Georgia Tech
College of Computing

CS4641 Spring 2025
Dimension Reduction

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

Supervised Learning vs. Unsupervised Learning

Training Data

Supervised $\{x^i, y^i\}_{i=1}^m$

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

Learning Algorithm

Target Function

$f : X \to Y$

Density Estimation $p(x)$

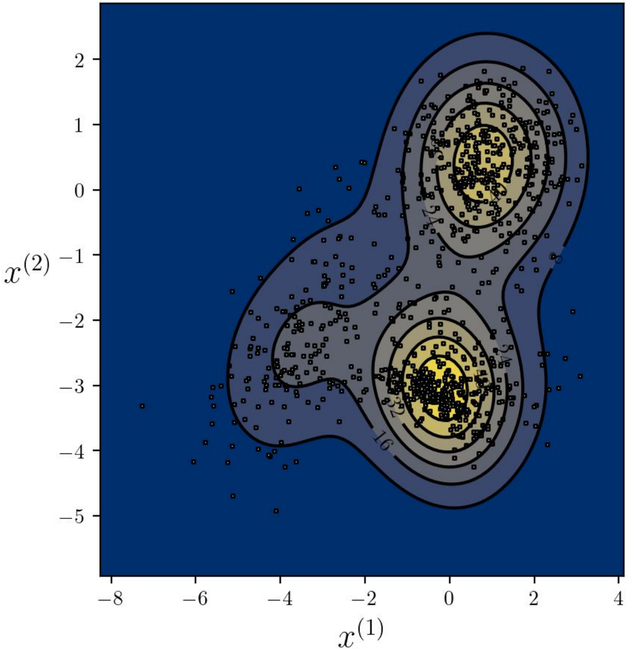
Clustering $f : X \to Y$

$Y \in \{0, 1, \ldots, k\}$

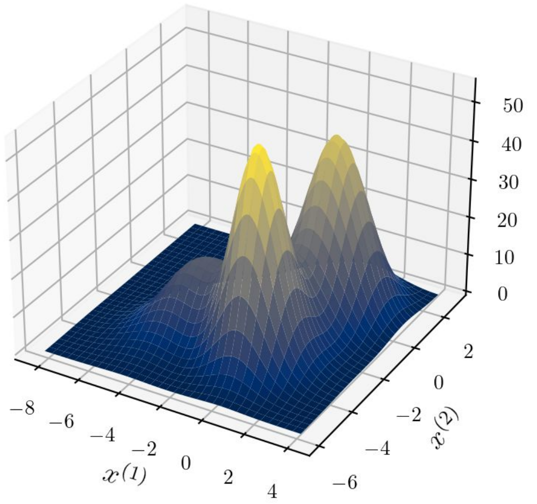
Dimension Reduction $f : X \to Y$

$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$

Density Estimation



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



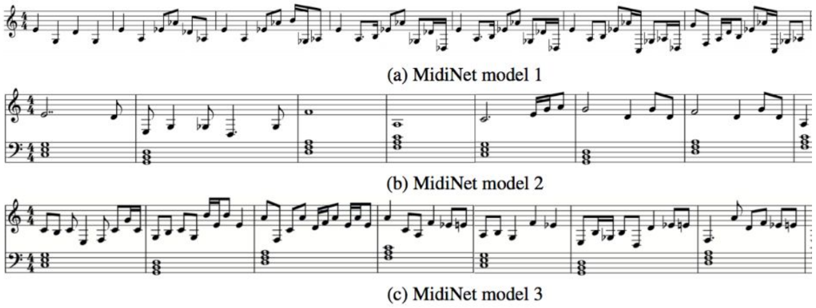
$p(x)$

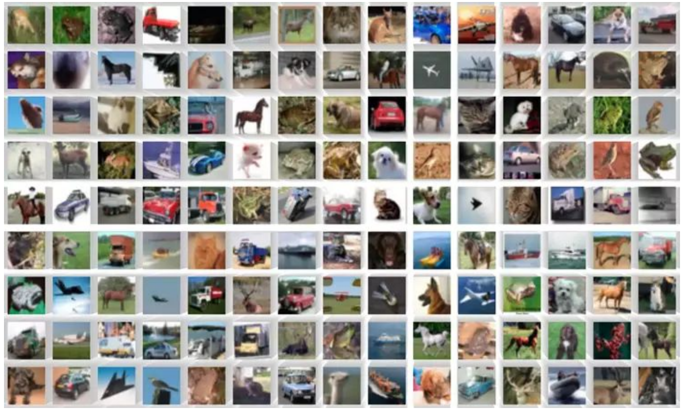
Generative Models

$x \sim p(x)$

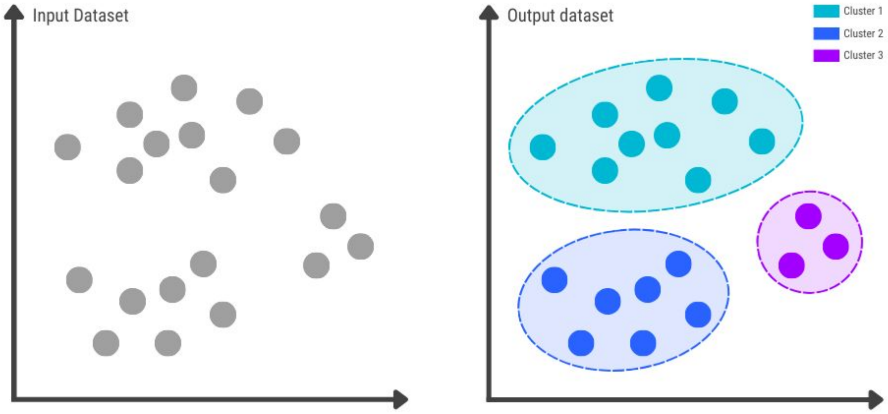
Density Estimation: Generative Models

$$x \sim p(x)$$

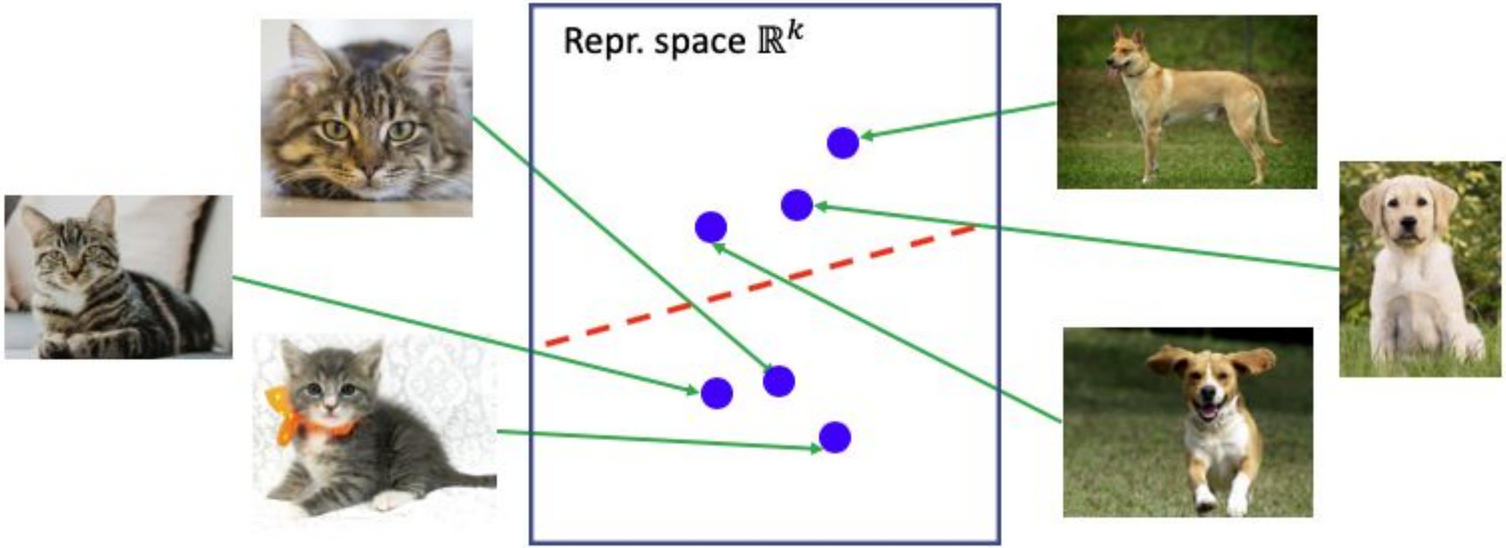




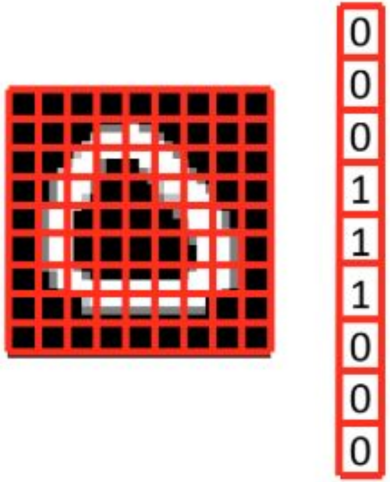
Clustering



Dimension Reduction/Representation Learning



Dimension Reduction/Representation Learning



vector in $\mathbb{R}^p$

Original data point

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

$$f(\mathbf{x}) : \mathbb{R}^p \to \mathbb{R}^d$$

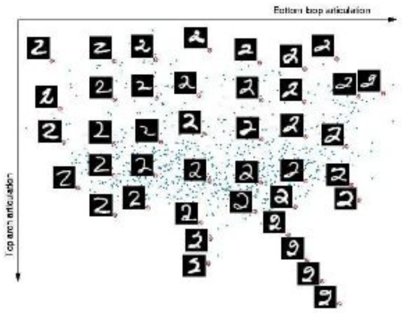
Reduced representation

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{pmatrix}$$

$$d \ll p$$

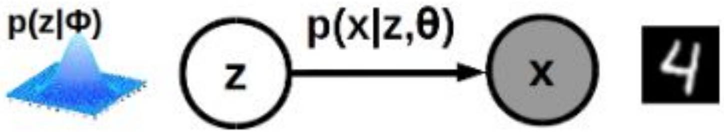
Why Dimension Reduction

- 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 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.

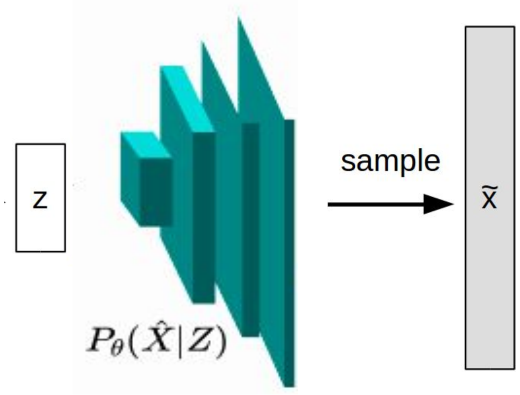
Deep Gaussian Distribution

Gaussian Distribution

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_z)}} \exp\Big(-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_z)^T \boldsymbol{\Sigma}_z^{-1} (\mathbf{x} - \boldsymbol{\mu}_z)\Big)$$

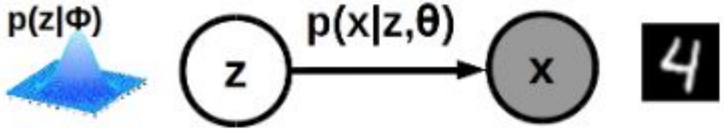
Deep Gaussian Distribution

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}(z), \boldsymbol{\Sigma}(z)) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}(z))}} \exp\Big(-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}(z))^T \boldsymbol{\Sigma}(z)^{-1} (\mathbf{x} - \boldsymbol{\mu}(z))\Big)$$



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

Deep Latent Variable Models: Deep Gaussian LVM



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

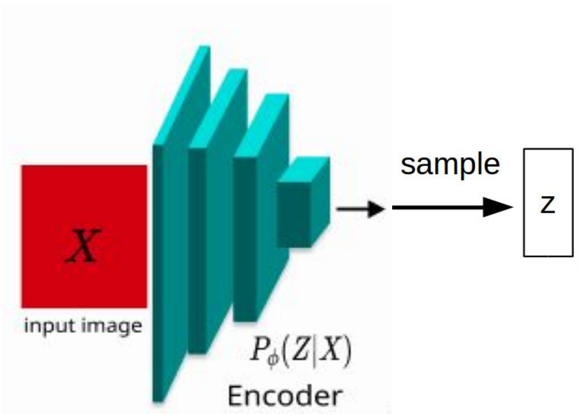
$$p(z) = \mathcal{N}(0, \sigma I)$$

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

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

Evidence Lower Bound

$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^{m} \int q(z^i|x^i) \Big(\log p(x^i|z^i) p(z^i) \Big) dz^i - \underbrace{\int q(z^i|x^i) \log q(z^i|x^i) dz_i}_{H(q(z|x))}$$



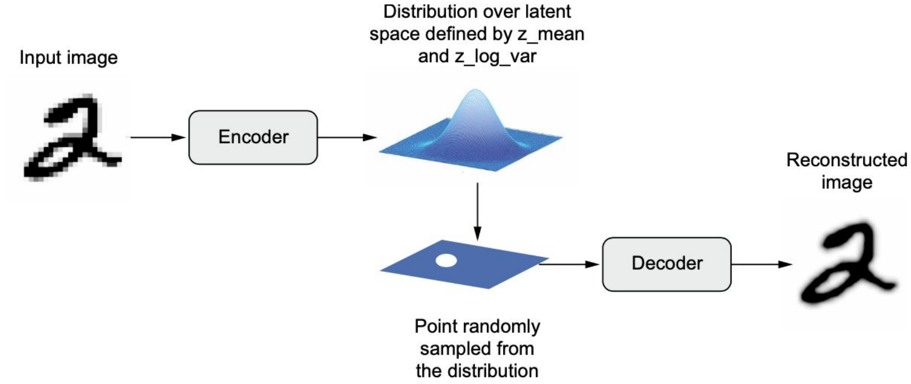
$$q(z|x) = \mathcal{N}\left(z \mid \mu_z(x), \operatorname{diag}(\sigma_z^2(x))\right)$$

deep neural network

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

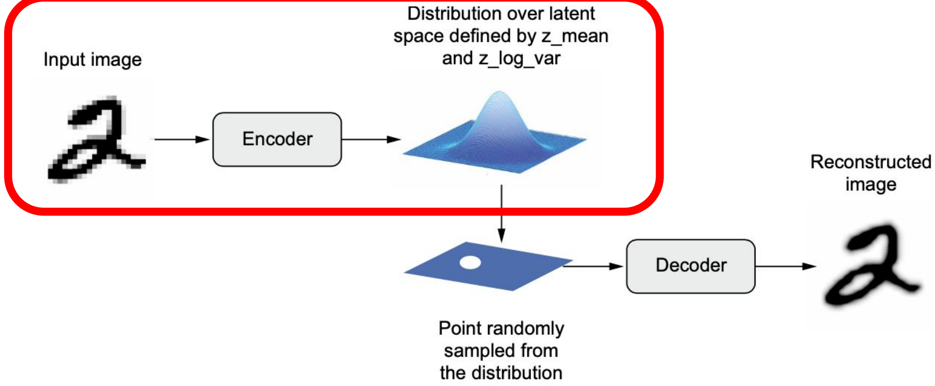
Evidence Lower Bound

$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^{m} \int q(z^i|x^i) \Big(\log p(x^i|z^i) p(z^i) \Big) dz^i \underbrace{- \int q(z^i|x^i) \log q(z^i|x^i) dz_i}_{H(q(z|x))}$$



Evidence Lower Bound

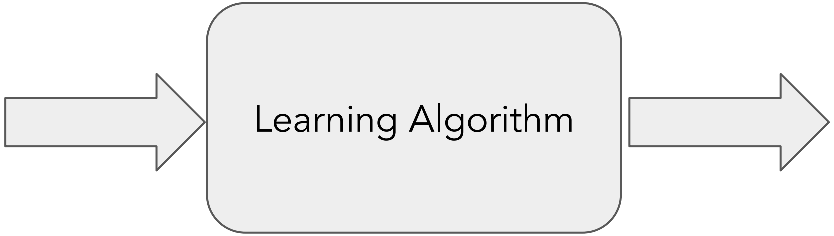
$$\max_{W_z^1, W_z^2} \max_{\sigma, W_\sigma, W_\mu} \sum_{i=1}^{m} \int q(z^i|x^i)\Big(\log p(x^i|z^i)p(z^i)\Big)dz^i - \underbrace{\int q(z^i|x^i)\log q(z^i|x^i)dz_i}_{H(q(z|x))}$$



Probabilistic Principal Component Analysis as LVM

Training Data

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



Target Function:

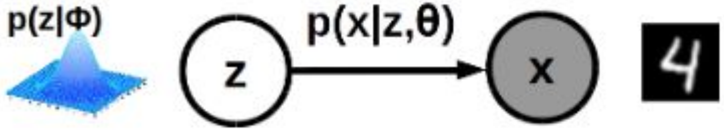
$$f : X \rightarrow Y$$

$$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$$

Density Estimation Pipeline

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

Gaussian LVM

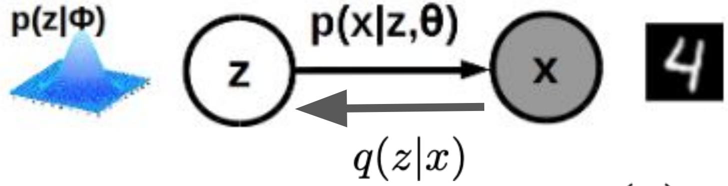


$$p(x) = \int p(x|z)p(z)dz$$

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

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

Gaussian LVM for Dimension Reduction



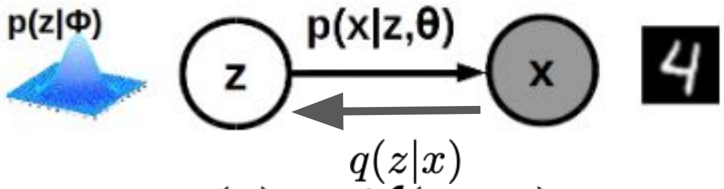
$$p(x) = \int p(x|z)p(z)dz$$

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

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

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

Gaussian LVM for Dimension Reduction



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

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

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

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

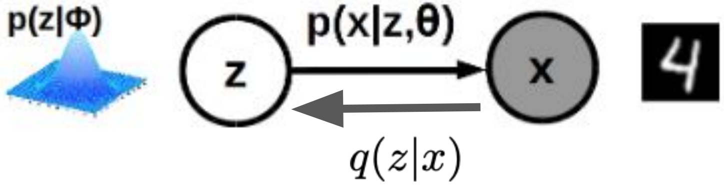
$$E[x] = E[Wz + \mu + \epsilon] = \mu$$

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

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

[*Chap 2.3 in Pattern Recognition and Machine Learning*]

Gaussian LVM for Dimension Reduction



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

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

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

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

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

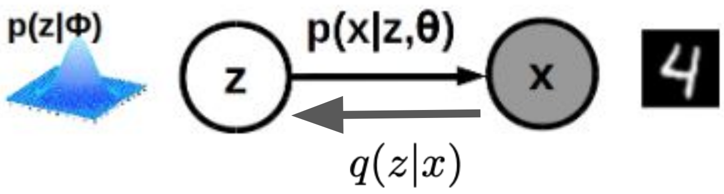
$$E[x] = E[Wz + \mu + \epsilon] = \mu$$

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

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

[Chap 2.3 in Pattern Recognition and Machine Learning](#)

Gaussian LVM for Dimension Reduction



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

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

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

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

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

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

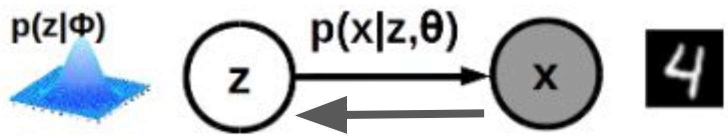
$$E[z|x] = (W^T W + \sigma^2 I)^{-1} W^T (x - \mu)$$

- Posterior variance:

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

Chap 2.3 in Pattern Recognition and Machine Learning

Gaussian LVM for Dimension Reduction



$q(z|x)$

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

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

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

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

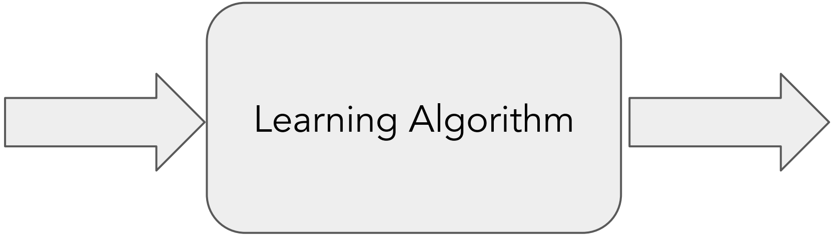
$$q(z|x) = \frac{p(x|z)p(z)}{p(x)} = \mathcal{N}(MW^\top (x - \mu), \sigma^2 M)$$

Chap 2.3 in Pattern Recognition and Machine Learning

Probabilistic Principal Component Analysis as LVM

Training Data

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



Target Function:

$$f : X \to Y$$

$$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$$

Density Estimation Pipeline

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

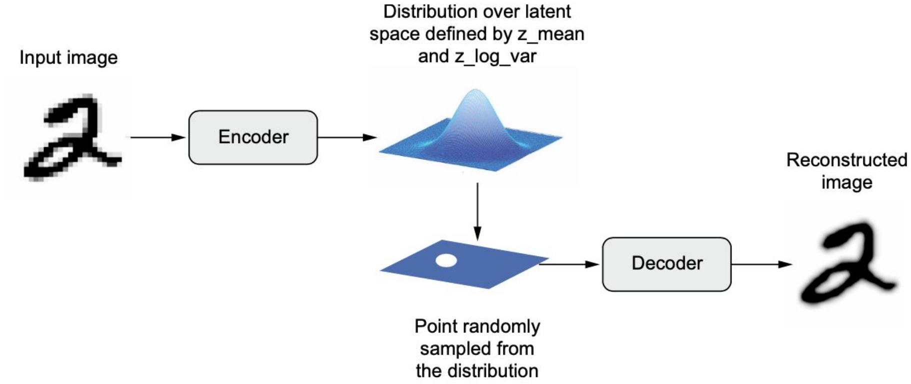
Revisit MLE of Deep LVM

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

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

Revisit Evidence Lower Bound

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



MLE of Gaussian LVM

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

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

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

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

The EM Algorithm for Gaussian LVM

- Initialize W and σ^2
  - E step: Compute the exp. complete data log-lik. using current W and σ^2

$$\sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \|x_n\|^2 - \frac{1}{\sigma^2} E[z_n]^T W x_n + \frac{1}{2\sigma^2} \mathrm{tr}(E[z_n z_n^T] W^T W) + \frac{1}{2} \mathrm{tr}(E[z_n z_n^T]) \right\}$$

where

$$E[z_n] = (W^T W + \sigma^2 I_k)^{-1} W^T x_n = M^{-1} W^T x_n$$

$$E[z_n z_n^T] = \mathrm{cov}(z_n) + E[z_n] E[z_n]^T = E[z_n] E[z_n]^T + \sigma^2 M^{-1}$$

  - M step: Re-estimate W and σ^2 (taking derivatives w.r.t W and σ^2, respectively)

$$W_{new} = \left[\sum_{n=1}^{N} x_n E[z_n]^T \right] \left[\sum_{n=1}^{N} E[z_n z_n^T] \right]^{-1} = \left[\sum_{n=1}^{N} x_n E[z_n]^T \right] \left[\sum_{n=1}^{N} E[z_n] E[z_n]^T + \sigma^2 M^{-1} \right]^{-1}$$

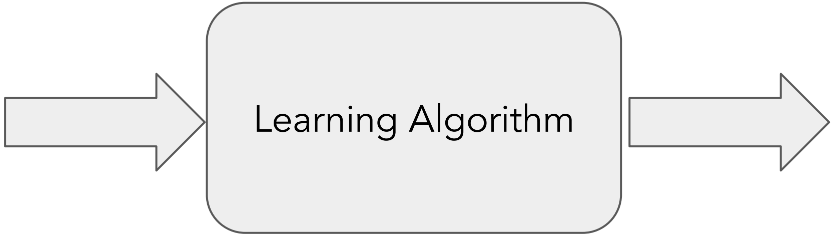
$$\sigma^2_{new} = \frac{1}{ND} \sum_{n=1}^{N} \{ \|x_n\|^2 - 2E[z_n]^T W_{new} x_n + \mathrm{tr}(E[z_n z_n^T] W_{new}^T W_{new}) \}$$

- Set $W = W_{new}$ and $\sigma^2 = \sigma^2_{new}$
- If not converged, go back to E step.

Probabilistic Principal Component Analysis as LVM

Training Data

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



Target Function:

$$f : X \to Y$$

$$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$$

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

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

- The optimal parameters for the maximal log-likelihood are

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$$

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

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

Denote

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

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

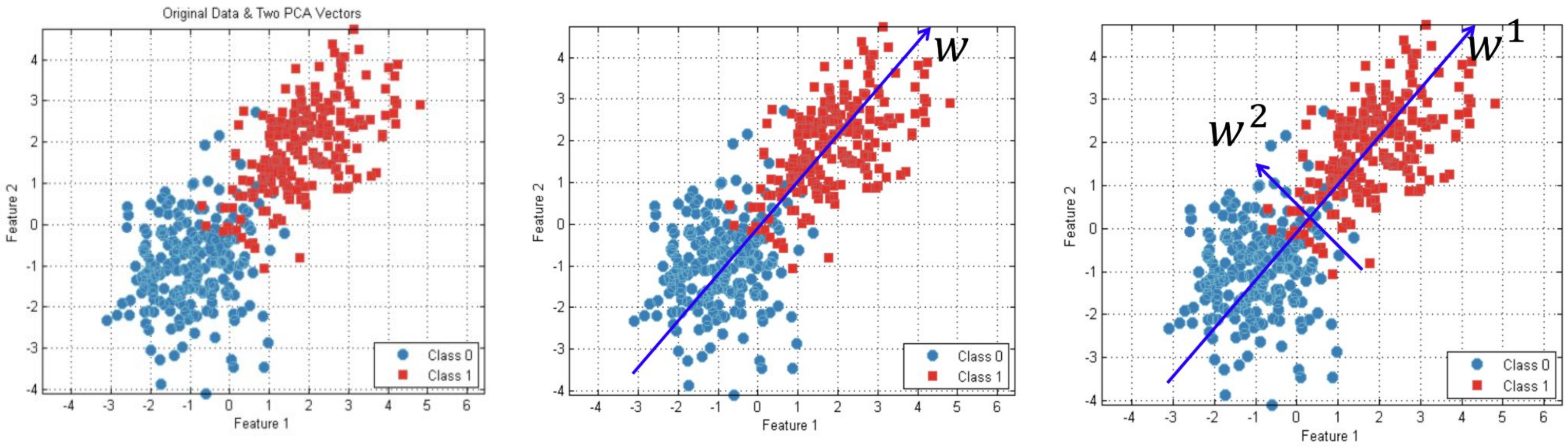
Covariance of Gaussian was

$$C = WW^T + \sigma^2 I$$

C should match S_{true}

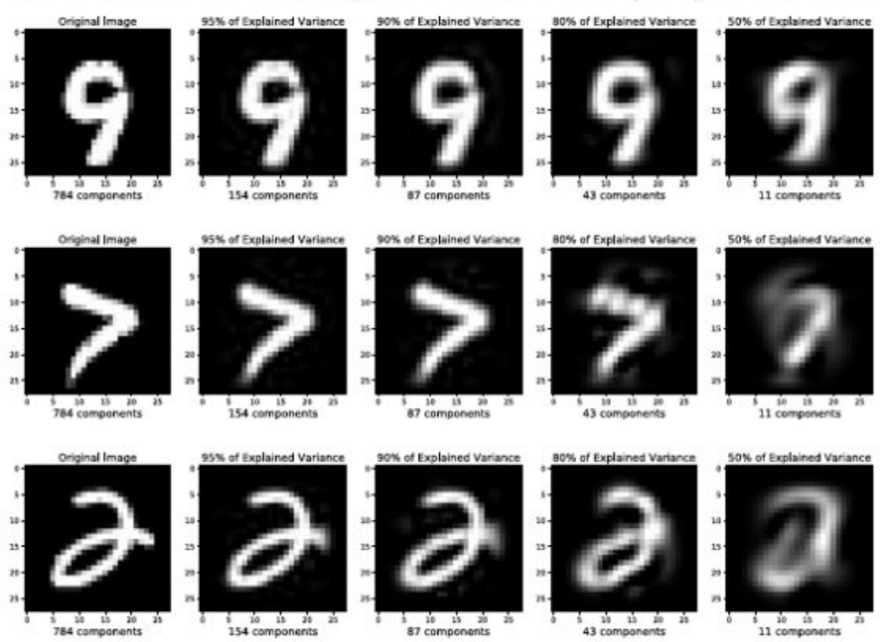
$$U_M \Lambda_M U_M^T = WW^T + \sigma^2 I$$

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)

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

CS4641 Spring 2025

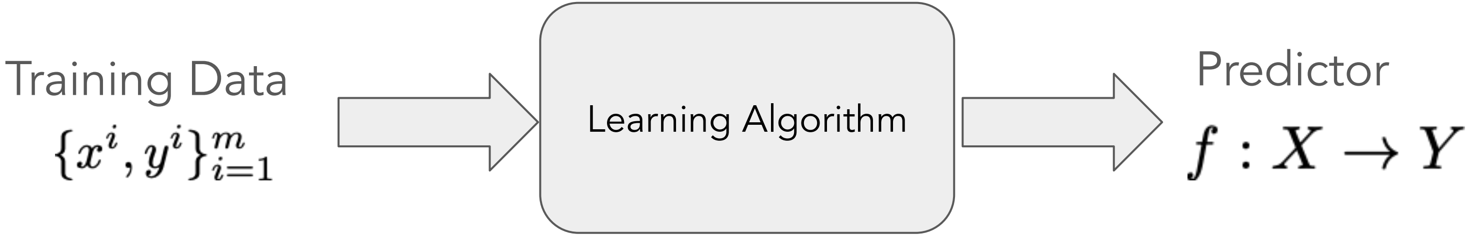
Review

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

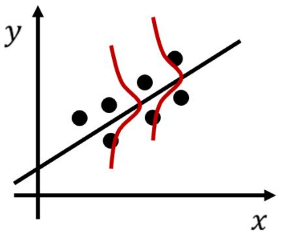
1. Build probabilistic models:
   Gaussian noise + linear model/polynomial model
2. Derive loss function:
   MLE vs. MAP
3. Select optimizer
   Necessary Condition vs. (Stochastic) GD

Probabilistic Model: Gaussian Likelihood

- Assume y is a linear in x plus noise ϵ

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

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



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

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

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

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

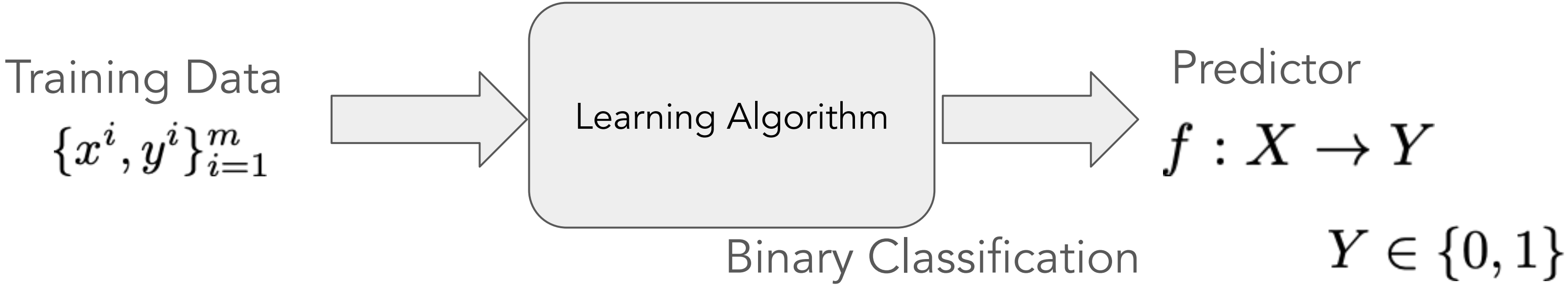
$$\propto -\frac{1}{m}\sum_{i=1}^m (y^i - \theta^\top x^i)^2 - \boxed{\lambda\|\theta\|_2^2}$$

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

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

MAP

Logistic regression model

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

Note that

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

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

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

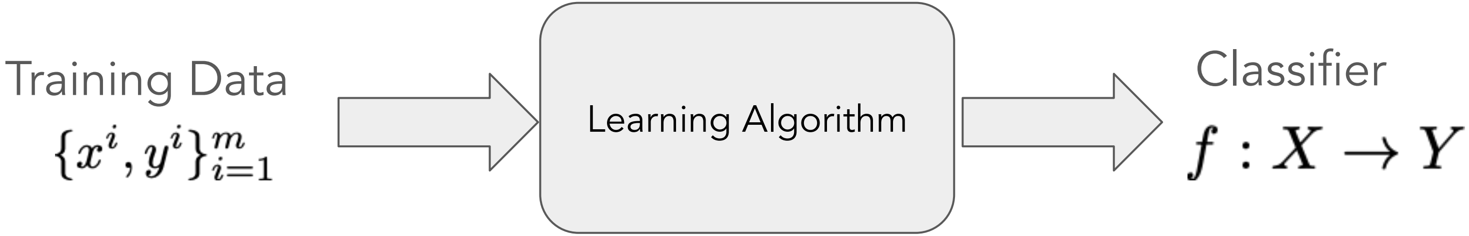
$$= \sum_i (y^i - 1)\,\theta^\top x^i - \log(1 + \exp(-\theta^\top x^i)) \; - \lambda\|\theta\|_2^2$$

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



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Multiclass Logistic Regression Pipeline

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

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

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

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

Gradient Calculation of MLE

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

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

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

MAP

- Likelihood

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

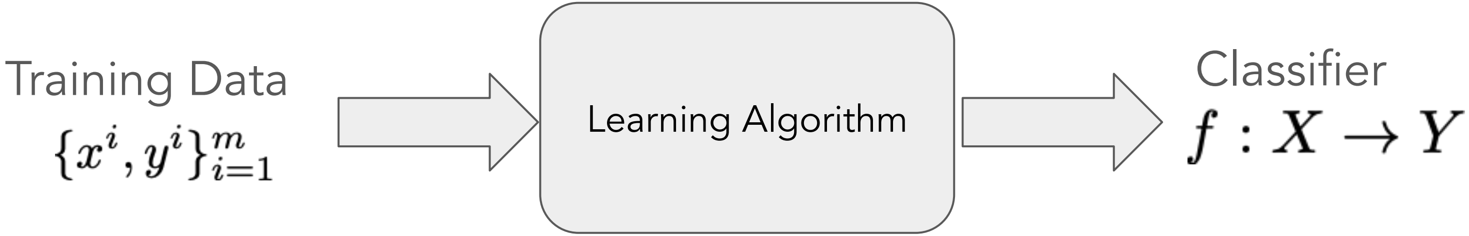
- Prior

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

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

$$= \sum_{i=1}^{m}\sum_{j=1}^{k} y_j^i \theta_j^\top x^i - \sum_{i=1}^{m} \log \sum_{c=1}^{k} \exp(\theta_c^\top x^i) - \lambda||\theta||_2^2$$

Naive Bayes



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Gaussian Naive Bayes Pipeline

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

Bayes' Rule

Softmax in Multiclass Classification

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

likelihood

Prior

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

posterior

normalization constant

Bayes' Rule

likelihood

Prior

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

posterior

normalization constant

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

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

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

Decision with Bayes' Rule

- The posterior probability of a test point

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

- Bayes decision rule:
  - If $q_i(x) > q_j(x)$, then $y = i$, otherwise $y = j$

- Alternatively:
  - If ratio $l(x) = \frac{P(x|y=i)}{P(x|y=j)} > \frac{P(y=j)}{P(y=i)}$, then $y = i$, otherwise $y = j$
  - Or look at the log-likelihood ratio $h(x) = \ln \frac{q_i(x)}{q_j(x)}$

MLE of Naive Bayes

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

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

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

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

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

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

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

- 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

Generative

- 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

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

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

Vector Formation

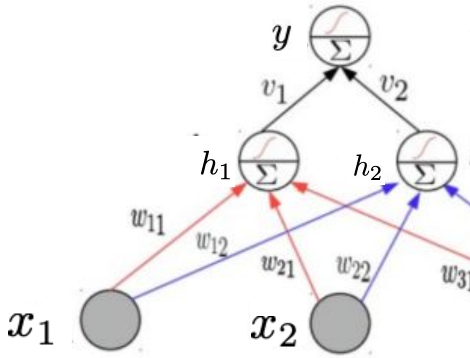
$$y = o(Vg(Wx))$$

$$V = [v_1, v_2]$$

$$h = [h_1, h_2]^\top = g(Wx)$$

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

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

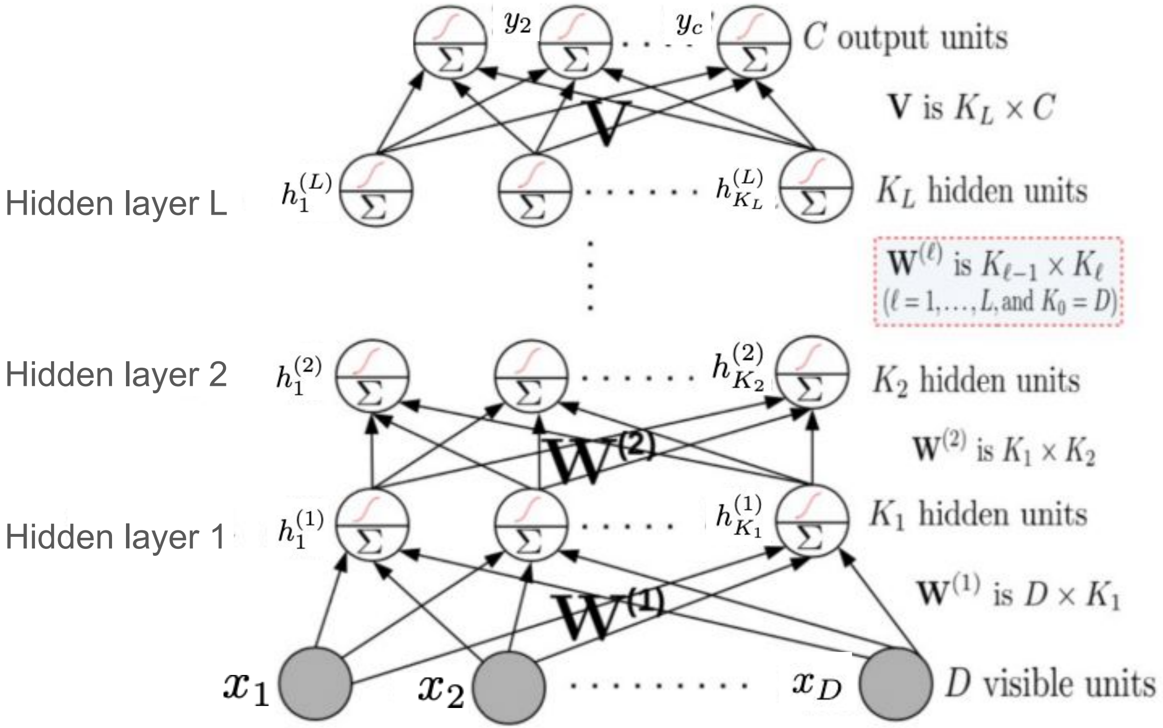


$$y = o\left(\sum_{k=1}^{K} v_k h_k\right)$$

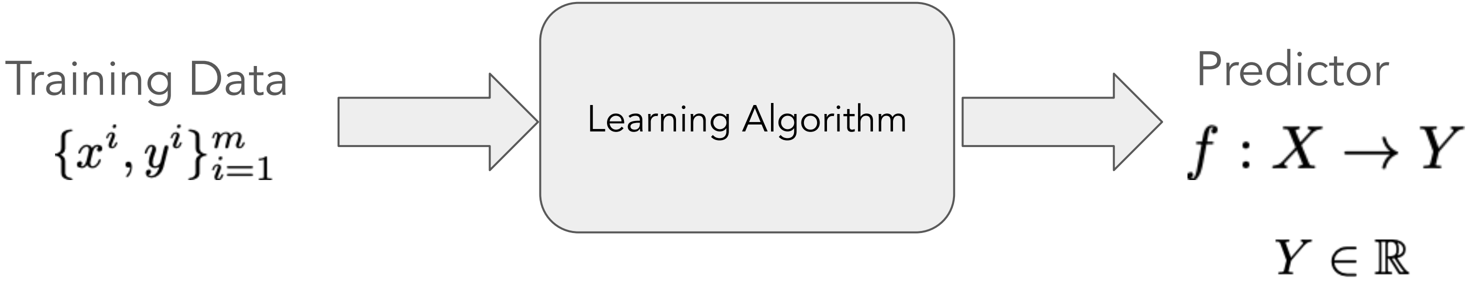
$$h_k = g\left(\sum_{i=1}^{n} w_{ki} x_i\right)$$

Multi-Layer Perception

$$y = f_L(W_L f_{L-1}(W_{L-1} \dots f_1(W_1 x))$$



Regression Algorithms



Linear Regression Pipeline

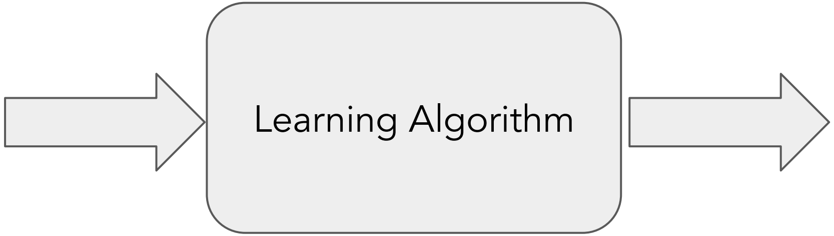
1. Build probabilistic models:
   Gaussian Distribution + Neural Network
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) GD

$$y = o(Vg(Wx))$$

Binary Classification Algorithms

Training Data

$\{x^i, y^i\}_{i=1}^m$



Predictor

$f : X \to Y$

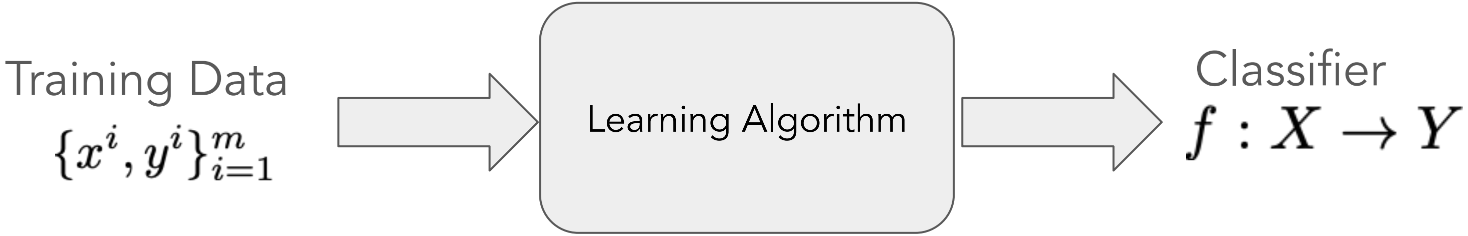
Binary Classification

$Y \in \{0, 1\}$

Binary Logistic Regression Pipeline

1. Build probabilistic models:
   Bernoulli Distribution + Neural Network $y = o(Vg(Wx))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Multiclass Logistic Regression Algorithms



Multiclass Classification $Y \in \{0, 1, \ldots, k\}$

Multiclass Logistic Regression Pipeline

1. Build probabilistic models:
   Categorical Distribution + Neural Network $y = o(Vg(Wx))$
2. Derive loss function: MLE and MAP
3. Select optimizer: (Stochastic) Gradient Descent

Select Optimizer

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

$$\theta = [V, W]$$

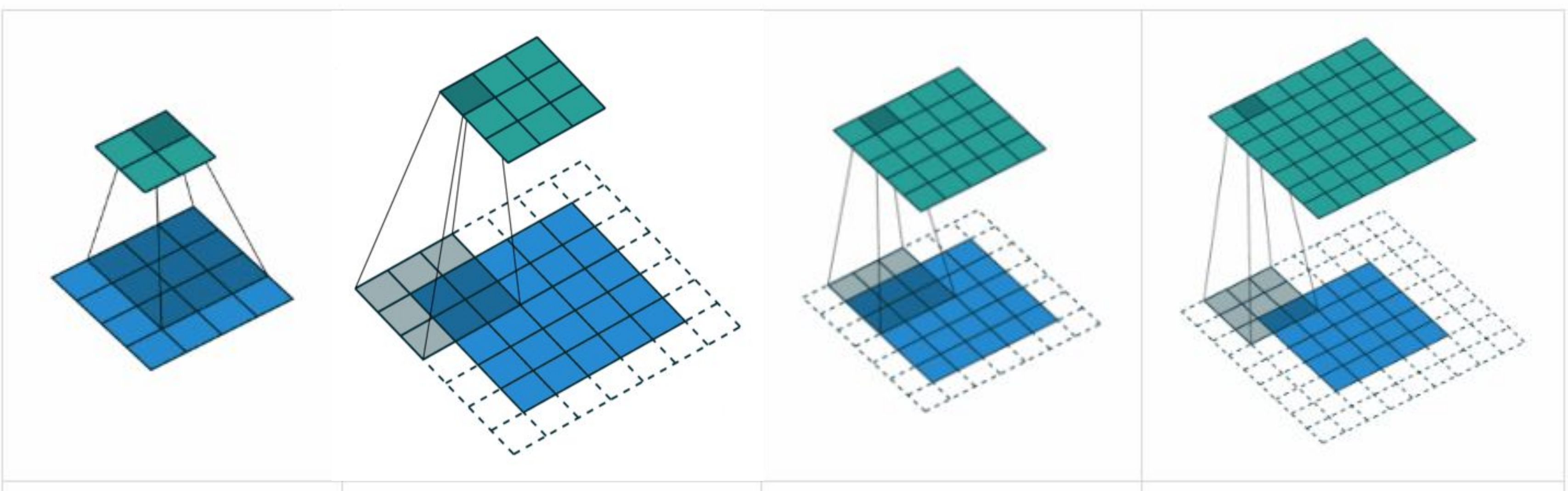
$$\ell(x^i, y^i, \theta) = (o(Vg(Wx^i)) - y^i)^2$$

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

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

- (Stochastic) Gradient Descent

Convolution Layer

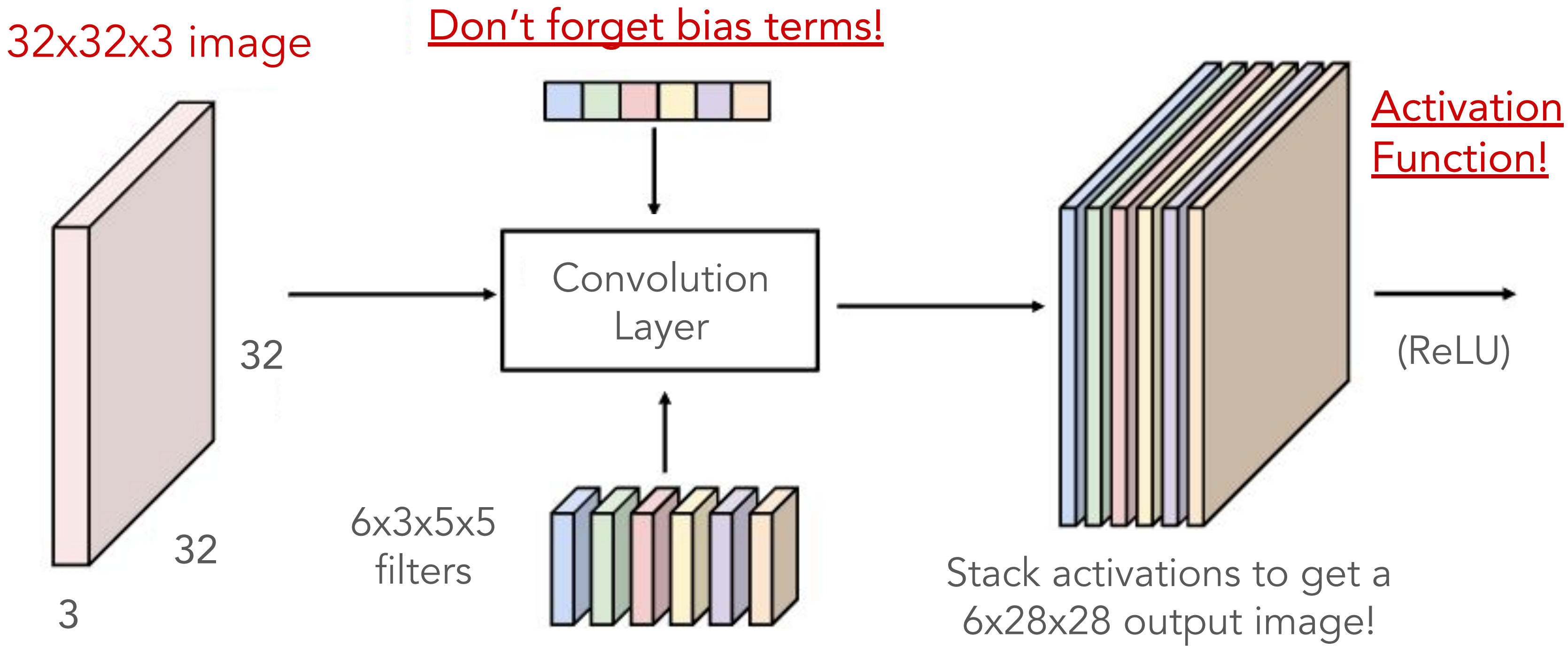


padding = 0, stride = 1 | padding = 1, stride = 2 | padding = 1, stride = 1 | padding = 2, stride = 1

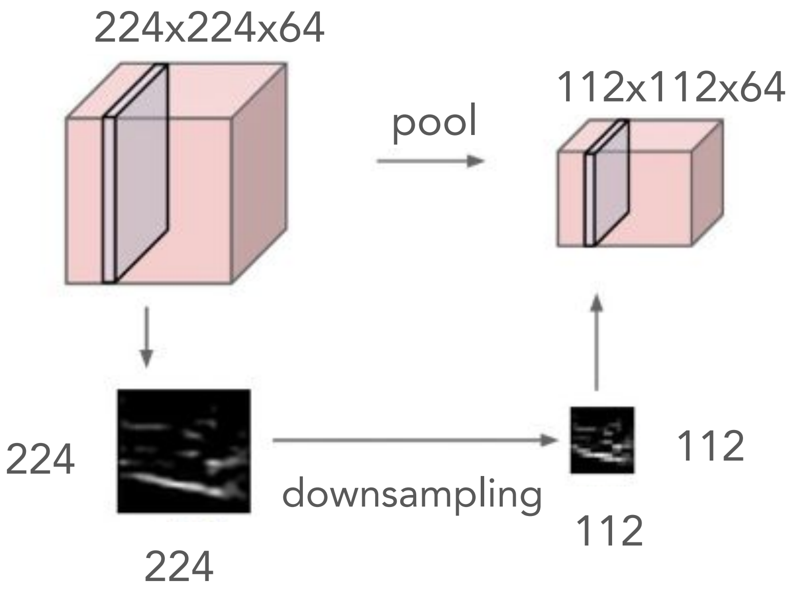
$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Animation from Hochschule der Medien

Convolution Layer

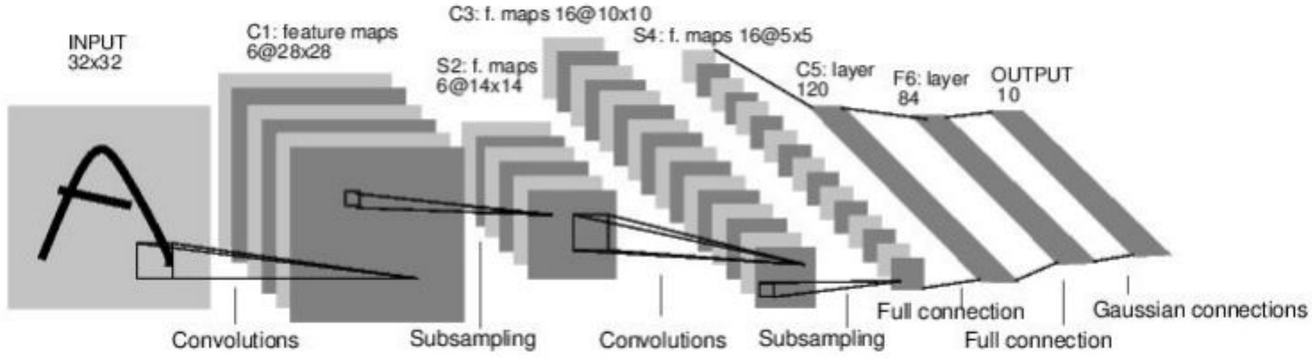


Pooling (Subsampling)



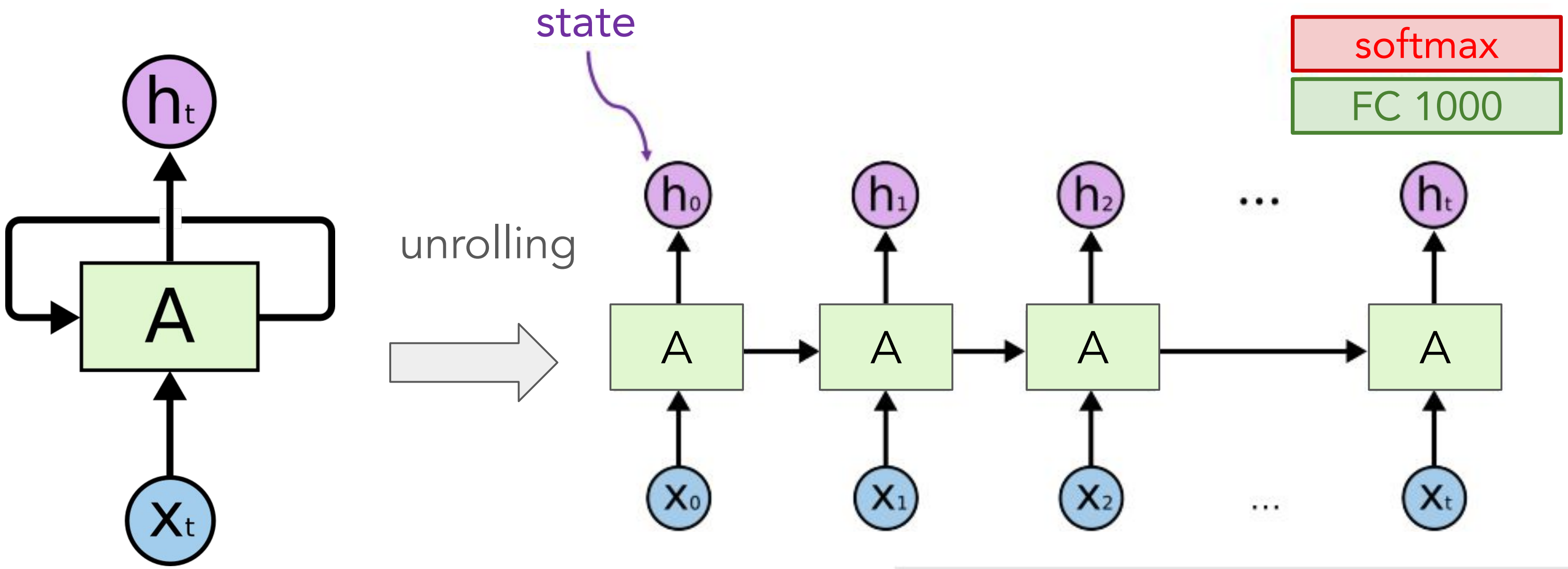
- Pooling layers simplify / subsample / compress the information in the output from the convolutional layer
- Reduce parameters

Put Everything Together



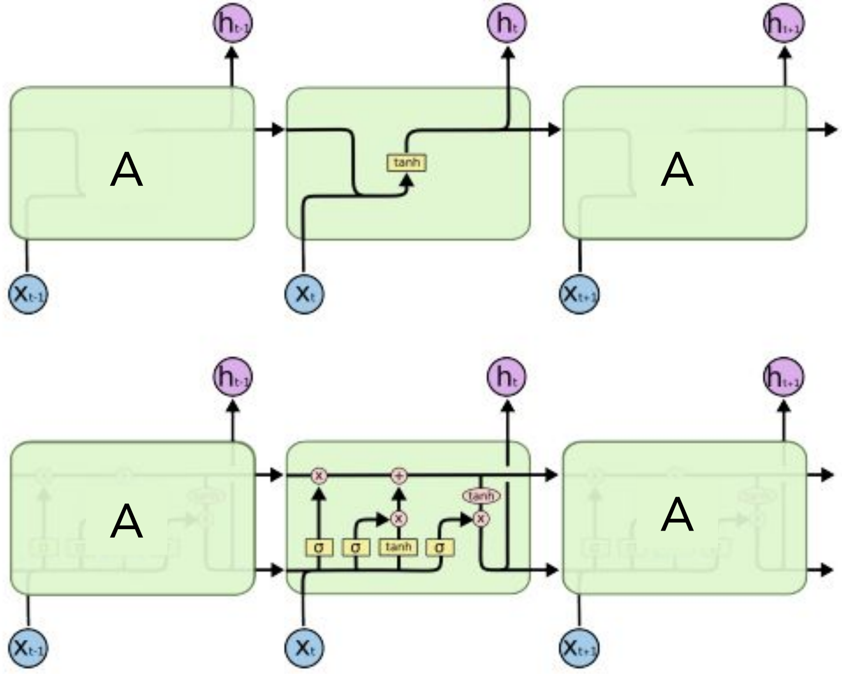
[LeNet-5, LeCun 1980]

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)

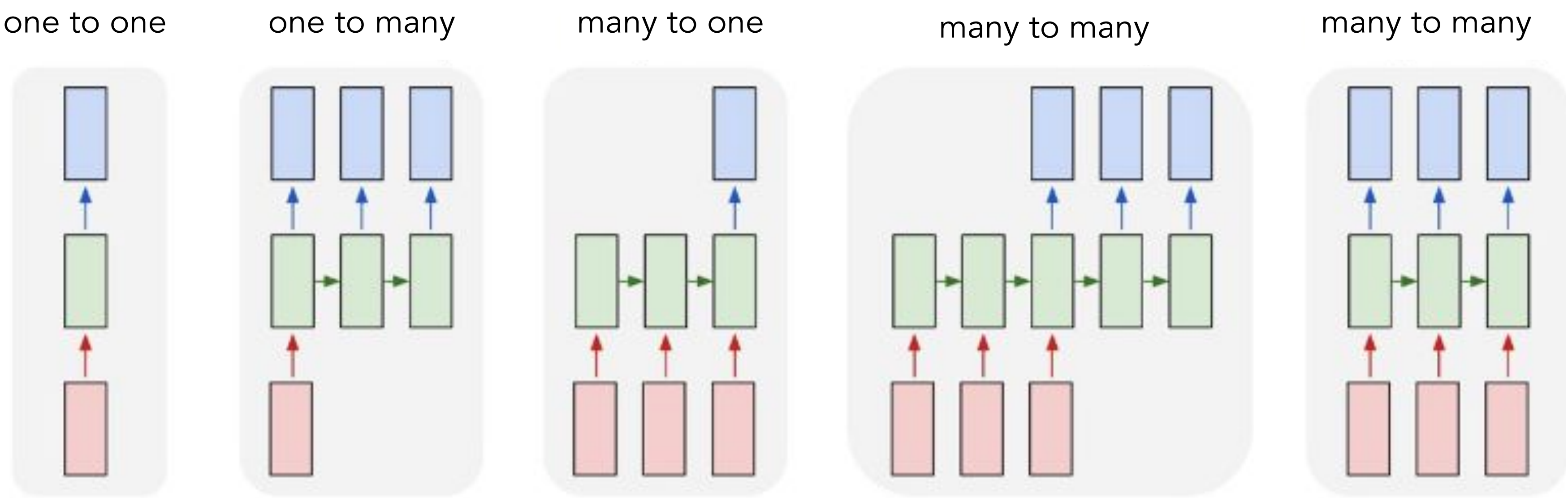


Simple RNN

LSTM

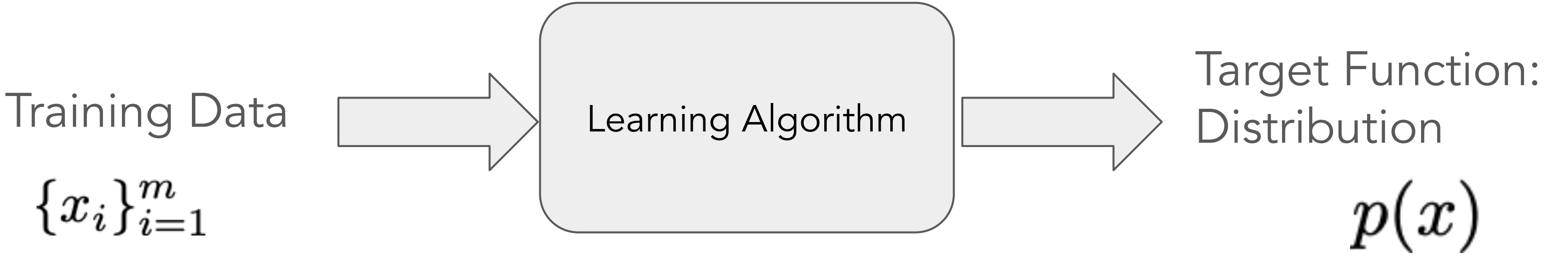
Figures from Christopher Olah's blog

Training of RNNs



Backpropagation Through Unrolling Steps

Density Estimation: Gaussian Mixture Model



Density Estimation Pipeline

1. Build probabilistic models
   - Gaussian Mixture Model
2. Derive loss function (by MLE or MAP….)
   - Approximate
   - MLE
3. Select optimizer
   - EM

Gaussian Mixture Model

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

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

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

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

What is the difference?

Expectation-Maximization

For t = 1……

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

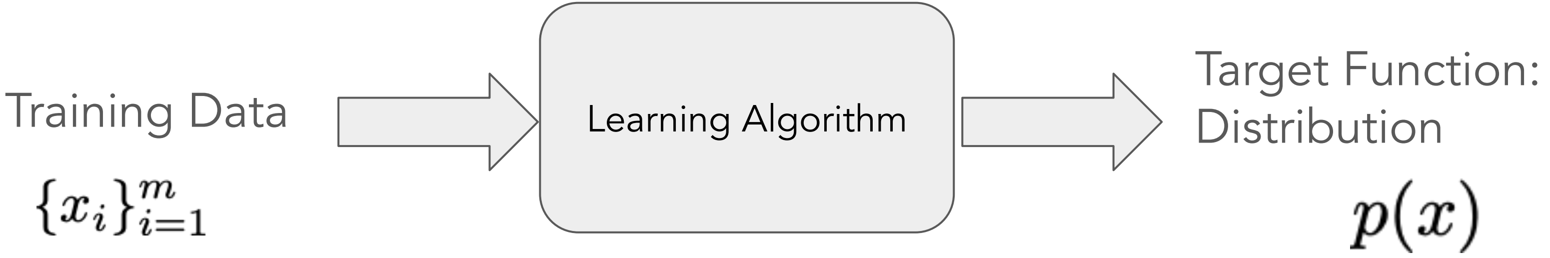
$$y_j^l = \frac{\pi_l \mathcal{N}(x_j | \mu_l, \Sigma_l)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_j | \mu_l, \Sigma_l)}$$

- M-Step: Update the parameters with current labels

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

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

$$\min_{\{\boldsymbol{\mu}\},\{\mathbf{y}\}} J(\{\boldsymbol{\mu}\},\{\mathbf{y}\}) = \min_{\{\boldsymbol{\mu}\},\{\mathbf{y}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)} \|\boldsymbol{\mu}_k - \mathbf{x}^{(n)}\|^2$$

$$\text{s.t. } \sum_k y_k^{(n)} = 1, \forall n, \text{where } y_k^{(n)} \in \{0,1\}, \forall k, n$$

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

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

  - (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......

- 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 (Gaussian-Naive Bayes)

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

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

Q&A