-Layer Perception

$$y = f_L(\mathbf{W}_L f_{L-1}(\mathbf{W}_{L-1} \dots f_1(\mathbf{W}_1 \mathbf{x})))$$



Regression Algorithms

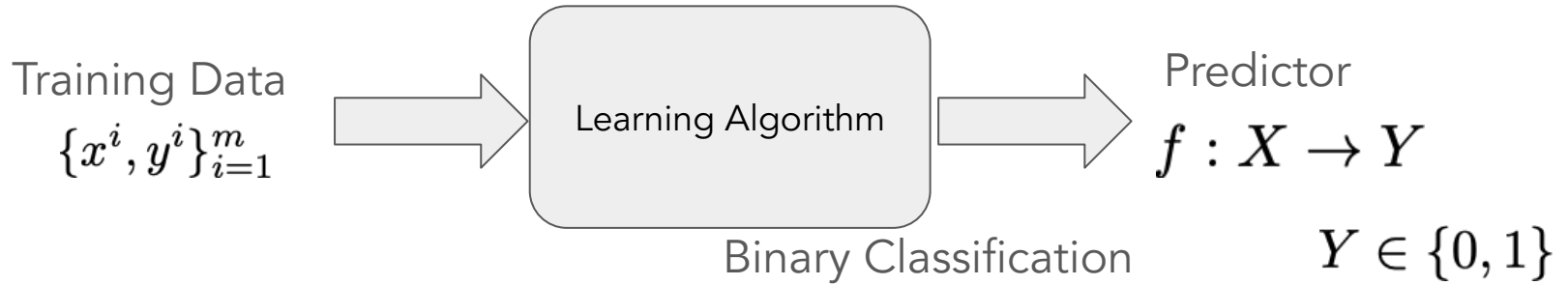


Linear Regression Pipeline

1. Build probabilistic models:
Gaussian Distribution + Neural Network
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) GD

$$y = o(\mathbf{V} g(\mathbf{W} x))$$

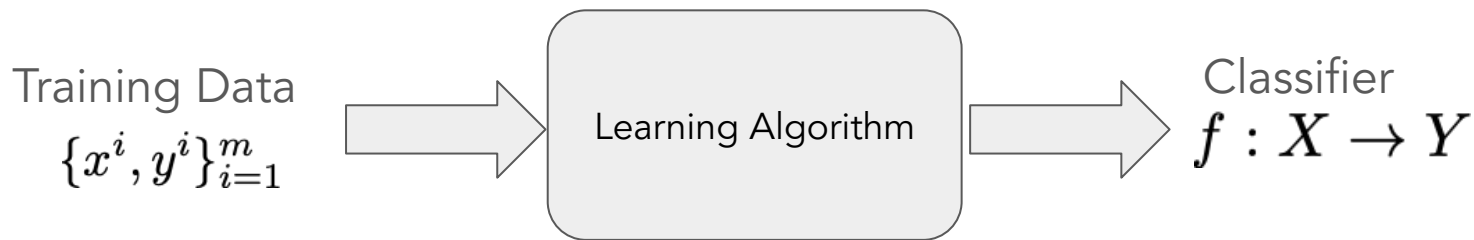
Binary Classification Algorithms



Binary Logistic Regression Pipeline

1. Build probabilistic models:
Bernoulli Distribution + Neural Network $y = o(\mathbf{V}g(\mathbf{W}x))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Multiclass Logistic Regression Algorithms



Multiclass Classification $Y \in \{0, 1, \dots, k\}$
Multiclass Logistic Regression Pipeline

1. Build probabilistic models:
Categorical Distribution + Neural Network $y = o(\mathbf{V}g(\mathbf{W}x))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Select Optimizer

$$L(\theta) = \sum_{i=1}^m \ell(x^i, y^i, \theta) + \lambda \Omega(\theta)$$
$$\theta = [V, W]$$

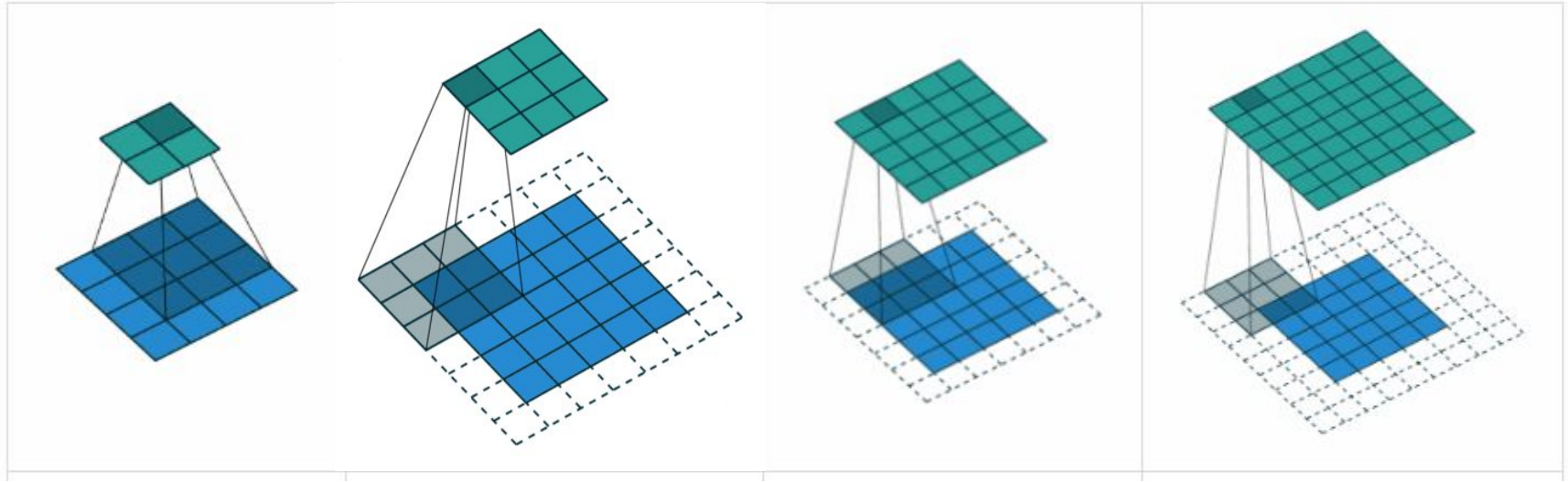
$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

$$\ell(x^i, y^i, \theta) = -y^i \log \sigma(o(Vg(Wx^i)))$$
$$-(1 - y^i) \log(1 - \sigma(o(Vg(Wx^i))))$$

$$\ell(x^i, y^i, \theta) = - \sum_{j=1}^k y^i \log \frac{\exp(o(V_j g(Wx^i)))}{\sum_{c=1}^k \exp(o(V_c g(Wx^i)))}$$

- (Stochastic) Gradient Descent

Convolution Operator



padding = 0, stride = 1

padding = 1, stride = 2

padding = 1, stride = 1

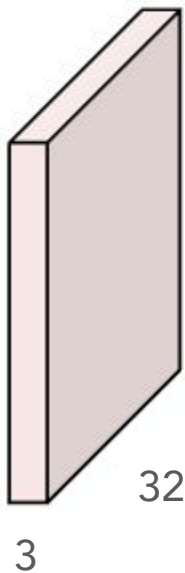
padding = 2, stride = 1

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

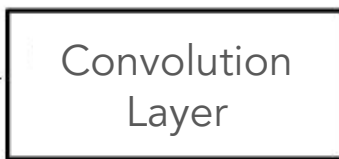
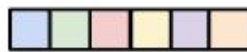
Animation from [Hochschule der Medien](https://www.hochschule-der-medien.de/)

Convolution Layer

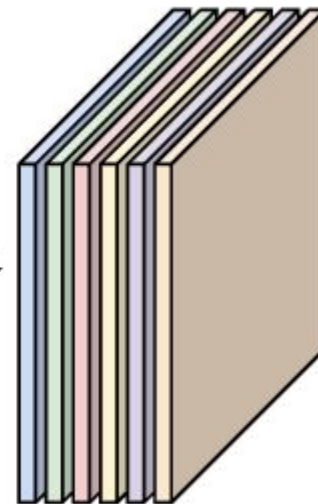
32x32x3 image



Don't forget bias terms!



6x3x5x5 filters

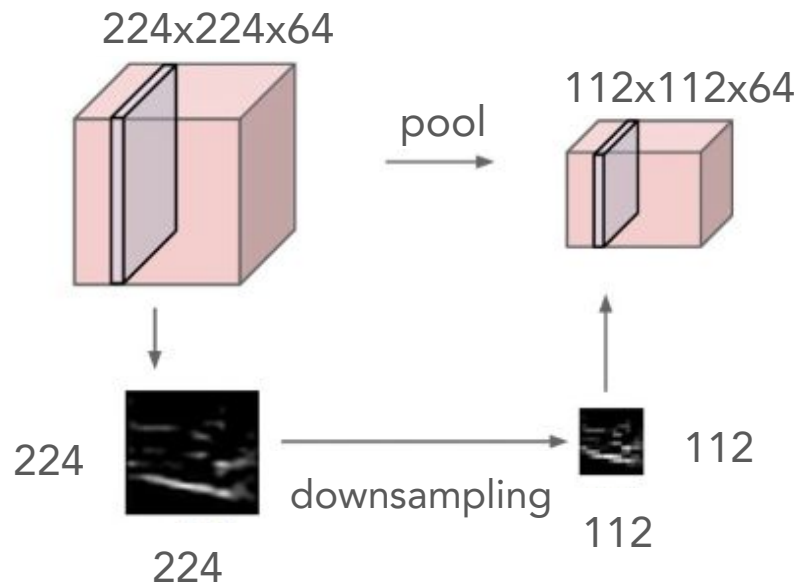


Activation Function!

(ReLU)

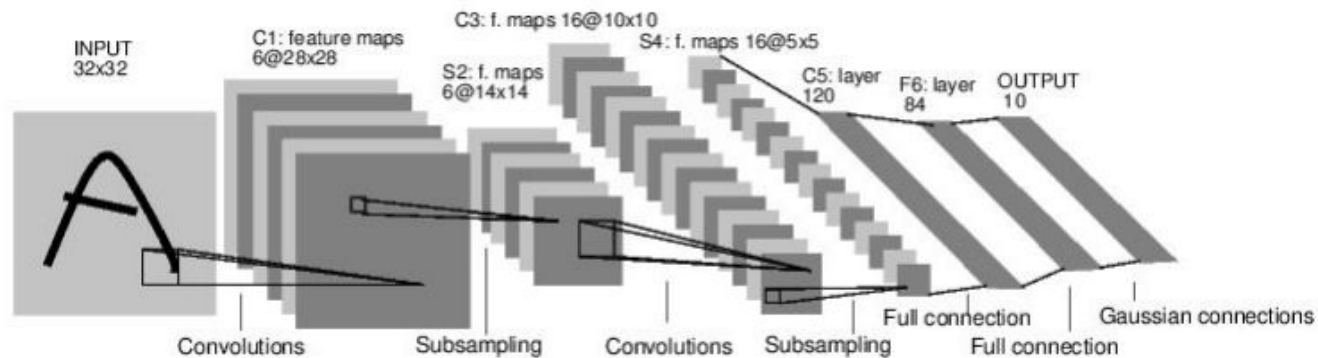
Stack activations to get a 6x28x28 output image!

Pooling (Subsampling)



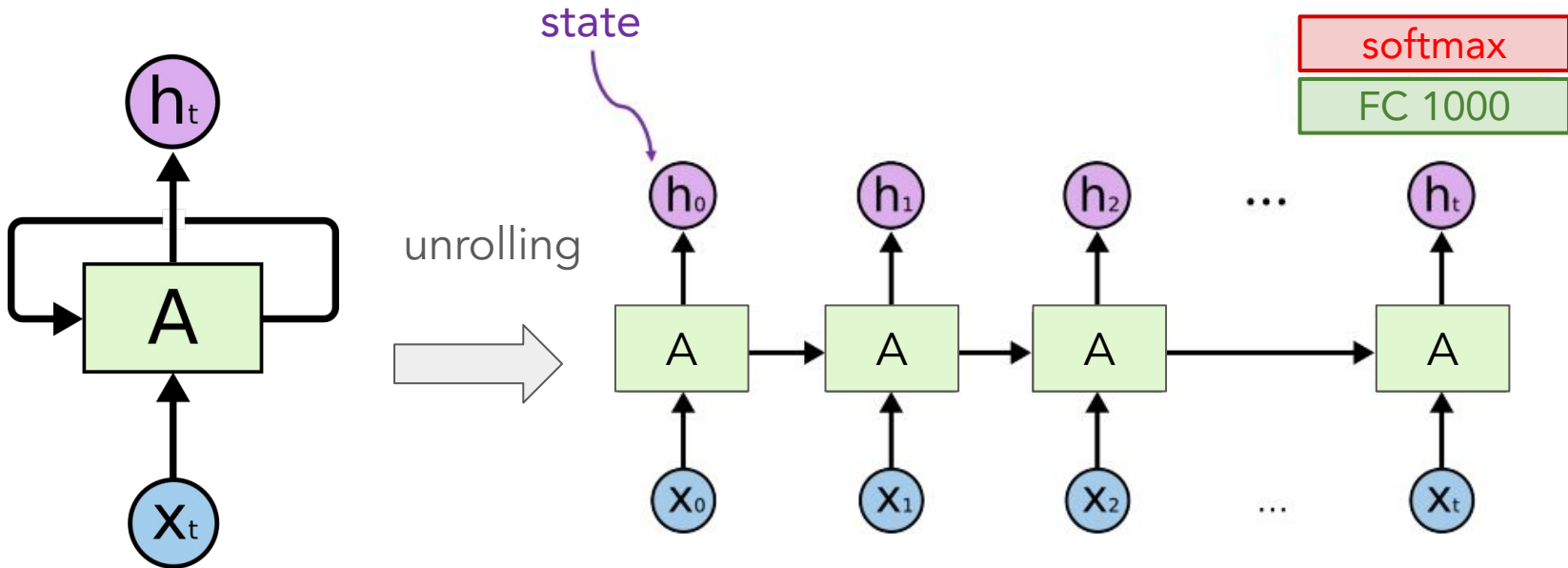
- Pooling layers simplify / subsample / compress the information in the output from the convolutional layer
- Reduce parameters

Put Everything Together



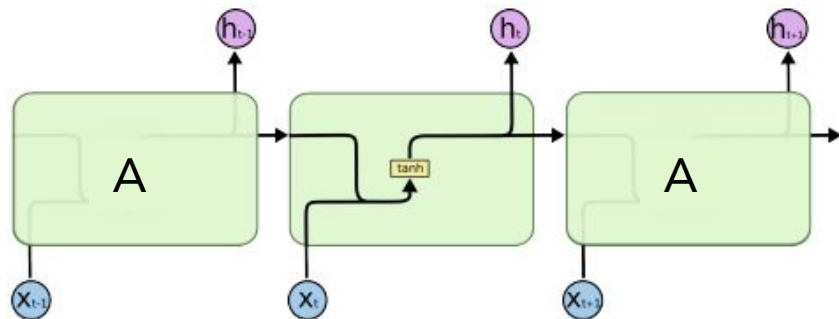
[LeNet-5, LeCun 1980]

Recurrent Neural Network

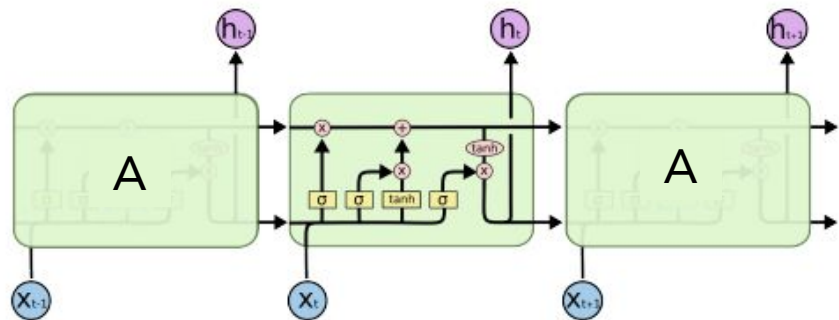


For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt.

Long Short Term Memory (LSTM)



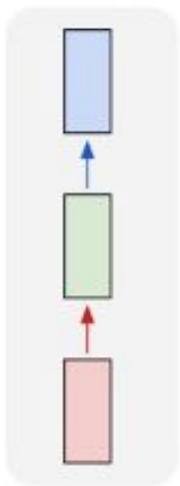
Simple RNN



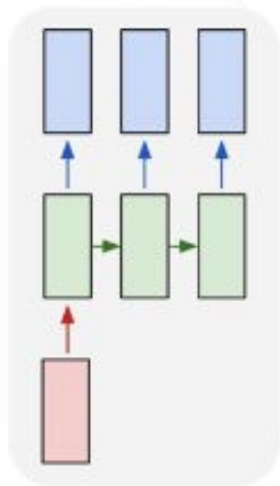
LSTM

Training of RNNs

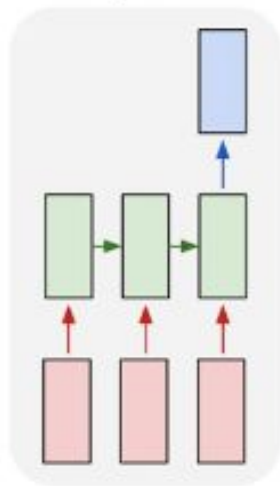
one to one



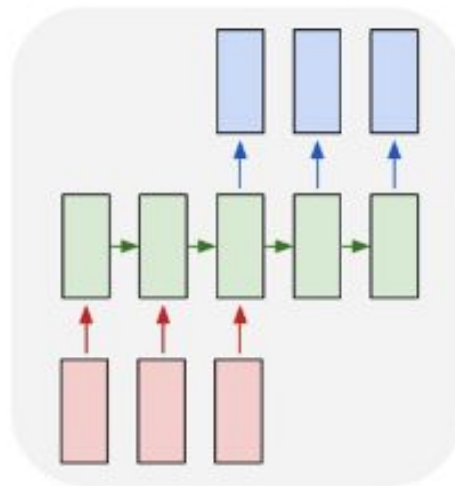
one to many



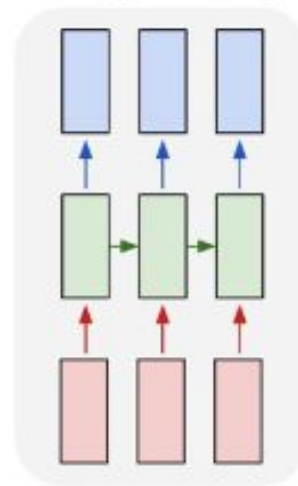
many to one



many to many

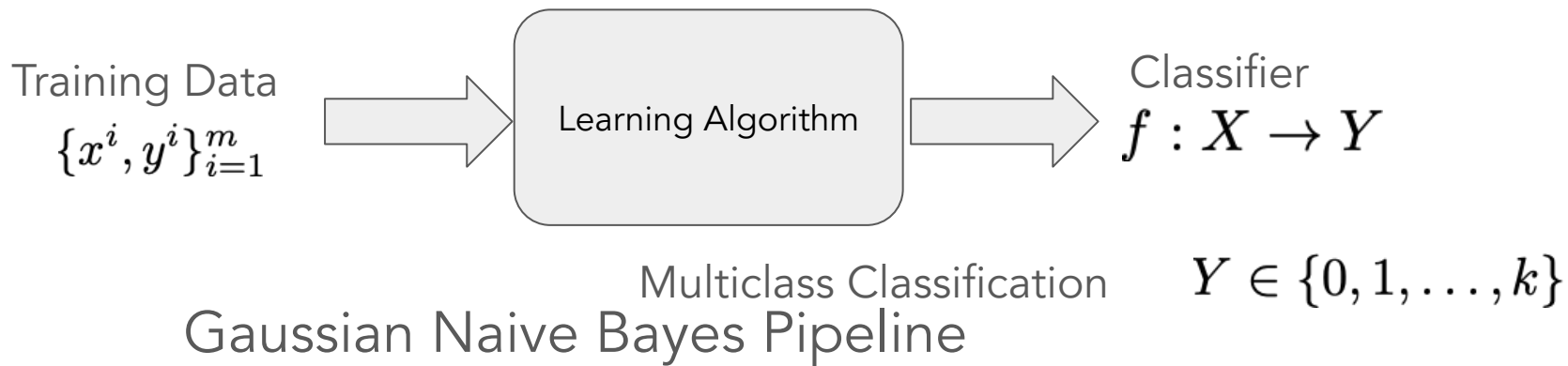


many to many



Backpropagation Through Unrolling Steps

Naive Bayes



1. Build probabilistic models: [Gaussian Likelihood](#)
2. Derive loss function: [MLE or MAP](#)
3. Select optimizer: [Necessary Condition](#)

Bayes' Rule

Softmax in Multiclass
Classification

$$p(y_i = 1 | \theta_i^\top x) = \frac{\exp(\theta_i^\top x)}{\sum_{i=1}^k \exp(\theta_i^\top x)}$$

A diagram illustrating Bayes' Rule with red arrows pointing from labels to parts of the equation. The label 'likelihood' points to $P(x|y)$, 'Prior' points to $P(y)$, 'posterior' points to $P(y|x)$, and 'normalization constant' points to $\sum_z P(x, y)$.

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x, y)}{\sum_z P(x, y)}$$

Bayes' Rule

The diagram shows the equation $P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x,y)}{\sum_z P(x,y)}$. Red arrows point from labels to parts of the equation: 'likelihood' points to $P(x|y)$, 'Prior' points to $P(y)$, 'posterior' points to $P(y|x)$, and 'normalization constant' points to $P(x)$.

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x,y)}{\sum_z P(x,y)}$$

likelihood Prior

posterior normalization constant

Prior: $P(y)$ $\pi = (\pi_1, \pi_2, \dots, \pi_k), \quad \sum_{i=1}^k \pi_i = 1, \pi_i \geq 0$

Likelihood (class conditional distribution) : $p(x|y) = \mathcal{N}(x|\mu_y, \Sigma_y)$

Posterior: $P(y|x) = \frac{P(y)\mathcal{N}(x|\mu_y, \Sigma_y)}{\sum_y P(y)\mathcal{N}(x|\mu_y, \Sigma_y)}$

Decision with Bayes' Rule

- The posterior probability of a test point

$$q_i(\mathbf{x}) := P(y = i | \mathbf{x}) = \frac{P(\mathbf{x} | y) P(y)}{P(\mathbf{x})}$$

- Bayes decision rule:

- If $q_i(\mathbf{x}) > q_j(\mathbf{x})$, then $y = i$, otherwise $y = j$

- Alternatively:

- If ratio $l(\mathbf{x}) = \frac{P(\mathbf{x} | y = i)}{P(\mathbf{x} | y = j)} > \frac{P(y = j)}{P(y = i)}$, then $y = i$, otherwise $y = j$

- Or look at the log-likelihood ratio $h(\mathbf{x}) = \ln \frac{q_i(\mathbf{x})}{q_j(\mathbf{x})}$

MLE of Naive Bayes

$$\theta = [\mu, \Sigma, \pi], \quad Z = \sqrt{(2\pi)^D \det(\Sigma)}$$

$$p(x^i | y_j^i = 1, \theta) = \frac{1}{Z} \exp\left(-\frac{1}{2}(x^i - \mu_j)^\top \Sigma_j^{-1}(x^i - \mu_j)\right)$$

$$\log L(\theta) = \log p(x, y | \theta) = \log p(y | \theta) + \log p(x | y, \theta)$$

$$\log L(\theta) = \sum_{i=1}^N \sum_{j=1}^k y_j^i \log \pi_j - \sum_{i=1}^N \log Z - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^k y_j^i (x^i - \mu_j)^\top \Sigma_j^{-1} (x^i - \mu_j)$$

Want $\arg \max_{\theta} \log L(\theta)$ subject to $\sum_{j=1}^k \pi_j = 1$

Gradient Calculation of MLE

$$Z = \sqrt{(2\pi)^D \det(\Sigma)}$$

Take derivative w.r.t μ_k $\sum_{i=1}^N \sum_{j=1}^k y_j^i \log \pi_j - \log Z - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^k y_j^i (x^i - \mu_j)^\top \Sigma_j^{-1} (x^i - \mu_j)$

$$\frac{\partial \log L}{\partial \mu_k} = \sum_{i=1}^N y_k^i \Sigma_k^{-1} (x^i - \mu_k) = 0$$

$$\mu_k = \frac{\sum_{i=1}^N y_k^i x^i}{\sum_{i=1}^N y_k^i}$$

Necessary Condition for MLE

Take derivative w.r.t Σ_k^{-1} (not Σ_k)

Note:

$$\frac{\partial \log L}{\partial \Sigma_k^{-1}} = - \sum_{i=1}^N y_k^i \left[-\frac{\partial \log Z_k}{\partial \Sigma_k^{-1}} - \frac{1}{2} (x^i - \mu_k)(x^i - \mu_k)^\top \right] = 0$$

$$\Sigma_k = \frac{\sum_{i=1}^N y_k^i (x^i - \mu_k)(x^i - \mu_k)^\top}{\sum_{i=1}^N y_k^i}$$

Necessary Condition for MLE

Use Lagrange multiplier to derive π_k $\sum_{i=1}^N \sum_{j=1}^k y_j^i \log \pi_j - \log Z - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^k y_j^i (x^i - \mu_j)^\top \Sigma_j^{-1} (x^i - \mu_j)$

$$\frac{\partial L(\theta)}{\partial \pi_k} + \lambda \frac{\partial \sum_k \pi_k}{\partial \pi_k} = 0 \Rightarrow \lambda = - \sum_{i=1}^N y_k^i \frac{1}{\pi_k}$$

$$\pi_k = - \frac{\sum_{i=1}^N y_k^i}{\lambda}$$

Apply constraint: $\sum_k \pi_k = 1 \Rightarrow \lambda = -N$

$$\pi_k = \frac{\sum_{i=1}^N y_k^i}{N}$$

Discriminative vs. Generative Classifier

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x, y)}{\sum_y P(x, y)}$$

Discriminative

Generative

- Directly estimate decision boundary $h(x) = -\ln \frac{q_i(x)}{q_j(x)}$ or posterior distribution $p(y|x)$
- $h(x)$ or $f(x) := p(y=1|x)$ is a function of x , and
 - Does **not** have probabilistic meaning
 - Hence can **not** be used to sample data points
- Estimate the probabilistic generative mechanism $P(x|y)P(y)$
- Derive decision boundary through Bayes' rule

Discriminative vs. Generative Classifier

Binary
Logistic Regression

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

Gaussian
Naive Bayes Classifier

$$\begin{aligned} P(y|x) &= \frac{P(x|y)P(y)}{P(x)} = \frac{P(x, y)}{\sum_y P(x, y)} \\ &= \frac{P(y)\mathcal{N}(x|\mu_y, \Sigma_y)}{\sum_y P(y)\mathcal{N}(x|\mu_y, \Sigma_y)} \end{aligned}$$

Density Estimation: Gaussian Mixture Model



Density Estimation Pipeline

1. Build probabilistic models
Gaussian Mixture Model
2. Derive loss function (by MLE or MAP....)
ELBO Approximation of MLE
3. Select optimizer EM

Gaussian Mixture Model

Class mixture prior: $P(y)$ $\pi = (\pi_1, \pi_2, \dots, \pi_k), \sum_{i=1}^k \pi_i = 1, \pi_i \geq 0$

Class conditional distribution: $p(x|y) = \mathcal{N}(x|\mu_y, \Sigma_y)$

Marginal distribution: $P(x) = \sum_y P(x|y)P(y) = \sum_{i=1}^k \pi_i \mathcal{N}(x|\mu_i, \Sigma_i)$

Connection to Gaussian Naive Bayes

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x, y)}{\sum_z P(x, y)}$$

Prior: $P(y)$ $\pi = (\pi_1, \pi_2, \dots, \pi_k)$, $\sum_{i=1}^k \pi_i = 1, \pi_i \geq 0$

Likelihood (class conditional distribution): $p(x|y) = \mathcal{N}(x|\mu_y, \Sigma_y)$

$$\text{Posterior: } P(y|x) = \frac{P(y)\mathcal{N}(x|\mu_y, \Sigma_y)}{\sum_y P(y)\mathcal{N}(x|\mu_y, \Sigma_y)}$$

What is the difference?

Expectation-Maximization

For $t = 1, \dots$

- **E-Step**: Guess sample labels based on current model

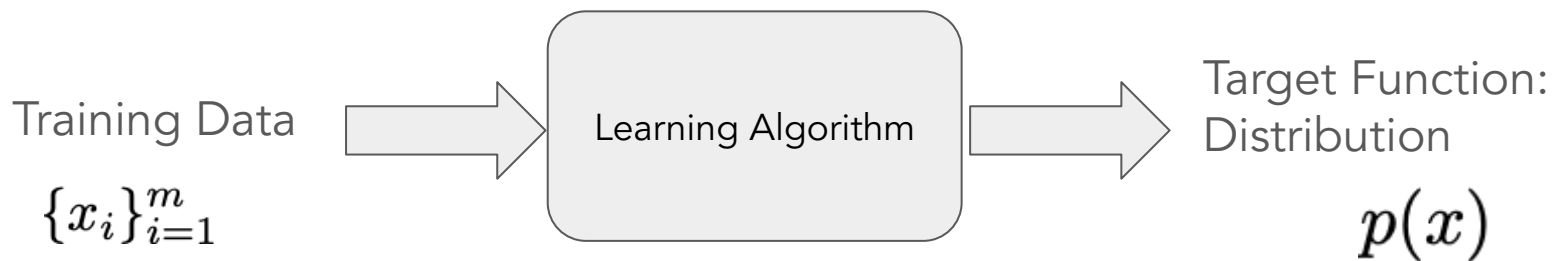
$$y_j^l = \frac{\pi_l \mathcal{N}(x_j | \mu_l, \Sigma_l)}{\sum_{l=1}^k \pi_l \mathcal{N}(x_j | \mu_l, \Sigma_l)}$$

- **M-Step**: Update the parameters with current labels

$$\mu_k = \frac{\sum_{i=1}^m y_k^i x^i}{\sum_{i=1}^m y_k^i} \quad \pi_k = \frac{\sum_{i=1}^m y_k^i}{m} \quad \Sigma_k = \frac{\sum_{i=1}^m y_k^i (x^i - \mu_k) (x^i - \mu_k)^\top}{\sum_{i=1}^m y_k^i}$$

This procedure is actually maximizing the Evidence Lower Bound of MLE

K-means is **Approximating** Gaussian Mixture Model



Density Estimation Pipeline

1. Build probabilistic models
Gaussian Mixture Model with fixed covariance
2. Derive loss function (by MLE or MAP....)
Approximated MLE
3. Select optimizer
Coordinate Descent

K-means from MLE Perspective

- K-means Objective:

Find cluster centers $\boldsymbol{\mu}$ and assignments \mathbf{y} to minimize the sum of squared distance of the data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centers

$$\begin{aligned} \min_{\{\boldsymbol{\mu}\}, \{\mathbf{y}\}} J(\{\boldsymbol{\mu}\}, \{\mathbf{y}\}) &= \min_{\{\boldsymbol{\mu}\}, \{\mathbf{y}\}} \sum_{n=1}^N \sum_{k=1}^K y_k^{(n)} \|\boldsymbol{\mu}_k - \mathbf{x}^{(n)}\|^2 \\ \text{s.t. } \sum_k y_k^{(n)} &= 1, \forall n, \text{ where } y_k^{(n)} \in \{0, 1\}, \forall k, n \end{aligned}$$

where $y_k^{(n)} = 1$ means that $\mathbf{x}^{(n)}$ is assigned to cluster k (with center $\boldsymbol{\mu}_k$).

K-means vs. GMM

- Initialize k cluster centers, $\{c^1, c^2, \dots, c^k\}$, randomly
- Do
 - (Assignment) Decide the cluster memberships of each data point, x^i , by assigning it to the nearest cluster center

$$y^i = \arg \min_{j=1, \dots, k} \|x^i - \mu_j\|^2.$$

- (Center Update) Adjust the cluster centers

$$\mu_j = \frac{1}{|\{i : y^i = j\}|} \sum_{i: y^i = j} x^i.$$

- While any cluster center has been changed

For $t = 1, \dots$

- E-Step: Guess sample labels based on current model

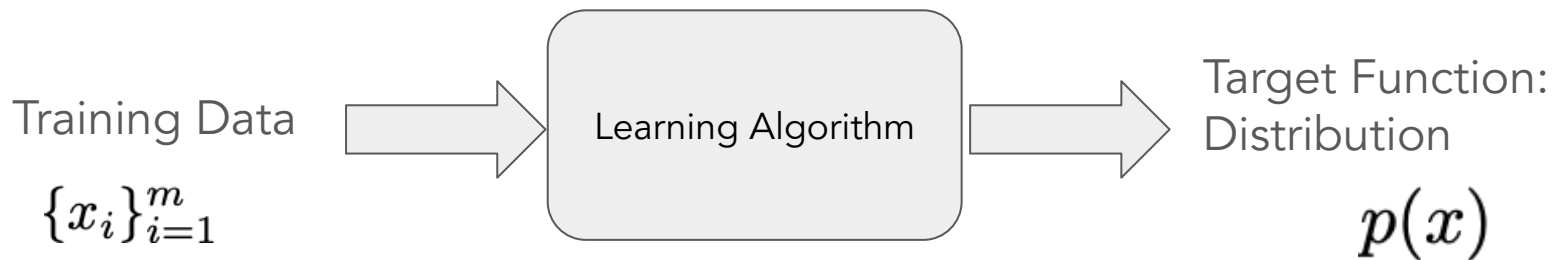
$$y_j^t = \frac{\pi_t \mathcal{N}(x_j | \mu_t, \Sigma_t)}{\sum_{l=1}^k \pi_t \mathcal{N}(x_j | \mu_l, \Sigma_l)}$$

- M-Step: Update the parameters with current labels (Gaussian-Naive Bayes)

$$\mu_k = \frac{\sum_{i=1}^m y_k^i x^i}{\sum_{i=1}^m y_k^i} \quad \pi_k = \frac{\sum_{i=1}^m y_k^i}{m} \quad \Sigma_k = \frac{\sum_{i=1}^m y_k^i (x^i - \mu_k)(x^i - \mu_k)^\top}{\sum_{i=1}^m y_k^i}$$

K-means can be understood as hard-GMM
GMM can be understood as soft k-means

Generative Model: Latent Variable Models



Density Estimation Pipeline

1. Build probabilistic models
Deep Latent Variable Model
2. Derive loss function (by MLE or MAP....)
ELBO
3. Select optimizer
Stochastic Gradient with Reparametrization Trick

Reparameterization Trick

$$\max_{q(z|x)} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^m \mathbb{E}_{q(z^i|x^i)} \left[\log p(x^i|z^i)p(z^i) - \log q(z^i|x^i) \right]$$

$$z \sim q(z|x) = \mathcal{N}(z | \mu_z(x), \text{diag}(\sigma_z^2(x)))$$

$$z = \mu_z(x) + \sigma_z(x)\epsilon \quad \epsilon \sim \mathcal{N}(0, I)$$

$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^m \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\log p(x^i | \mu(x^i) + \sigma(x^i)\epsilon) p(\mu(x^i) + \sigma(x^i)\epsilon) \right. \\ \left. - \log q(\mu(x^i) + \sigma(x^i)\epsilon | x^i) \right]$$

Exam Schedule

- The [practice exam](#) has been released on **March 6th** on the course website.
- One exam will be held on **March 18th** in lieu of the regular class.
- One exam for students, who have registered with the ODS and been approved, will be held on **March 18th** in **CODA C1303**
- We will have a makeup exam on **March 17th** in **CODA C1315**
 - You need an **approval of absence** to register this makeup exam
 - Please register the makeup exam by **March 16th** (**sending email to instructors and TAs**).
- For students attending exam in CODA, you may need the building access. If so, please gather at **2nd floor in CODA on 3:15pm** and TA will lead you to the room.

Q&A

Good Luck!

Makeup Exam registration is due **Today**

HW3 is due on **Mar 30th**