PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER ∞: SUMMARY

Teaching Objectives

- Fundamental knowledge about machine learning and pattern recognition, from Bayesian approaches to deep learning frameworks through lectures, quizzes and assignments

- Machine learning system development methods with Python through labs and projects

- Model-based and data-driven machine learning system design and integration skills through the final project, literature surveys and reports

Lecture Schedule

Section 0	Course Introduction	
Section 1	Preliminary	（HW1）
Section 2	Probability Distributions	（HW2）
Section 3	Linear Regression and Classification	（HW3, HW4）
Section 4	Neural Networks	（HW5）
Section 5	Sparse Kernel Machine	（HW6）
--Midterm Exam--		
Section 6	Mixture Models and EM learning	（HW7）
Section 8	Sequential Data (Hidden Markov Model)	（HW8）
Section 9	Bayesian Networks	
Section 10	Markov Decision Process	
Section 11	Reinforcement Learning	
--Final Exam--		

Lab Schedule

Section 0	Lab Introduction
Section 1	Preliminary
Section 2	Bayes
Section 3	Regression
--Final Project Proposal--	
Section 4	Decision Tree
Section 5	Support Vector Machine
Section 6	Convolution Neural Network
Section 7	K-Mean Clustering
Section 8	Gaussian Mixture Model
Section 9	Markov Decision Process
Section 10	Reinforcement Learning
--Final Project Report--	

Final Projects

[1] Reinforcement learning based planning using a self-driving car simulator

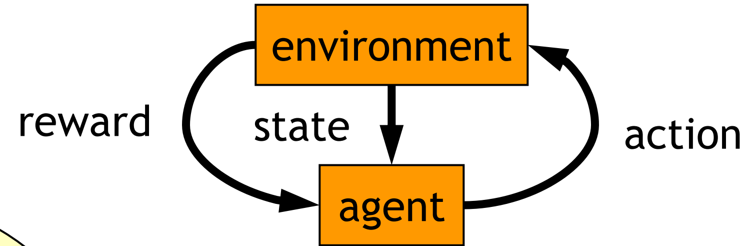
[2] Segmentation of 2D/3D measurements for self-driving applications

[3] Detection and recognition of traffic signs for self-driving applications

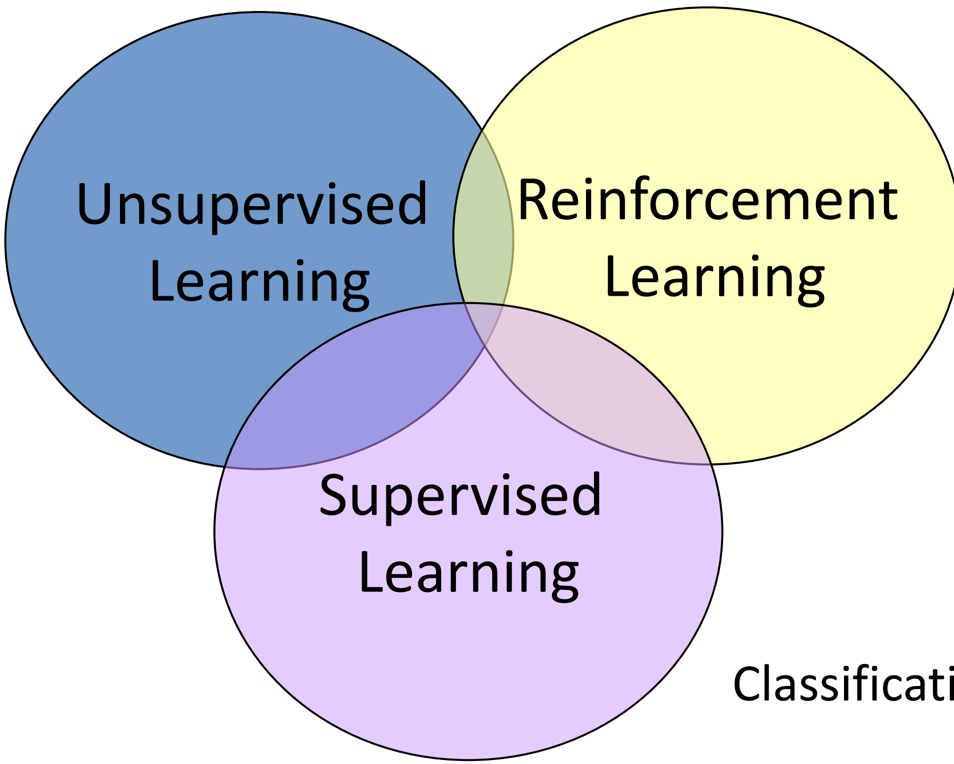
[4] Detection and tracking of 2D/3D objects for self-driving applications

[5] Generation of annotated self-driving datasets with the CARLA simulator

Machine Learning

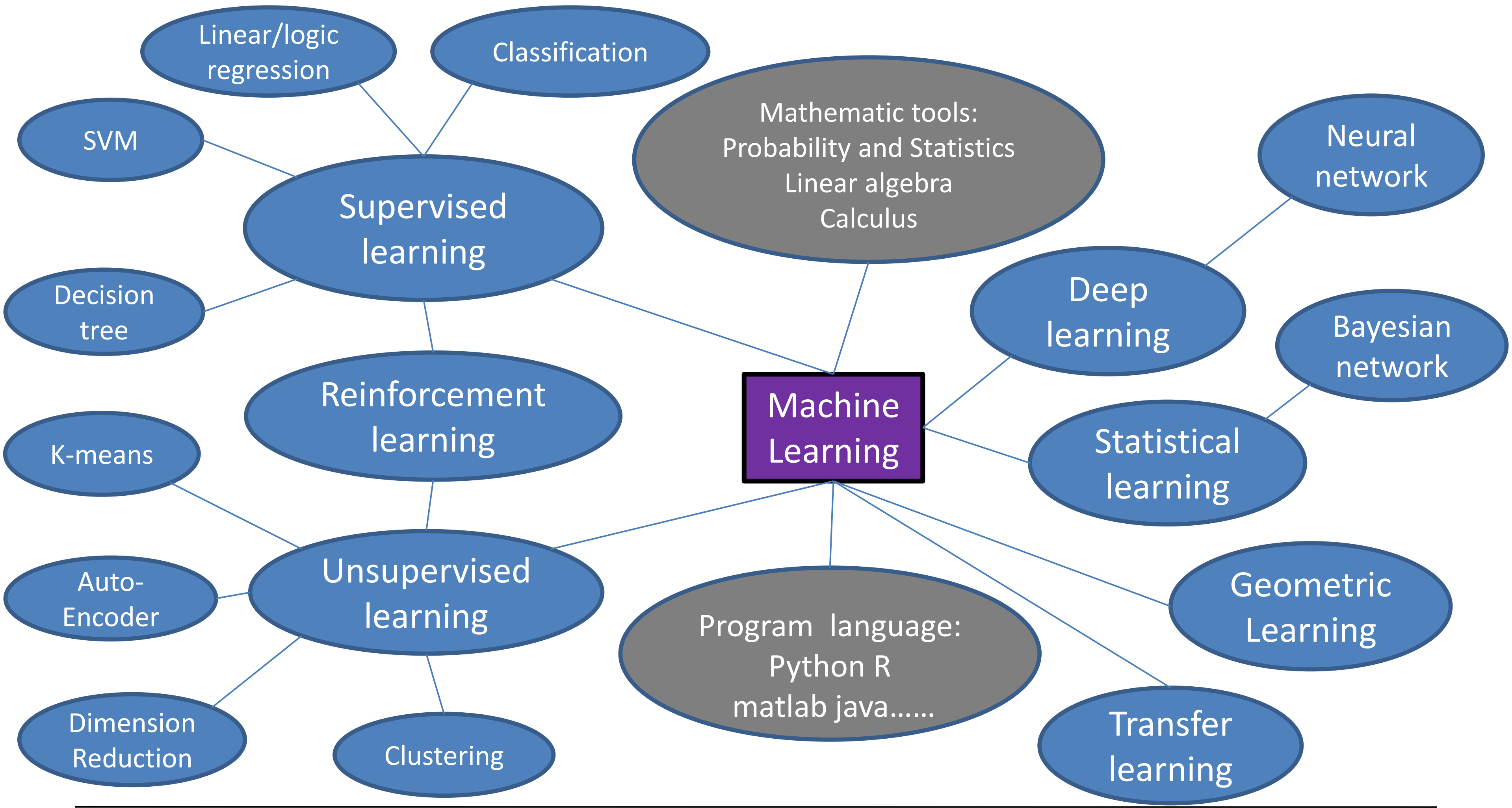


Clustering
Dimension reduction



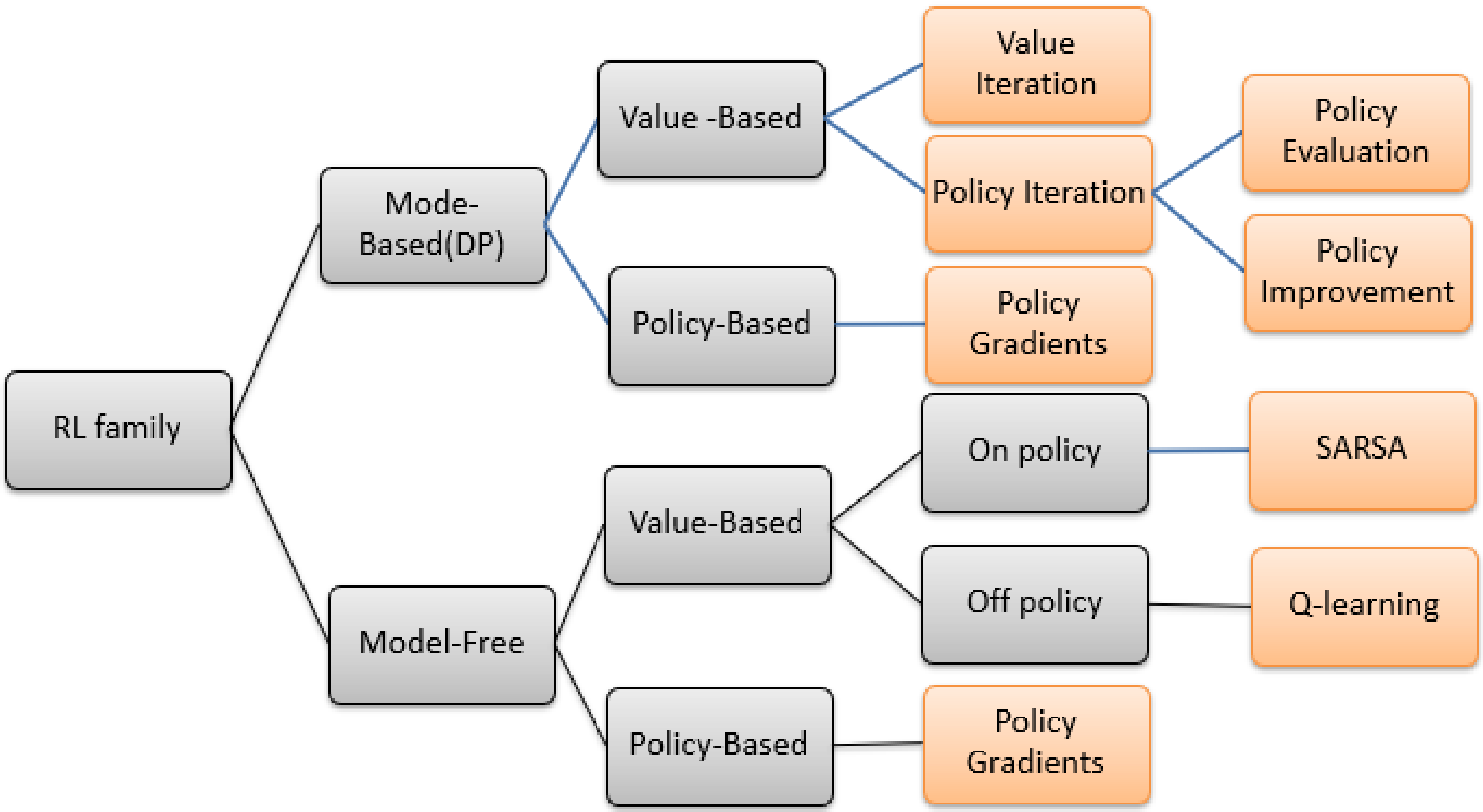
Classification, Regression

Framework

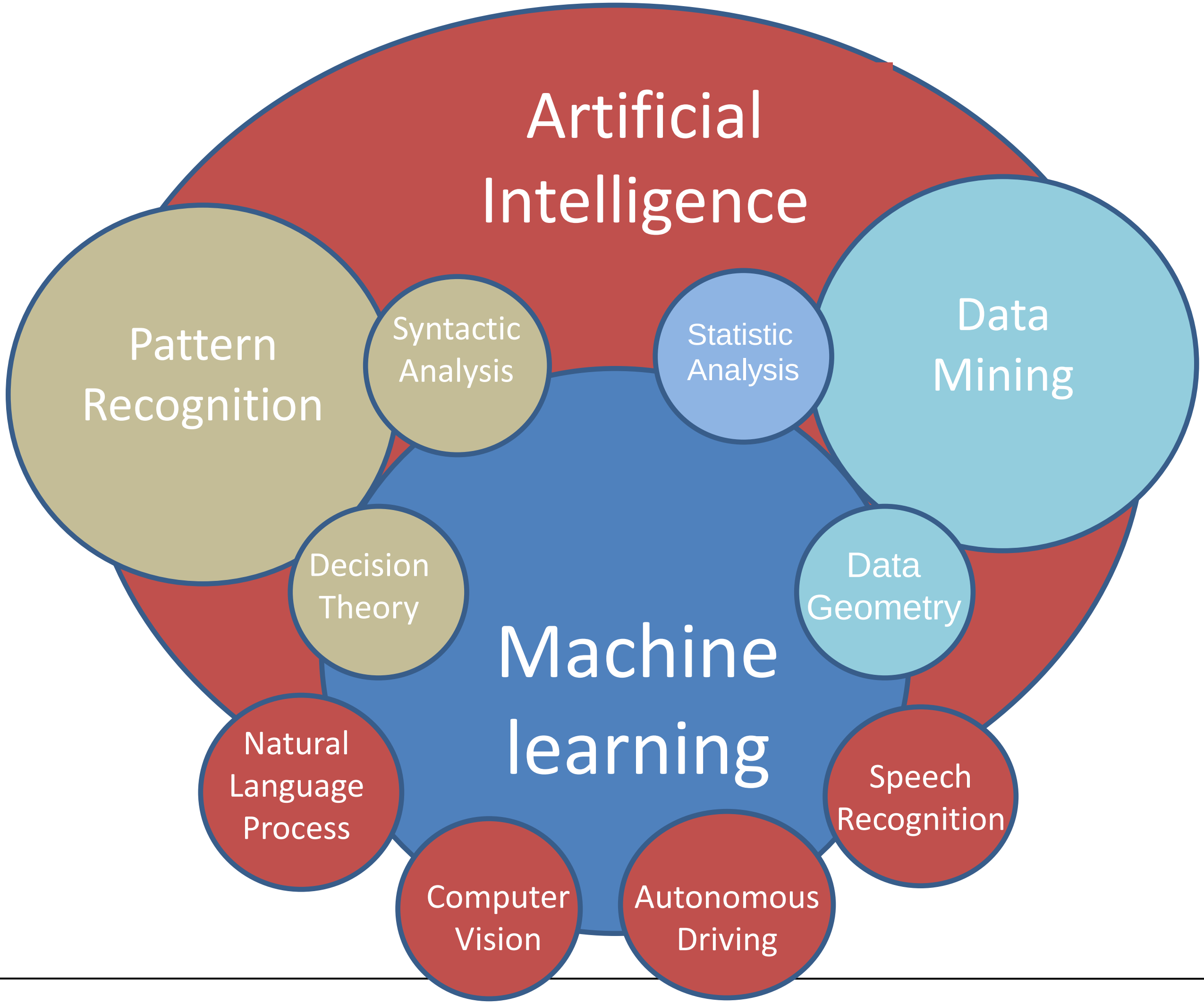


Linear/logic regression
Classification
SVM
Supervised learning
Decision tree
Mathematic tools:
Probability and Statistics
Linear algebra
Calculus
Neural network
Deep learning
Bayesian network
Reinforcement learning
Machine Learning
K-means
Statistical learning
Unsupervised learning
Auto-Encoder
Geometric Learning
Program language:
Python R
matlab java……
Dimension Reduction
Clustering
Transfer learning

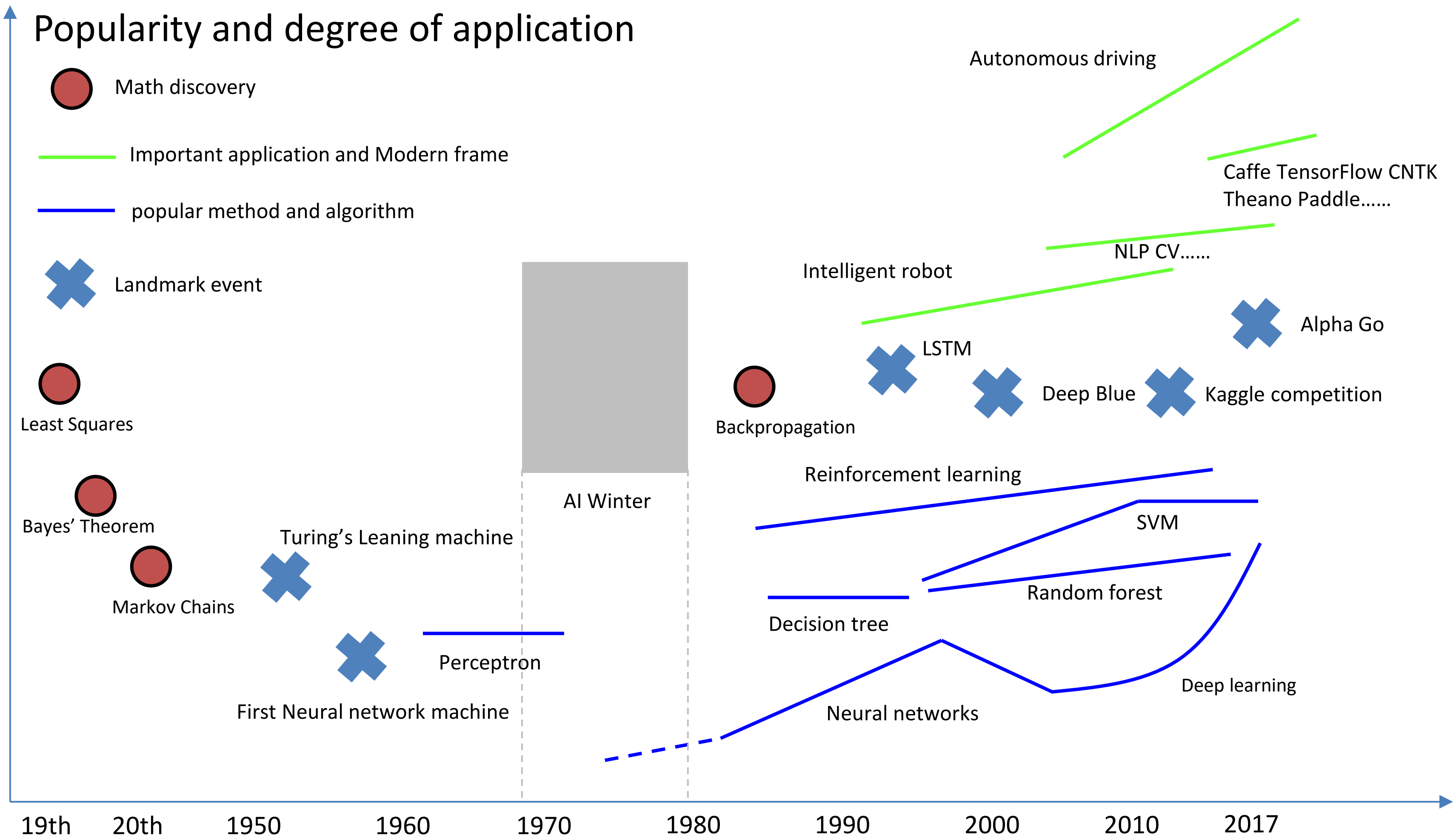
Reinforcement Learning



The Whole Picture



History


Popularity and degree of application
Math discovery
Important application and Modern frame
popular method and algorithm
Landmark event
Least Squares
Bayes' Theorem
Markov Chains
Turing's Leaning machine
First Neural network machine
Perceptron
AI Winter
Backpropagation
Intelligent robot
LSTM
Deep Blue
Kaggle competition
Alpha Go
NLP CV……
Caffe TensorFlow CNTK
Theano Paddle……
Autonomous driving
Reinforcement learning
SVM
Random forest
Decision tree
Neural networks
Deep learning
19th
20th
1950
1960
1970
1980
1990
2000
2010
2017

Machine learning

- **Machine Learning—minimization of some loss function for generalizing data sets with models.**

- **Datasets —annotated, indexed, organized**

- **Models —tree, distance, probabilistic, graph, bio-inspired**

- **Optimization —algorithms can minimize the loss.**

Datasets

- Collection
- Storage
- Annotation
- Indexing
- Organization
- Access

Simulators

- Data visualization
- Generate training data
- Algorithm evaluation

Benchmark Metrics

- System functionalities
- System scalability
- System robustness
- System efficiency

Models

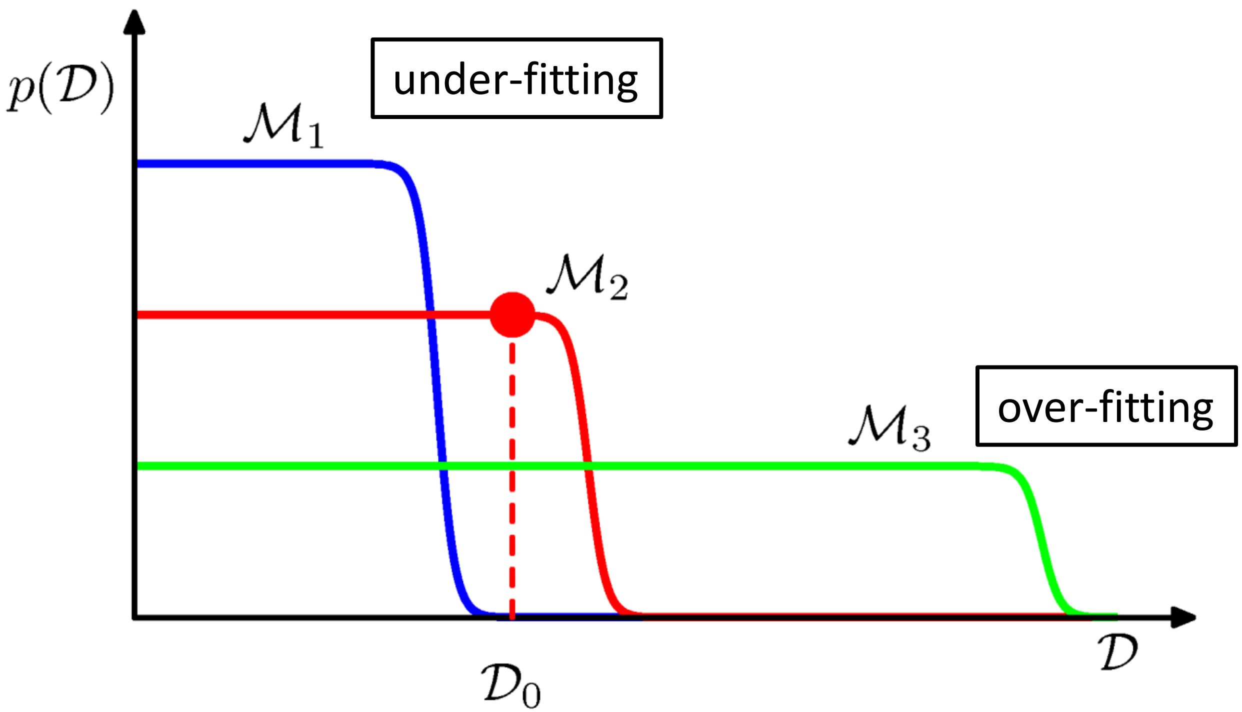
- Tree Models
- Distance-based Models
- Probabilistic Models
- Neural Network Models
- Graph-based Models

Models

- Boosting
- Ensemble Learning

Model Comparison

- Matching data and model complexity



Machine learning and Optimization

- **Machine Learning—minimization of some loss function for generalizing data sets with models.**

- **Datasets —annotated, indexed, organized**

- **Models —tree, distance, probabilistic, graph, bio-inspired**

- **Optimization —algorithms can minimize the loss.**

Problem Statement

- **Problem:** Predict the label $\hat{Y}$ and data $\hat{X}$ with training set (X,Y)?

Unknown model

Training label

Training data

Training: $Y = \underline{A} \circ X$

Prediction: $\hat{Y} = A \circ \hat{X}$

Predicting label

Predicting data

$$\begin{cases} Y \text{ and } X \text{ is known}: & \text{supervised learning} \\ Y \text{ or } X \text{ is unknown}: & \text{unsupervised learning} \end{cases}$$

$$\begin{cases} Y、\hat{Y} \text{ are continuous}: & \text{Regression} \\ Y、\hat{Y} \text{ are discrete}: & \text{classification} \end{cases}$$

Y *is known and* $Dim(Y) > Dim(X)$: dimensionality reduction

Optimization

- Finding (one or more) minimizer of a function subject to constraints

$$arg \min_{x} f_0(x)$$

$$s.t. f_i(x) \leq 0, i = \{1, \dots, k\}$$

$$s.t. h_j(x) = 0, j = \{1, \dots, l\}$$

- Most of the machine learning problems are, in the end, optimization problems

Lagrange Method

- Minimize an object function *s. t.* constraints of inequality

$$\min_{x} f(x) \quad s.t. \quad g(x) \geq 0$$

- By Introducing a Lagrange multiplier $\lambda \geq 0$, then we will have

$$\min_{x} \max_{\lambda \geq 0} \{\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)\}$$

- When certain conditions are satisfied, its dual problem is

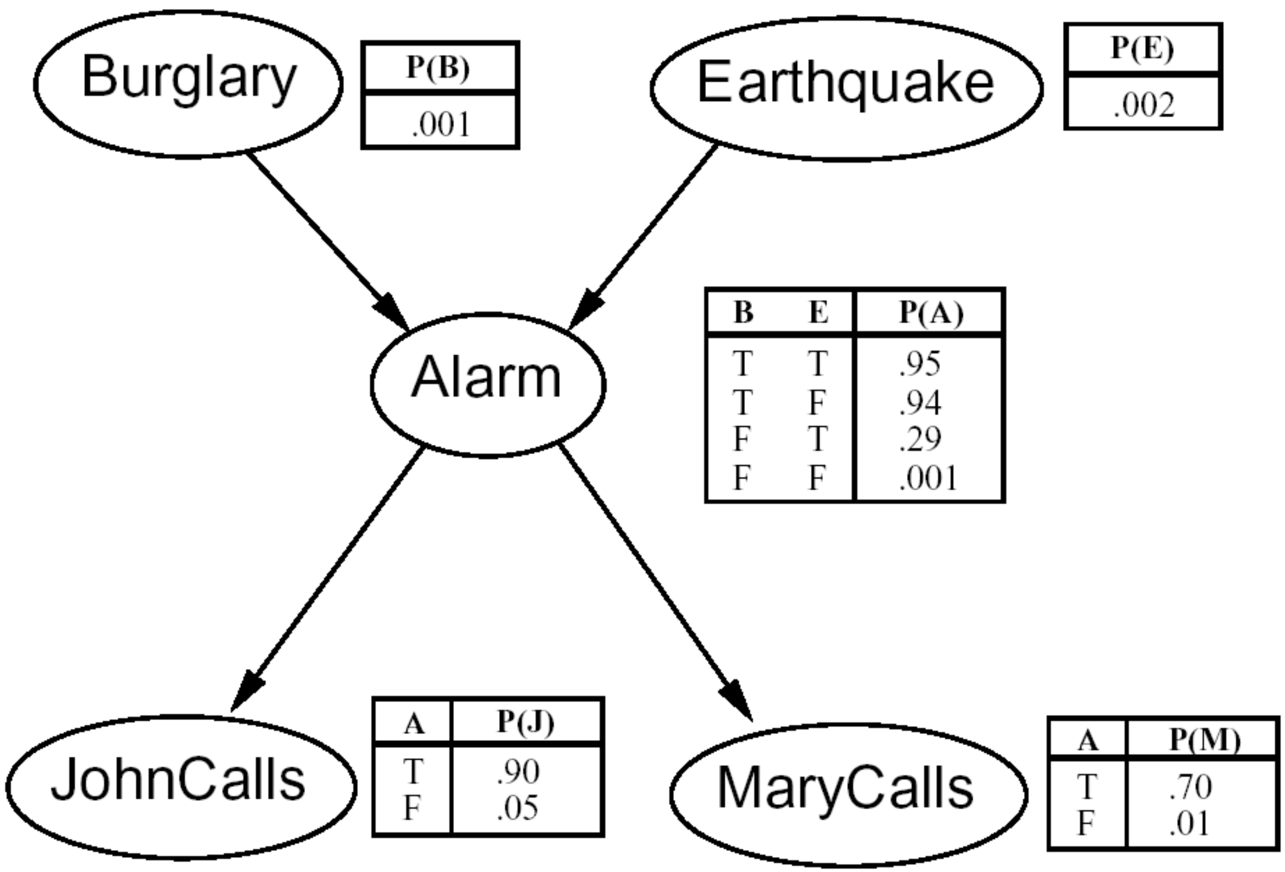
$$\max_{\lambda \geq 0} \min_{x} \{\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)\}$$

- By setting derivatives of $\mathcal{L}$ w.r.t. x equal to 0, we will have

$x = h(\lambda)$, and then the problem becomes $\lambda^* = \max_{\lambda \geq 0} Q(\lambda)$

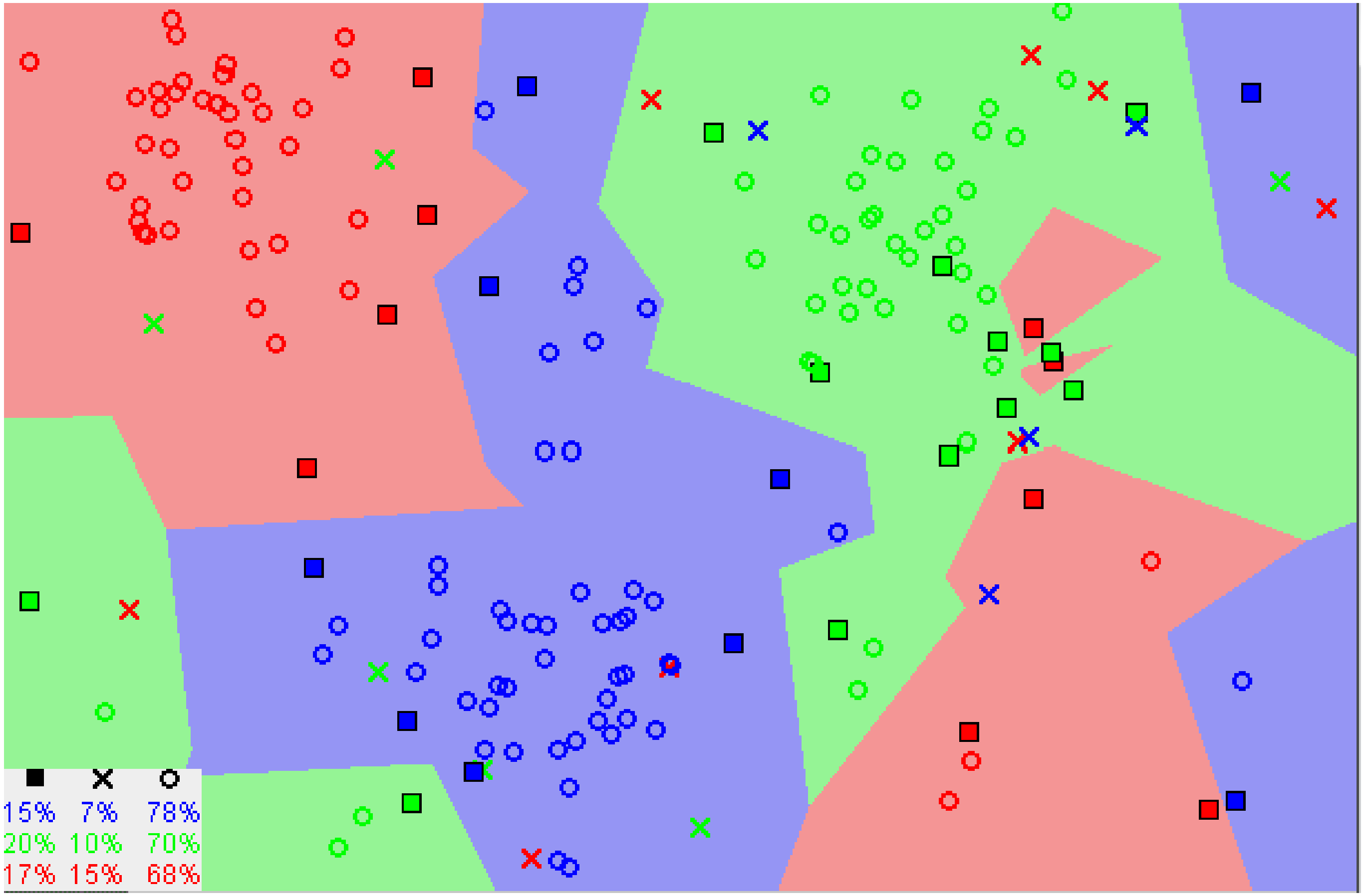
- Finally, $x^* = h(\lambda^*)$ is the solution

Bayes


Burglary
P(B)
.001
Earthquake
P(E)
.002
Alarm
B E P(A)
T T .95
T F .94
F T .29
F F .001
JohnCalls
A P(J)
T .90
F .05
MaryCalls
A P(M)
T .70
F .01

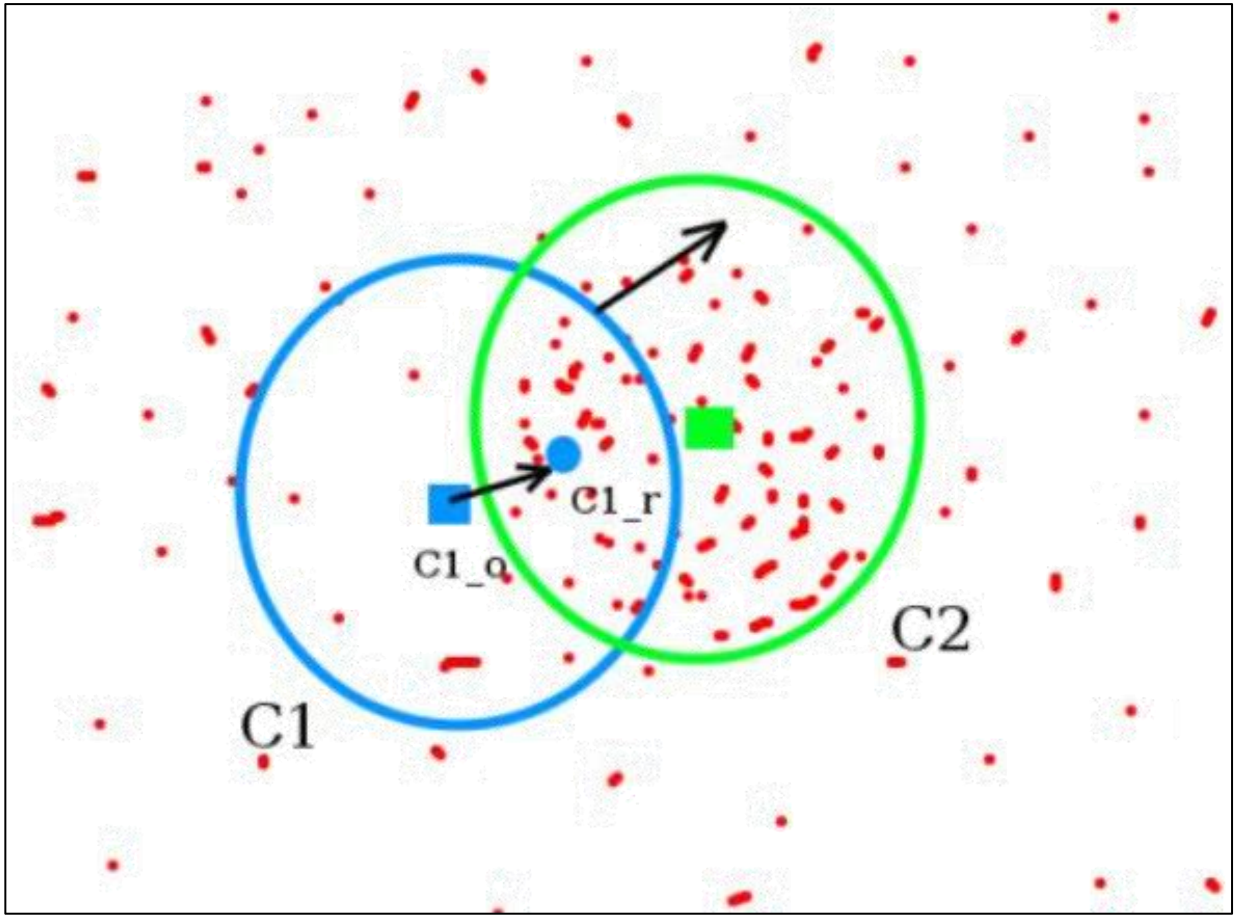
K-Nearest Neighbors

- Use training data for classification



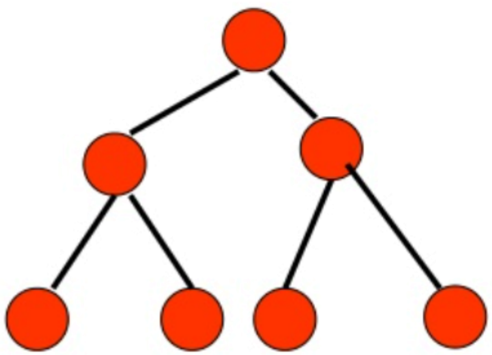
K-Means

- Mean-shift



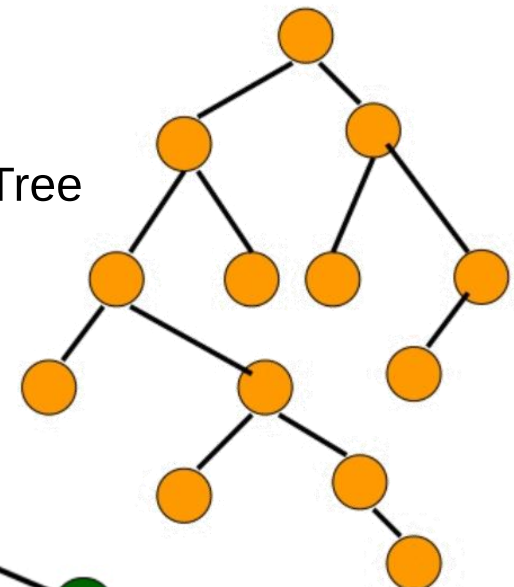
Decision Tree

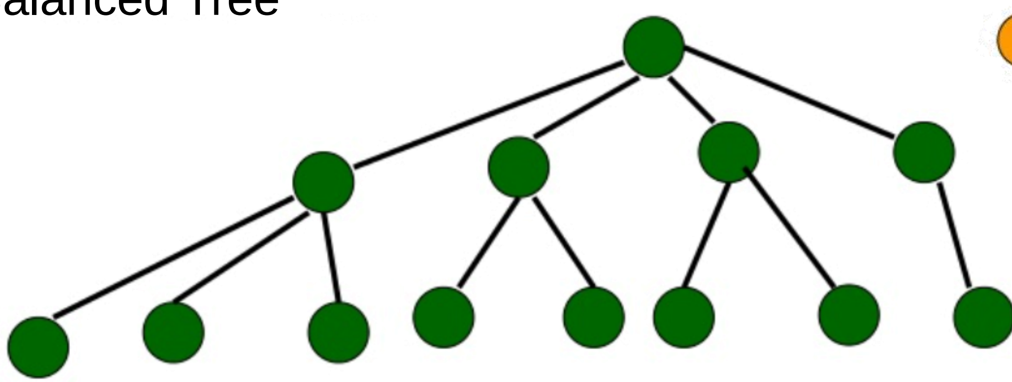
- Types of Decision Tree:



Balanced Tree

Deep Tree

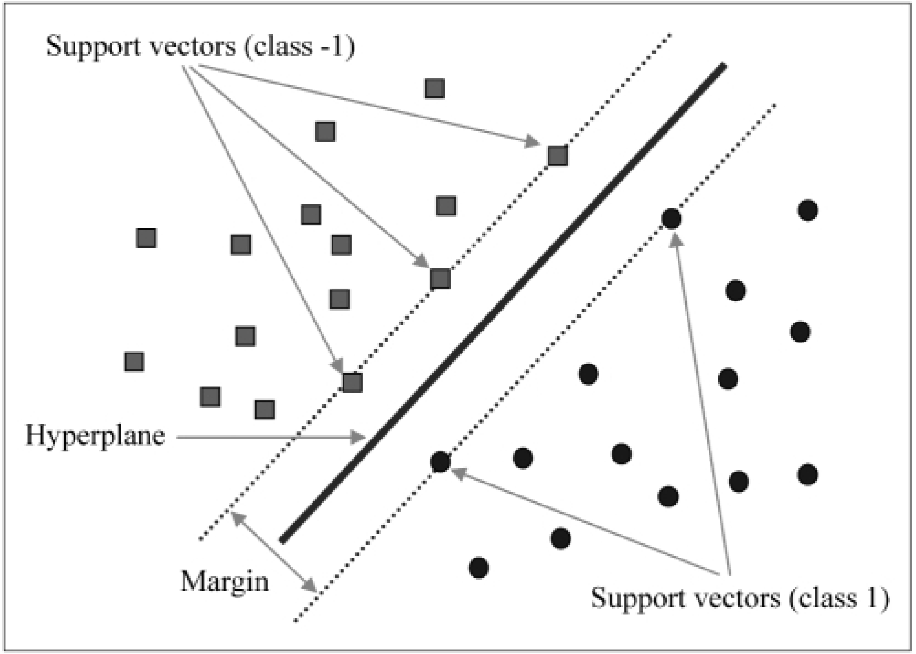


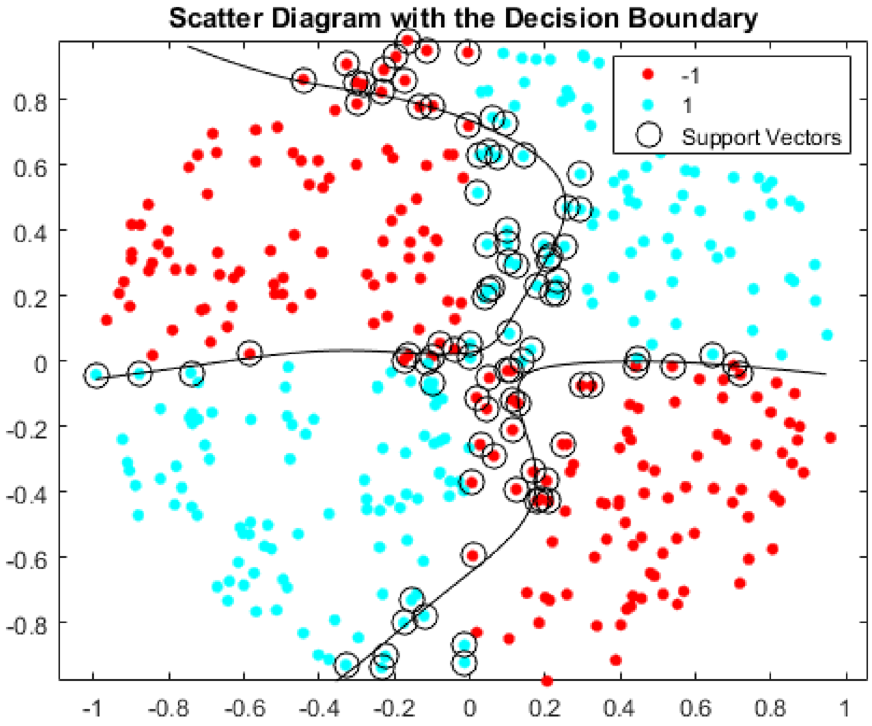


Bushy Tree

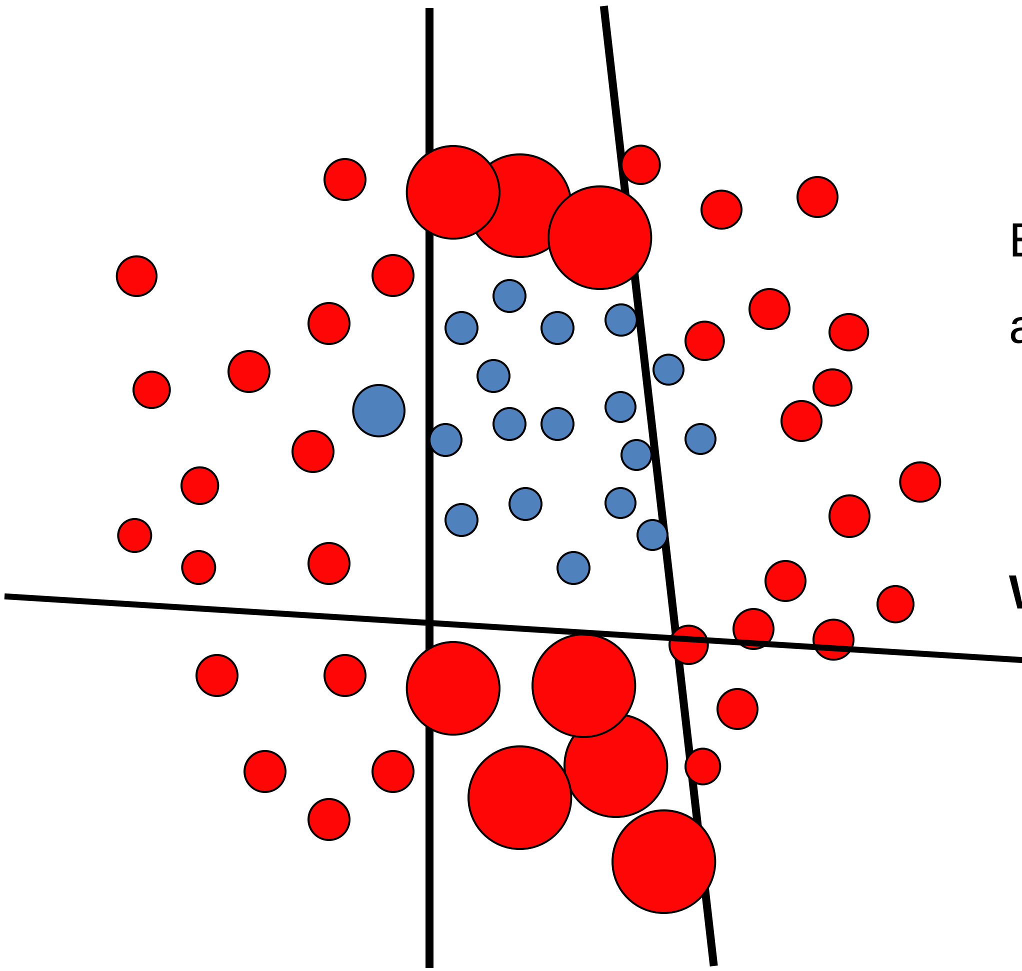
SVM

- Linear SVM: $$\arg\min_{w} \sum_{i=1}^{n} ||w||^2 + C \sum_{i=1}^{n} \xi_i$$
$$\text{s.t. } 1 - y_i x_i^T w \leq \xi_i$$
$$\xi_i \geq 0$$





Boosting



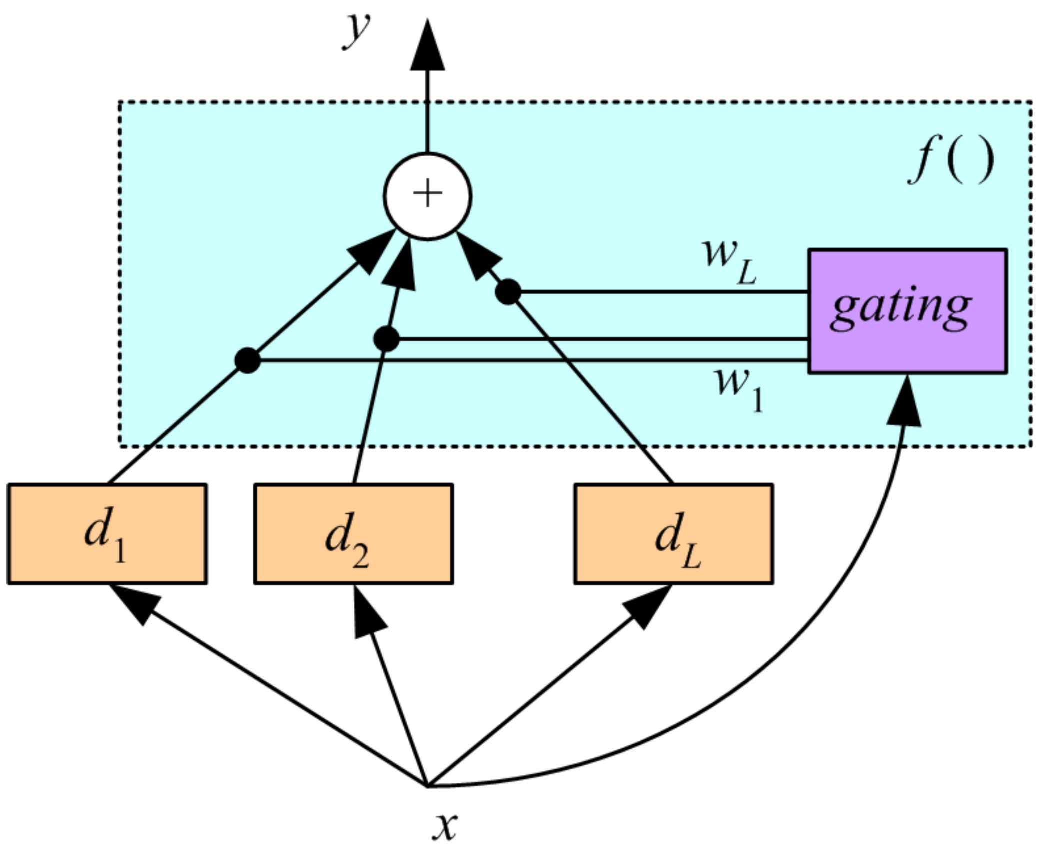
Each data point has a class label:

$$y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Ensemble Learning



$$y = \sum_{j=1}^{L} w_j d_j$$

Linear Statistical Learning

- **PCA**

  Basis

  Data

  Coefficients

  $$Y = AX$$

  $A_i \perp A_j$

- **ICA**

  Mixture Coefficients

  Data

  Components

  $$Y = AX$$

  $\max\ I(X)$

- **NMF**

  Basis

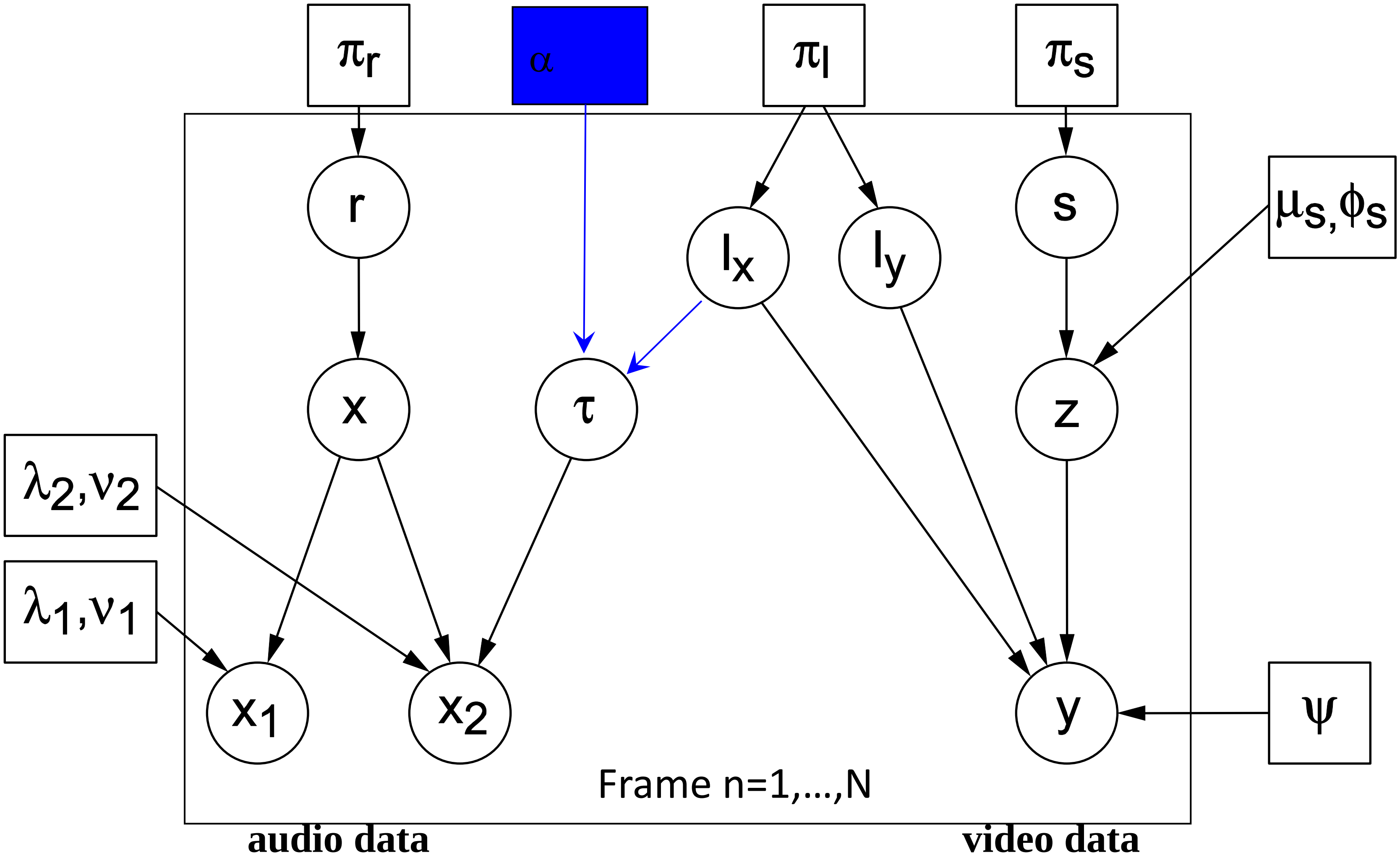
  Data

  Coefficients

  $$Y = AX$$

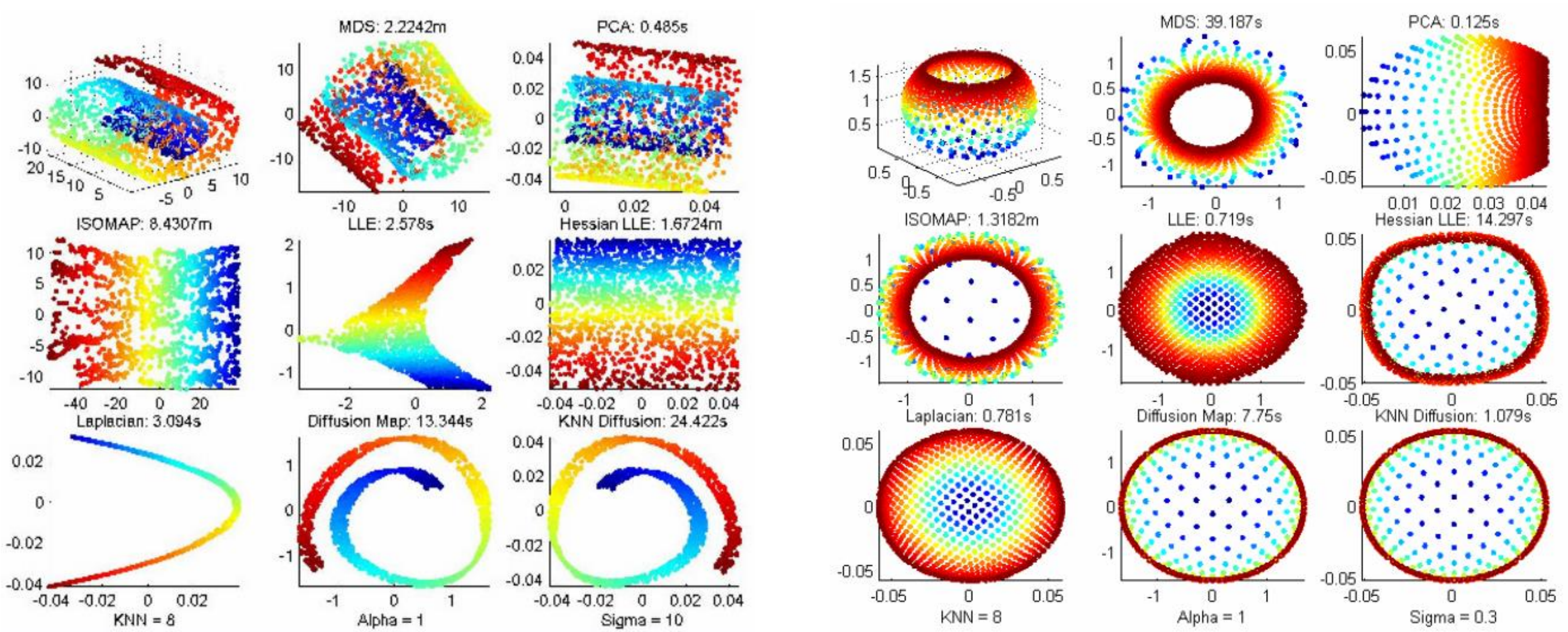
  $A, X > 0$

Bayesian Networks

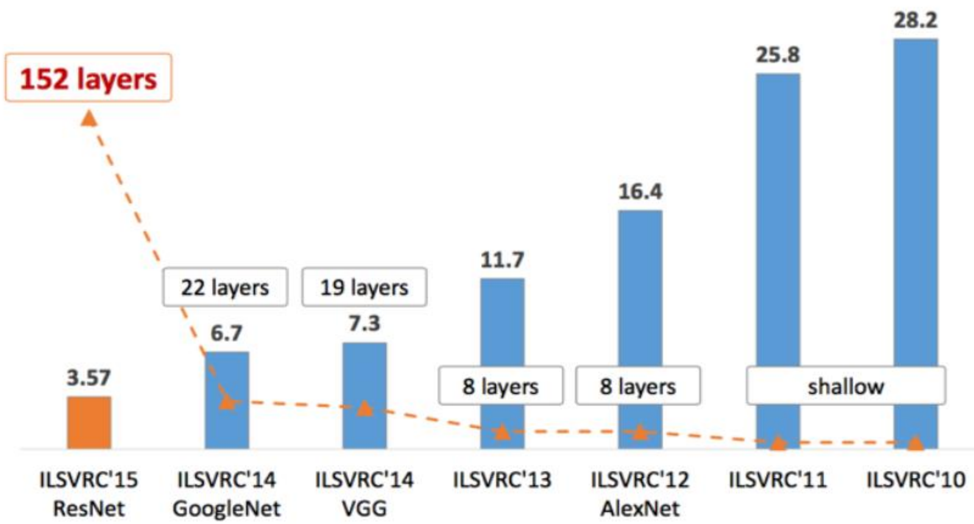


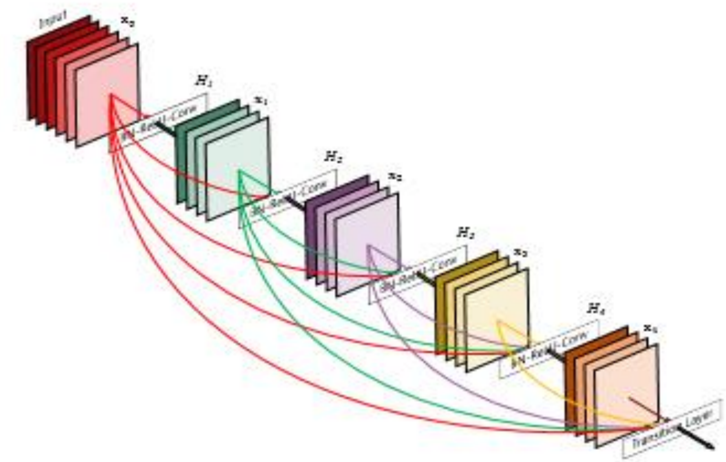
Nonlinear Statistical Learning

- Manifold learning



Deep Neural Networks

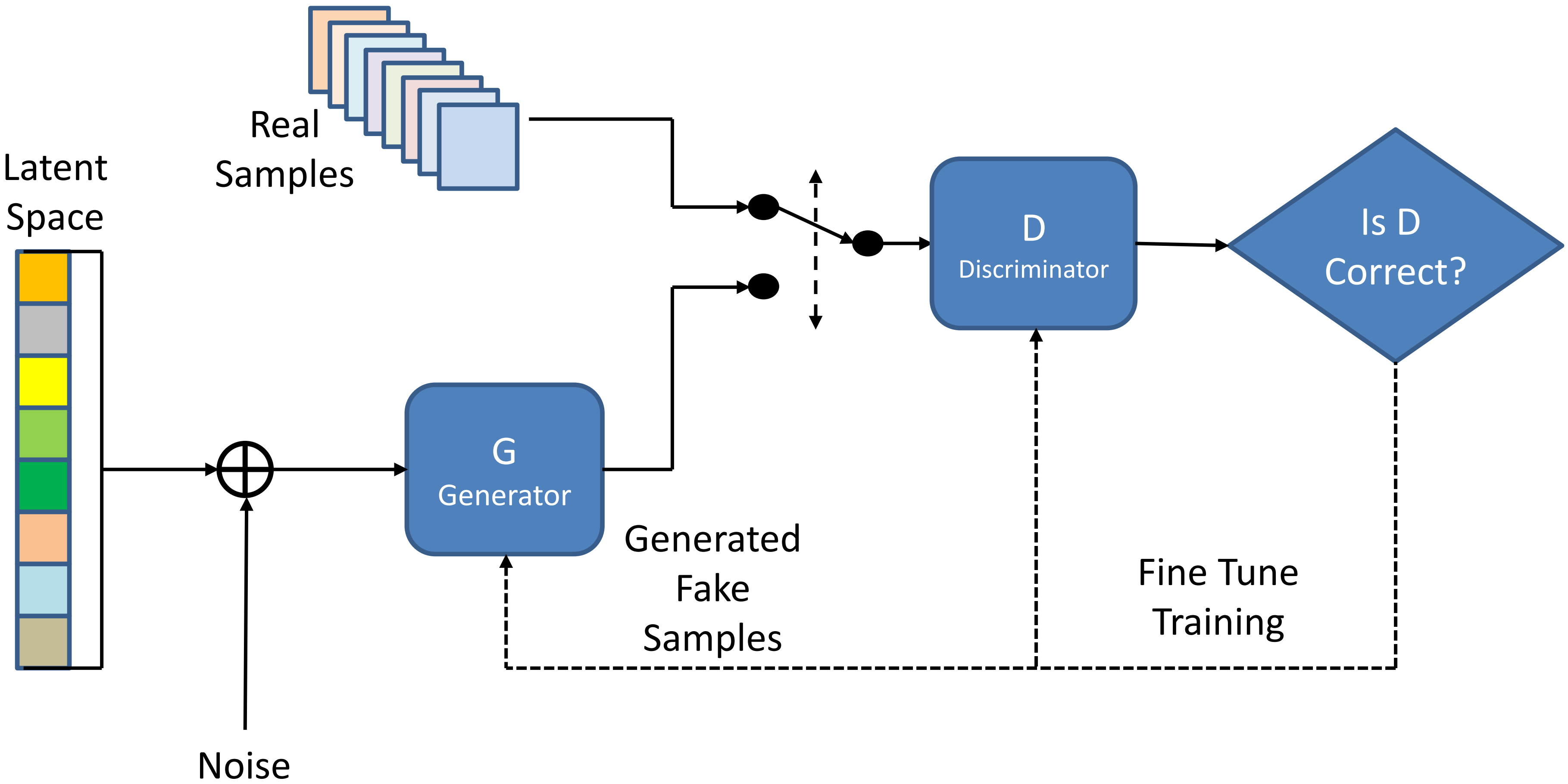




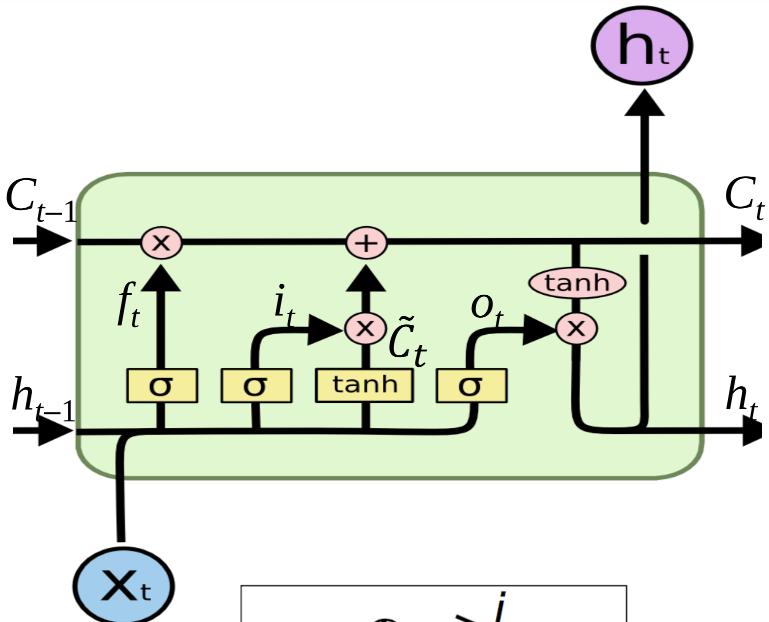
- Task： recognition
- Dataset： ILSVRC

■ Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks[J]. arXiv preprint arXiv:1608.06993, 2016.

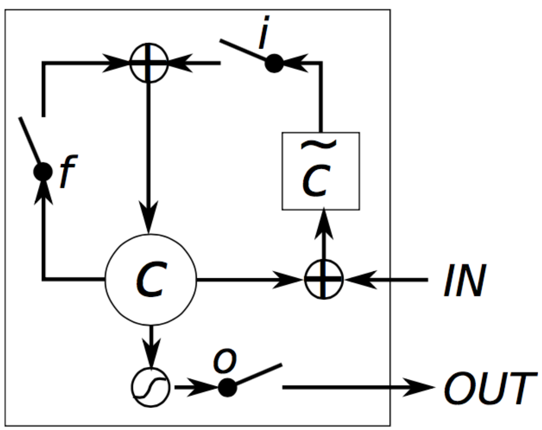
Generative Adversarial Networks



Long Short Term Memory

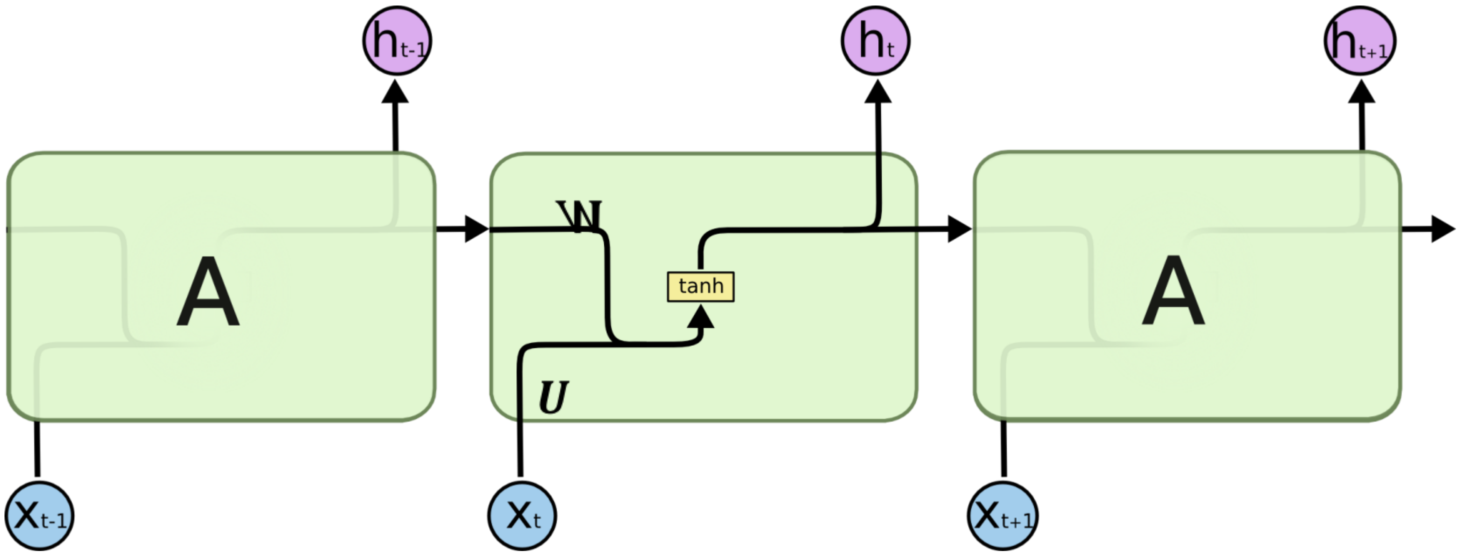


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



C_t: cell state
$\tilde{C}_t$: cell state prediction
f_t : forget gate
i_t : input gate
o_t : output gate
h_t : output
x_t : input

Recurrent Neural Networks



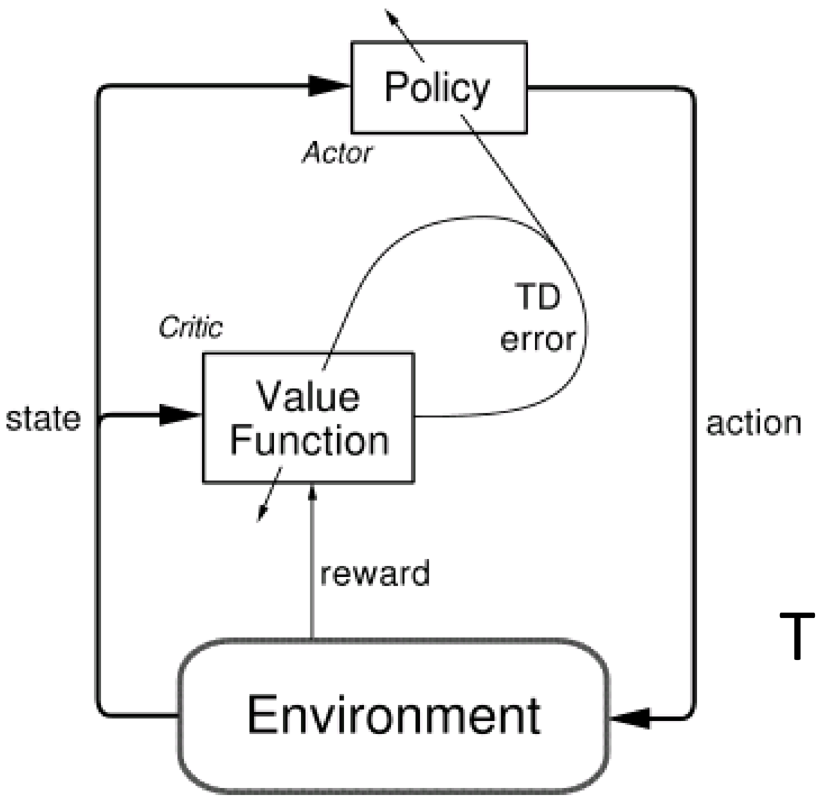
$$h_t = f(Ux_t + Wh_{t-1} + b)$$

$\mathbf{h}_t$: output

x_t : input

Reinforcement Learning

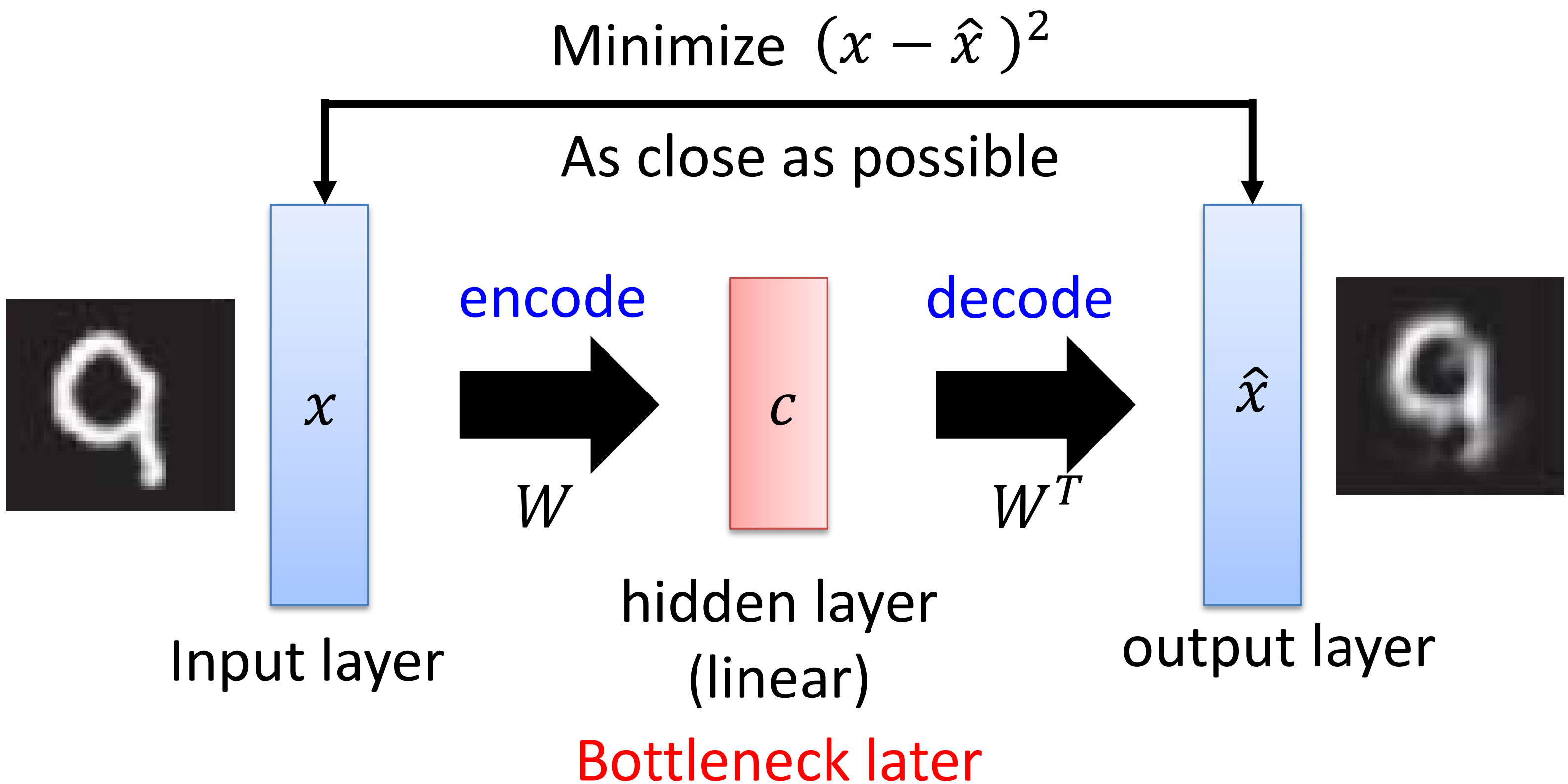
- State, action, and Reward



Update: Policy Function
Value Function

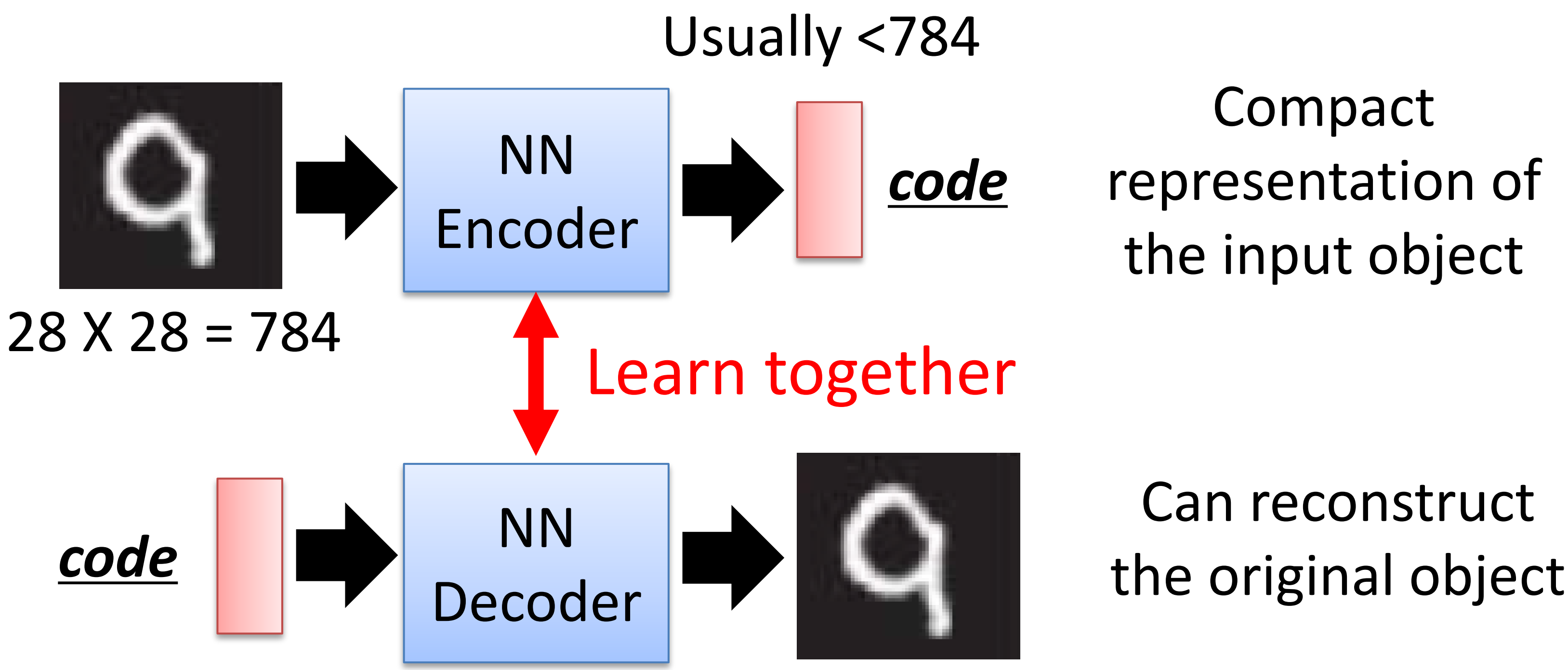
TD Error: Temporal Difference
between Real Reward
and Estimated Reward

PCA-Encoder



Minimize $(x - \hat{x})^2$
As close as possible
encode
decode
x
c
$\hat{x}$
W
W^T
Input layer
hidden layer
(linear)
output layer
Bottleneck later

Auto-Encoder



Gaussian Models

- Joint and Marginal Models
  - ✓ Joint probability
  - ✓ Marginal probability

- Conditional Models
  - ✓ Conditional probability
  - ✓ Posterior probability

- Predictive Models
  - ✓ Prior predictive
  - ✓ Posterior predictive

- Conjugate Prior Model
  - ✓ Gaussian
  - ✓ Gaussian-Gamma
  - ✓ Gaussian-Washart

Bernoulli and 1-out-of-K Models

- Probability Models
  - ✓ Bernoulli
  - ✓ 1-out-of-K

- Conjugate Prior Model
  - ✓ Beta
  - ✓ Dirichlet

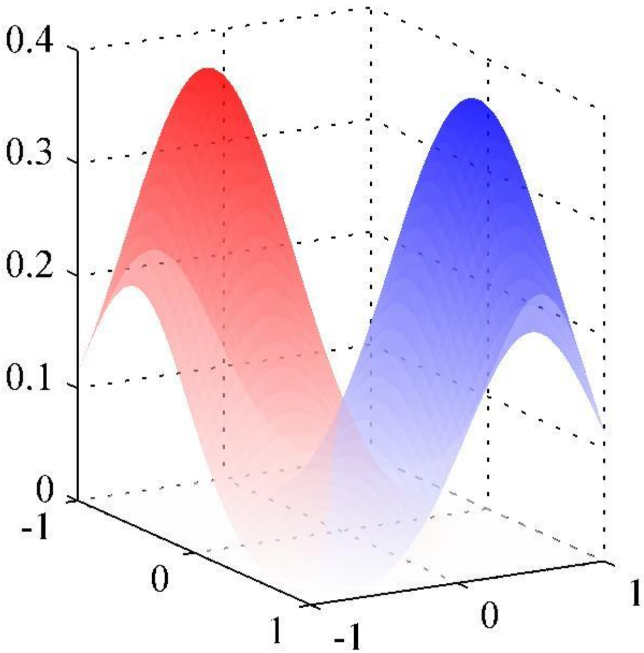
Bayesian Machine Learning

- Maximum Likelihood
  - ✓ Dataset
  - ✓ Error square cost function (model parameters)

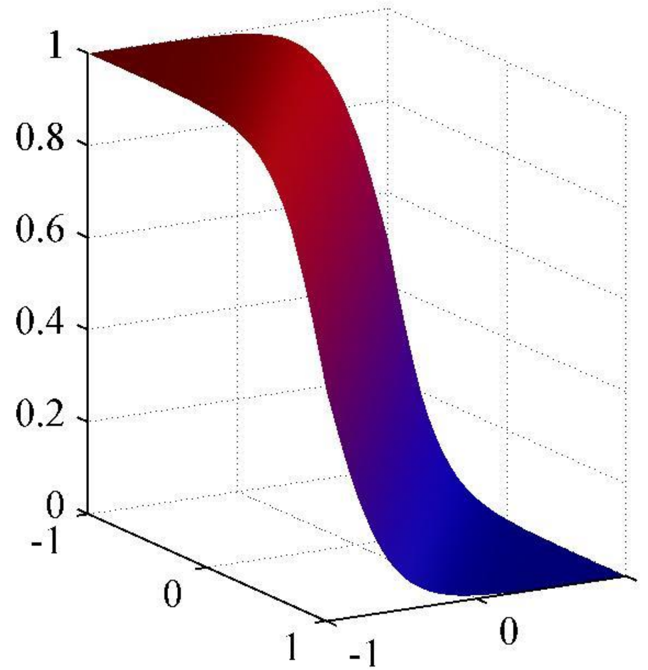
- Maximum A Posterior
  - ✓ Dataset
  - ✓ Cost function and regularization (model parameters)

- Expectation and Maximization
  - ✓ Expectation (hidden variables)
  - ✓ Maximization (model parameters)

Gaussian Mixture



$p(x|C_0)$ $p(x|C_1)$



$p(C_0|x)$ $p(C_1|x)$

ML Solution to Gaussian Mixtures

$$p(x, C_1) = p(C_1)p(x|C_1) = \pi N(x|\mu_1, \Sigma)$$

$$p(x, C_2) = p(C_2)p(x|C_2) = (1-\pi)N(x|\mu_2, \Sigma)$$

Likelihood $\quad p(\boldsymbol{t}, \boldsymbol{X}|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} [\pi N(x_n|\mu_1, \Sigma)]^{t_n} [(1-\pi)N(x_n|\mu_2, \Sigma)]^{1-t_n}$

$$\Rightarrow \qquad \pi = \frac{1}{N}\sum_{n=1}^{N} t_n = \frac{N_1}{N} \qquad \mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n x_n \qquad \mu_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-t_n)x_n$$

$$\Sigma = \pi\Sigma_1 + (1-\pi)\Sigma_2 \qquad \Sigma_i = \frac{1}{N_i}\sum_{x_n \in C_i}(x_n - \mu_i)(x_n - \mu_i)^T \quad \text{i=1,2}$$

Generative: MAP Gaussian Mixtures

$$\pi_0 = \frac{N_{10}}{N_{10} + N_{20}} \qquad x \in \mathcal{C}_i \sim \mathcal{N}(x|\mu_{i0}, \Sigma_{i0})$$

$$\pi_{MAP} = \frac{N_1 + N_{10}}{N + N_0} = \frac{N_1 + N_{10}}{N_1 + N_2 + N_{10} + N_{20}}$$

$$\begin{cases} {\sum_{iMAP}}^{-1} = {\sum_{iML}}^{-1} + {\sum_{i0}}^{-1} \\ {\sum_{iMAP}}^{-1} \mu_{iMAP} = {\sum_{iML}}^{-1} \mu_{iML} + {\sum_{i0}}^{-1} \mu_{i0} \end{cases}$$

$$\Sigma = \pi \Sigma_1 + (1 - \pi) \Sigma_2$$

Logistic Regression

- When there are only two classes we can model the conditional probability of the positive class as

$$p(C_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0) \qquad where \quad \sigma(z) = \frac{1}{1+\exp(-z)}$$

- If we use the right error function, something nice happens: The gradient of the logistic and the gradient of the error function cancel each other:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}), \qquad \nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\, \mathbf{x}_n$$

ML Solution to Logistic Regression

$$p(C_0|\phi) = y(\phi) = \sigma(w^T\phi) \qquad p(C_1|\phi) = 1 - p(C_1|\phi)$$

where $$\frac{d\sigma(a)}{da} = \sigma(1-\sigma)$$

$$p(t|w) = \prod_{n=1}^{N} {y_n}^{t_n}(1-y_n)^{1-t_n}$$

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N}[t_n \ln y_n + (1-y_n)\ln(1-y_n)]$$ Likelihood

$$\nabla E(w) = \sum_{n=1}^{N}(y_n - t_n)\phi_n \qquad H = \nabla\nabla E(w) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n{\phi_n}^T$$

$$w_{ML} \longleftarrow w^{new} = w^{old} - H^{-1}\nabla E(w) \qquad q(w) = N(w|w_{ML}, H^{-1})$$

MAP Solution to Logistic Regression

$$p(w) = N(w|m_0, S_0) \qquad p(w|t) \propto p(w)p(t|w)$$

$$E(w) = -\ln p(w|t) = \frac{1}{2}(w - m_0)^T {S_0}^{-1} (w - m_0) - \sum_{n=1}^{N} [t_n \ln y_n + (1 - y_n) \ln(1 - y_n)]$$

$$\nabla E(w) = {S_0}^{-1}(w - m_0) + \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

$$H = \nabla\nabla E(w) = {S_0}^{-1} + \sum_{n=1}^{N} y_n(1 - y_n)\phi_n{\phi_n}^T$$

$$w_{\mathrm{MAP}} \longleftarrow w^{new} = w^{old} - H^{-1}\nabla E(w) \qquad q(w) = N(w|w_{\mathrm{MAP}}, H^{-1})$$

Feed-forward Network Functions

- Goal: to extend linear model by making the basis functions depend on parameters, allow these parameters to be adjusted.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

First layer

activations

weights

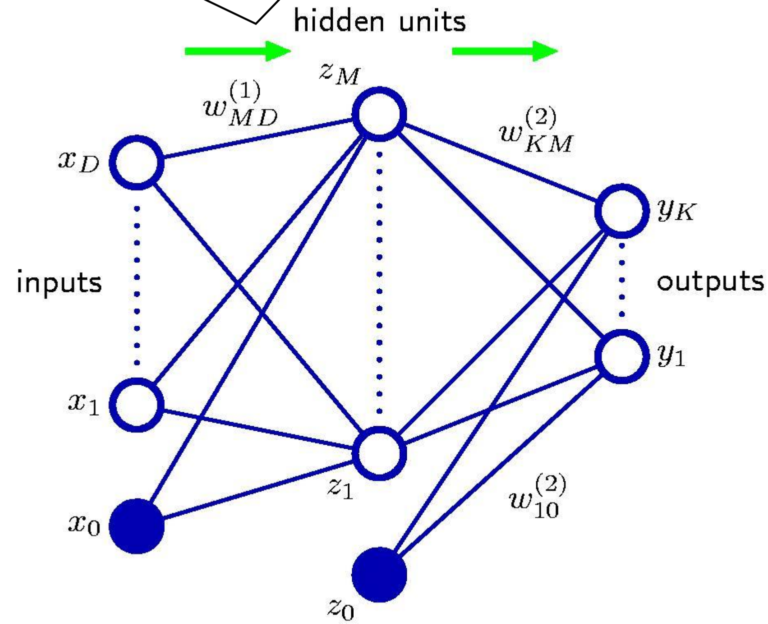
biases

Forward propagation of information

$$z_j = h(a_j)$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

$$y_k = \sigma(a_k)$$



Feed-forward Network Functions

- The overall network function, comprising two stage processing, becomes a linear regression model with adaptive basis functions

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} \underbrace{h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right)}_{\text{Adaptive basis functions}} + w_{k0}^{(2)}\right)$$

Bayesian Neural Networks I

$$p(t|\boldsymbol{x},\boldsymbol{w},\beta) = N(t|y(\boldsymbol{x},\boldsymbol{w}),\beta^{-1}) \qquad p(\boldsymbol{w}) = N(\boldsymbol{w}|0,\alpha^{-1}I) \qquad p(\boldsymbol{w}|t) \propto p(\boldsymbol{w})p(t|\boldsymbol{x},\boldsymbol{w},\beta)$$

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{w}|\boldsymbol{t}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\beta}{2}\sum_{n=1}^{N}[y_n(\boldsymbol{x},\boldsymbol{w}) - y_n]^2 + Constant$$

$$\nabla E(\boldsymbol{w}) = \alpha\boldsymbol{w} + \beta\sum_{n=1}^{N}(y_n - t_n)\boldsymbol{g}_n \qquad \boldsymbol{g} = \nabla_{\boldsymbol{w}} y(\boldsymbol{x},\boldsymbol{w})$$

$$\boldsymbol{A} = \nabla\nabla E(\boldsymbol{w}) = \alpha\mathbf{I} + \beta\boldsymbol{H}$$

$$\boldsymbol{w}_{\mathrm{MAP}} \longleftarrow \quad \boldsymbol{w}^{new} = \boldsymbol{w}^{old} - \boldsymbol{A}^{-1}\nabla E(\boldsymbol{w}) \qquad q(\boldsymbol{w}) = N(\boldsymbol{w}|\boldsymbol{w}_{MAP},\boldsymbol{A}^{-1})$$

Bayesian Neural Networks I

$$\mathrm{y}(\boldsymbol{x},\boldsymbol{w}) \simeq y(\boldsymbol{x},\boldsymbol{w}_{MAP}) + \boldsymbol{g}^T{}_{MAP}(\boldsymbol{w}-\boldsymbol{w}_{MAP})$$

$$p(t|\boldsymbol{x},\boldsymbol{w},\beta) = N(t|y(\boldsymbol{x},\boldsymbol{w}_{MAP}) + \boldsymbol{g}^T{}_{MAP}(\boldsymbol{w}-\boldsymbol{w}_{MAP}),\beta^{-1})$$

$$q(\boldsymbol{w}) = N(\boldsymbol{w}|\boldsymbol{w}_{MAP},\boldsymbol{A}^{-1})$$

$$p(t|\boldsymbol{x},D,\alpha,\beta) = \int p(t|\boldsymbol{x},\boldsymbol{w},\beta)q(\boldsymbol{w})d\boldsymbol{w}$$

$$p(t|\boldsymbol{x},D,\alpha,\beta) = N(t|y(\boldsymbol{x},\boldsymbol{w}_{MAP}),\boldsymbol{g}^T{}_{MAP}\boldsymbol{A}^{-1}\boldsymbol{g}_{MAP}+\beta^{-1})$$

Bayesian Neural Networks II

$p(\boldsymbol{w}) = N(\boldsymbol{w}|0, \alpha^{-1}I) \quad p(\boldsymbol{w}|t) \propto p(\boldsymbol{w})p(t|\boldsymbol{w})$

$$E(w) = -\ln p(w|t) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{n=1}^{N}[t_n \ln y_n + (1 - tn)\ln(1 - yn)]$$

$$\nabla E(w) = \alpha\boldsymbol{w} + \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{g}_n$$

$$\boldsymbol{A} = \nabla\nabla E(\boldsymbol{w}) = \alpha\boldsymbol{I}+\boldsymbol{H} \qquad \boldsymbol{H} = \sum_{n=1}^{N} y_{n}(1 - y_n)\boldsymbol{g}_n\boldsymbol{g}_n{}^T$$

$$\boldsymbol{w}_{\mathrm{MAP}} \longleftarrow \boldsymbol{w}^{new} = \boldsymbol{w}^{old} - \boldsymbol{A}^{-1}\nabla E(\boldsymbol{w}) \qquad q(\boldsymbol{w}) = N(\boldsymbol{w}|\boldsymbol{w}_{MAP}, A^{-1})$$

The Kullback-Leibler Divergence

$$\begin{aligned}
\mathrm{KL}(p\|q) &= \overbrace{-\int p(\mathbf{x}) \ln q(\mathbf{x})\, \mathrm{d}\mathbf{x}}^{\text{Cross Entropy } \mathrm{C}(p\|q)} - \left(\overbrace{-\int p(\mathbf{x}) \ln p(\mathbf{x})\, \mathrm{d}\mathbf{x}}^{\text{Entropy } \mathrm{H}(p)} \right) \\
&= -\int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} \mathrm{d}\mathbf{x}
\end{aligned}$$

$$\mathrm{KL}(p\|q) \simeq \frac{1}{N} \sum_{n=1}^{N} \{ \overbrace{-\ln q(\mathbf{x}_n|\boldsymbol{\theta})}^{\text{Cross Entropy}} + \overbrace{\ln p(\mathbf{x}_n)}^{\text{Negative Entropy}} \}$$

$$\mathrm{KL}(p\|q) \geqslant 0 \qquad\qquad \mathrm{KL}(p\|q) \not\equiv \mathrm{KL}(q\|p)$$

KL divergence describes a distance between model p and model q

Cross Entropy for Machine Learning

<u>Goal of Machine Learning</u>： $p(\,real\ data\,) \approx p(\,model \mid \theta\,)$

we assume： $p(\,training\ data\,) \approx p(\,real\ data\,)$

<u>Operation of Machine Learning</u>： $p(training\ data\,) \approx p(model \mid \theta\,)$

$$\boxed{\min_{\theta} \mathrm{KL}(p(\,training\ data\,) \,||\, p(model \mid \theta)\,)}$$

$$\Leftrightarrow \boxed{\min_{\theta} \mathrm{C}(p(\,training\ data\,) \,||\, p(model \mid \theta)\,)}$$

as $\mathrm{H}(p(\,training\ data\,)\,)$ is fixed

Cross Entropy for Machine Learning

$$\boxed{\mathrm{C}(p(\,training\ data\,)\,||\,p(model\,|\,\theta)\,)}$$

Bernoulli model: $p(\,model \mid \theta\,) = \rho^{t}(1-\rho)^{1-t}$

Cross entropy : $\mathrm{C} = -\frac{1}{N}\sum_n t_n \ln\rho + (1-t_n)\ln(1-\rho)$

t_n: *training data*

ρ: *model parameter*

Gaussian model: $p(\,model \mid \theta\,) \propto e^{-0.5(t-\mu)^2}$

Cross entropy : $\mathrm{C} \propto \frac{1}{N}\sum_n (t_n - \mu)^2$

t_n: *training data*

μ: *model parameter*

SVM v.s. Logistic Regression I

For data points on the correct side, $\xi = 0$

For the remaining points, $\xi = 1 - y_n t_n$

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2 \Rightarrow \sum_{n=1}^{N} E_{SV}(y_n, t_n) + \lambda\|\mathbf{w}\|^2$$

where $\lambda = (2C)^{-1}$

$E_{SV}(y_n, t_n) = [1 - y_n t_n]_+$ **: hinge error function**

where $[\cdot]_+$ denotes the positive part

SVM v.s. Logistic Regression II

- From maximum likelihood logistic regression

$$p(t=1\,|\,y)=\sigma(y)$$

$$p(t=-1\,|\,y)=1-\sigma(y)=\sigma(-y)$$

$$\Rightarrow\; p(t\,|\,y)=\sigma(yt)$$

- Error function with quadratic regularization

$$\sum_{n=1}^{N} E_{LR}(y_n t_n)+\lambda\|\mathbf{w}\|^2$$

where $E_{LR}(yt)=\ln\left(1+\exp\left(-yt\right)\right)$

SVM v.s. Logistic Regression III

y: Bernoulli parameter

a: natural parameter

Cross-Entropy

$$-\ln p(t|y) = -t \ln y - (1-t)\ln(1-y)$$

$$-\ln p(t|a) = -\ln \sigma(at) = \ln(1+e^{-at}) \qquad y = \sigma(a)$$

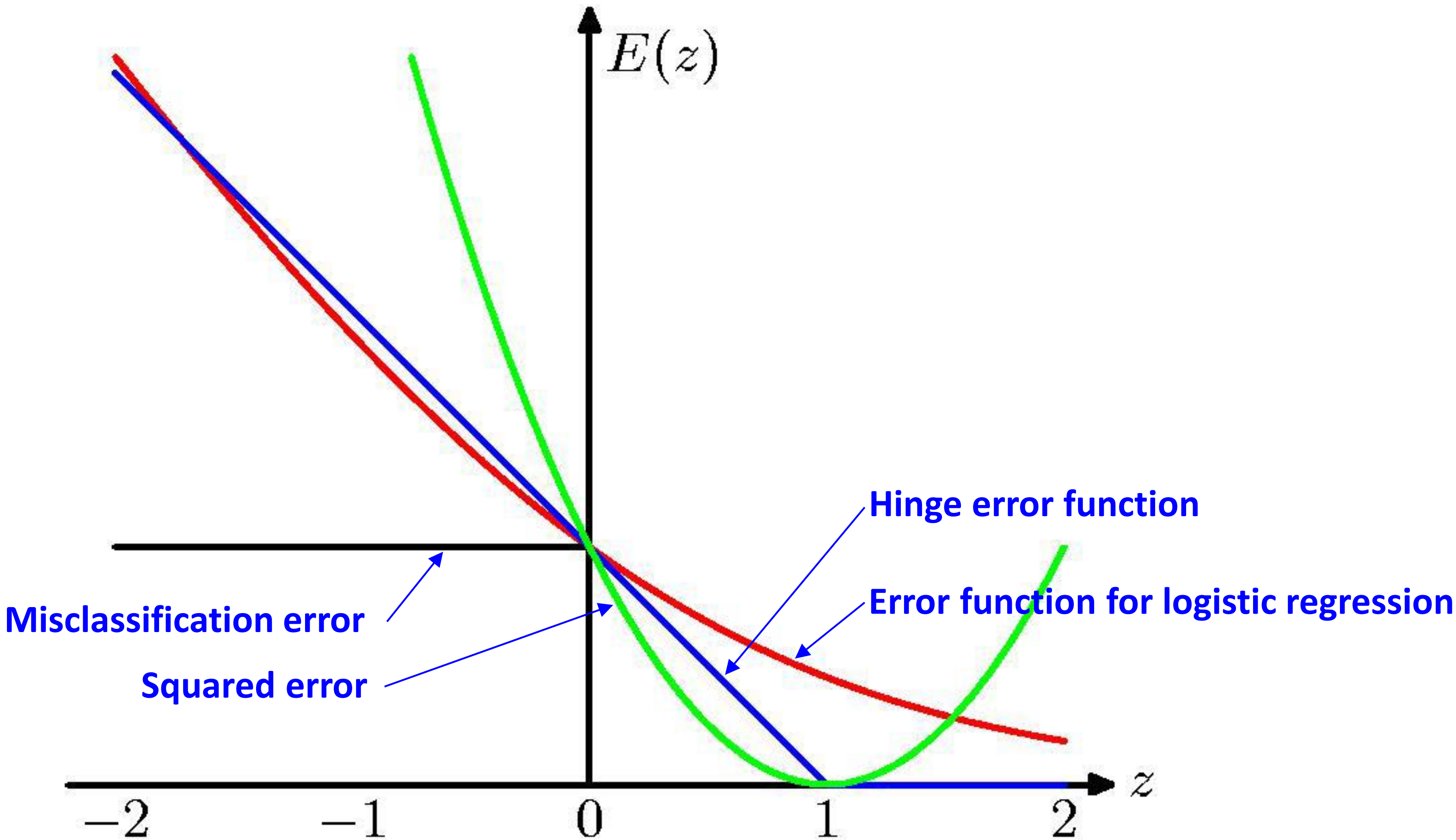
Cross-Entropy with prior

$$-\ln p(t|y) + \alpha^{-1}\mathbf{w}^T\mathbf{w} = -t \ln y - (1-t)\ln(1-y) + \alpha^{-1}\mathbf{w}^T\mathbf{w}$$

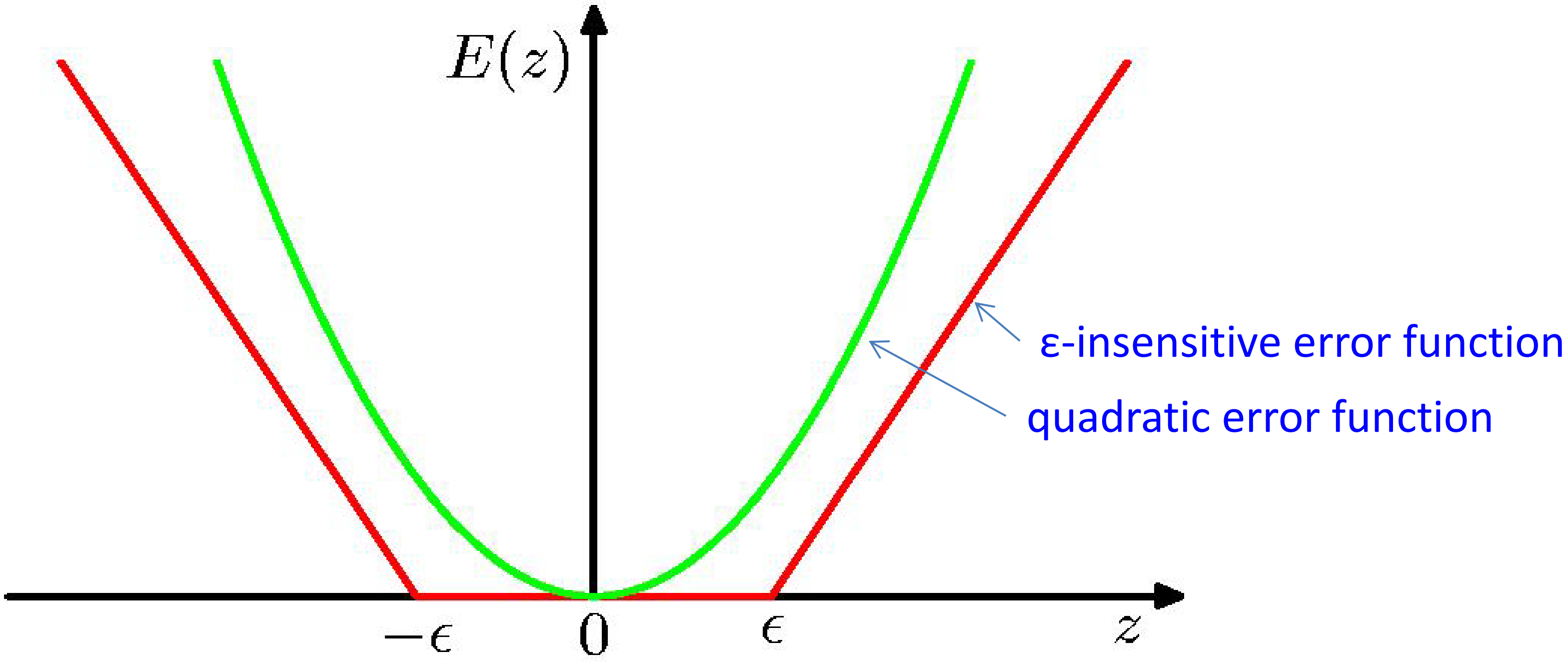
$$-\ln p(t|a) + \alpha^{-1}\mathbf{w}^T\mathbf{w} = \ln(1+e^{-at}) + \alpha^{-1}\mathbf{w}^T\mathbf{w}$$

softplus: $\boxed{\ln(1+e^{-x})}$

Comparison of Error Functions



Comparison of Error Functions



Reduction of Dimensionality (PCA)

Basis

Data

Coefficients

$$\boldsymbol{Y} = \boldsymbol{AX}$$

principal component analysis

$$\boxed{\min_{A_i} {A_i}^T COV(Y_i) A_i}$$

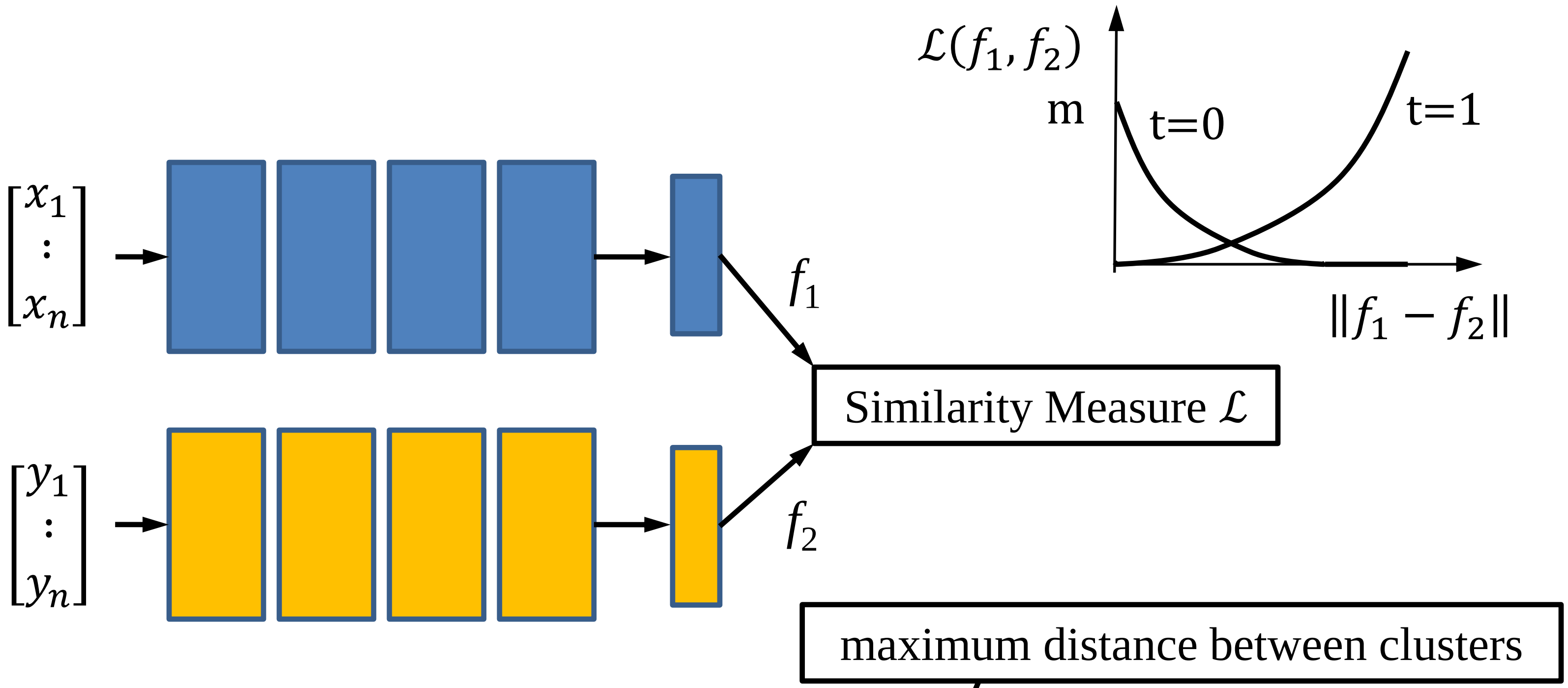
A: rotation

$${A_i^*}^T COV(Y_i) A_i^* = \lambda_i$$

A_i^*: optimal solution

$$s.t. \qquad {A_i}^T A_i = 1 \qquad E[Y_i] = \mathbf{0}$$

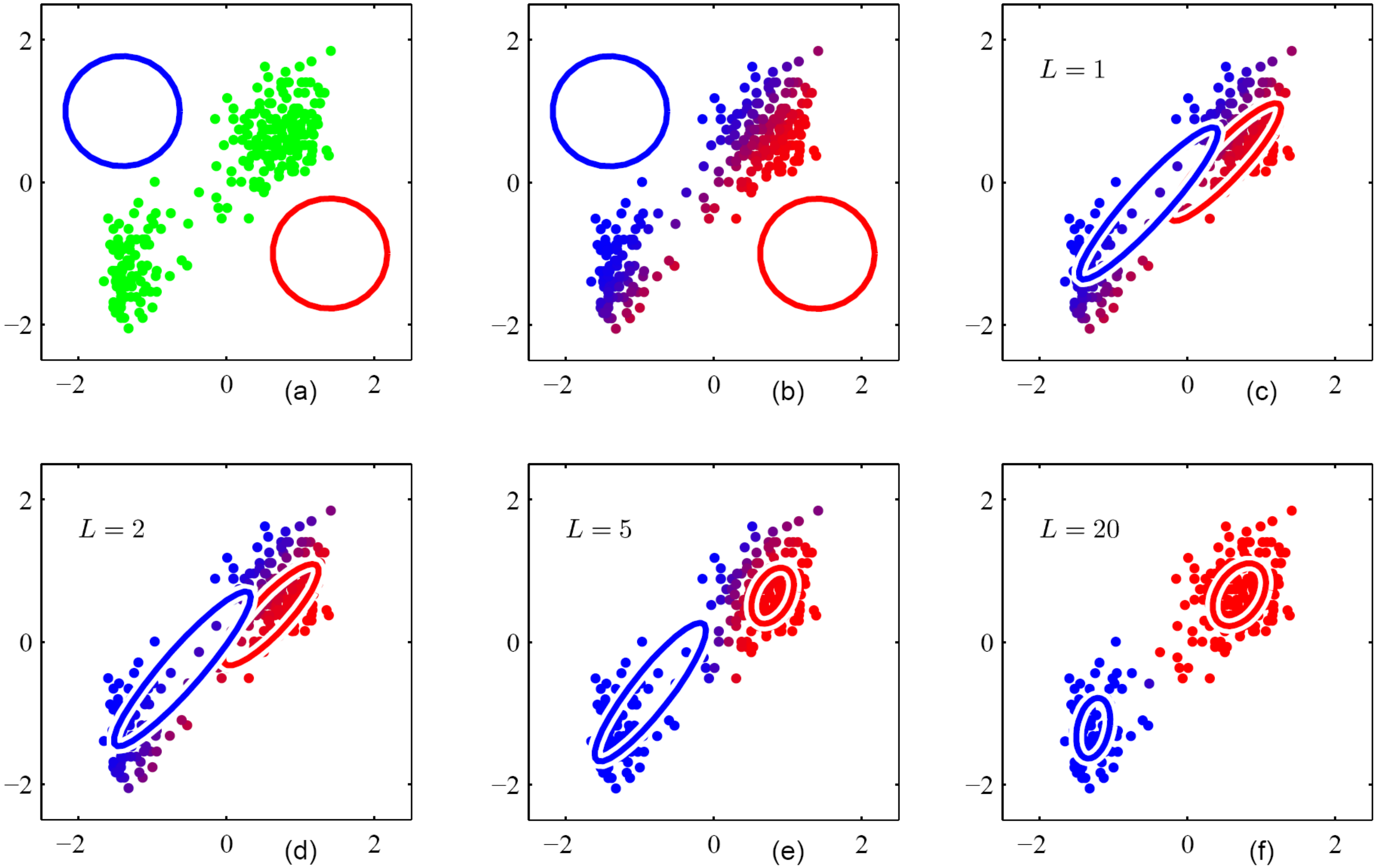
Feature Extraction (Contrastive Loss)



$$\mathcal{L}(f_1, f_2) = t\|f_1 - f_2\|^2 + (1 - t)[m - \|f_1 - f_2\|^2]_+$$

t =1: two vectors belong to the same category; [ ]$_+$: non-negative

EM for Gaussian Mixtures



EM for Gaussian Mixtures

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters
  1. Initialize the means μ_k, covariance Σ_k and mixing coefficients π_k
  2. E step

$$\gamma(z_{nk}) = \frac{\pi_k N(\boldsymbol{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x}_n | \mu_j, \Sigma_j)}$$

  3. M step

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\boldsymbol{x}_n - \mu_k^{new})(\boldsymbol{x}_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

  4. Evaluate the log likelihood

$$\ln p(\boldsymbol{X} | \mu, \Sigma, \Pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}_n | \mu_k, \Sigma_k) \right\}$$

EM for Bernoulli Mixtures

$$\ln p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{K} \pi_k p(\mathbf{x_n} \mid \boldsymbol{\mu}_k)\right\}$$

$$p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\mu}) = \prod_{k=1}^{K} p(\mathbf{x} \mid \boldsymbol{\mu}_\mathbf{k})^{z_k} \qquad (\mathbf{z} = (z_1, \cdots, z_K)^{\mathrm{T}} \text{ is a binary indicator variables})$$

$$p(\mathbf{z} \mid \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{\,z_k}$$

(complete - data log likelihood function) :

$$\ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left\{\ln \pi_k + \sum_{i=1}^{D} [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})]\right\}$$

$$\mathrm{E}_{\mathrm{Z}}[\ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\mu}, \boldsymbol{\pi})] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \left\{\ln \pi_k + \sum_{i=1}^{D} [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})]\right\}$$

$$(\text{E - step}) \quad \gamma(z_{nk}) = E[z_{nk}] = \frac{\pi_k p(\mathbf{x_n} \mid \boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \pi_j p(\mathbf{x_n} \mid \boldsymbol{\mu}_j)}, \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk}), \qquad \bar{\mathbf{x}}_\mathbf{k} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x_n}$$

$$(\text{M - step}) \qquad \boldsymbol{\mu}_k = \bar{\mathbf{x}}_\mathbf{k}, \qquad \pi_\mathrm{k} = \frac{N_k}{N}$$

* In contrast to the mixture of Gaussians, there are no singularities in which the likelihood goes to infinity

Hidden Markov Models

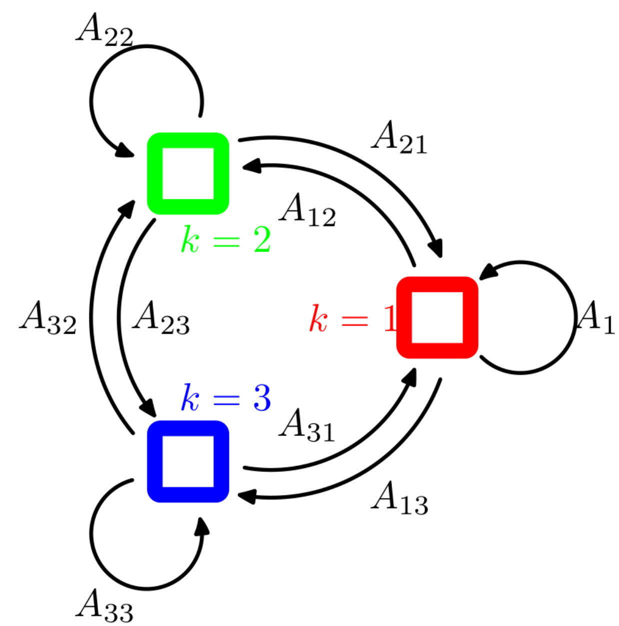
Conditional distribution for latent variable

$$p(\mathbf{z}_n|\mathbf{z}_{n-1,}\mathbf{A}) = \prod_{k=1}^{K}\prod_{j=1}^{K} A_{jk}^{z_{n-1,j}z_{nk}}$$

$$p(\mathbf{z}_1|\boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$$

$$\sum_k \pi_k = 1$$

A means transition probabilities



As in the case of a standard mixture model, the latent variables are the discrete multinomial variables zn using 1-of-K coding scheme

a model whose latent variables have three possible states corresponding to the three boxes. The black lines denote the elements of the transition matrix Ajk

EM for HMM

EM algorithm

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$$

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \sum \gamma(\mathbf{z}) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = \mathbb{E}[z_{n-1,j} z_{nk}] = \sum_{\mathbf{z}} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}$$

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

Sum-Product v.s. Max-Product

- Sum-Product Algorithm (evaluation)
  - ✓ Compute the joint distribution from the Product
  - ✓ Infer marginal distributions from the Sum

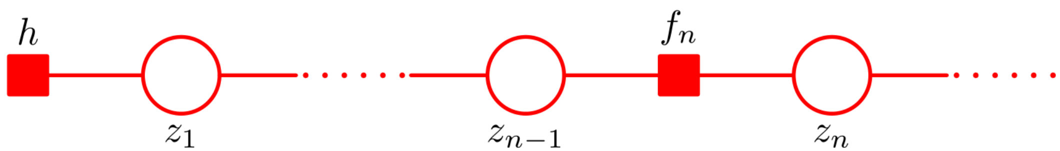
$$p(x_1, x_2) = \sum_{x_3} p(x_1, x_3) p(x_2, x_3)$$

- Max-Product Algorithm (decoding)
  - ✓ Compute the joint distribution from the Product
  - ✓ Perform ML estimation from the Max

$$x_1^* = \max_{x_1} p(x_1, x_3) p(x_2, x_1)$$

Sum-Product for HMMs

- Transforming the directed graph into a factor graph



$$
\begin{aligned}
h(\mathbf{z}_1) &= p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) \\
f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)
\end{aligned}
$$

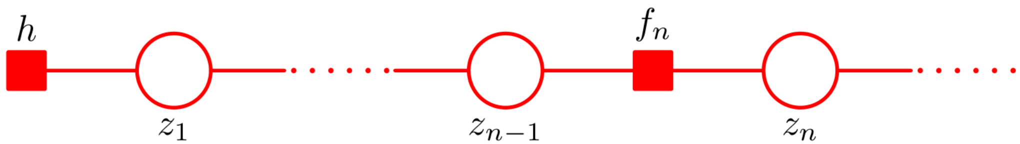
$$
\begin{aligned}
\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1}) &= \mu_{f_{n-1}\to \mathbf{z}_{n-1}}(\mathbf{z}_{n-1}) \\
\mu_{f_n\to \mathbf{z}_n}(\mathbf{z}_n) &= \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1}, \mathbf{z}_n)\mu_{\mathbf{z}_{n-1}\to f_n}(\mathbf{z}_{n-1})
\end{aligned}
$$

$$\boxed{\mu_{f_n\to \mathbf{z}_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1}, \mathbf{z}_n)\mu_{f_{n-1}\to \mathbf{z}_{n-1}}(\mathbf{z}_{n-1})}$$

$$\boxed{\alpha(\mathbf{z}_n) = \mu_{f_n\to \mathbf{z}_n}(\mathbf{z}_n)}$$

Sum-Product for HMMs

- Transforming the directed graph into a factor graph



$$\mu_{f_{n+1}\rightarrow f_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1})\mu_{f_{n+2}\rightarrow f_{n+1}}(\mathbf{z}_{n+1})$$

$$\boxed{\beta(\mathbf{z}_n) = \mu_{f_{n+1}\rightarrow \mathbf{z}_n}(\mathbf{z}_n)}$$

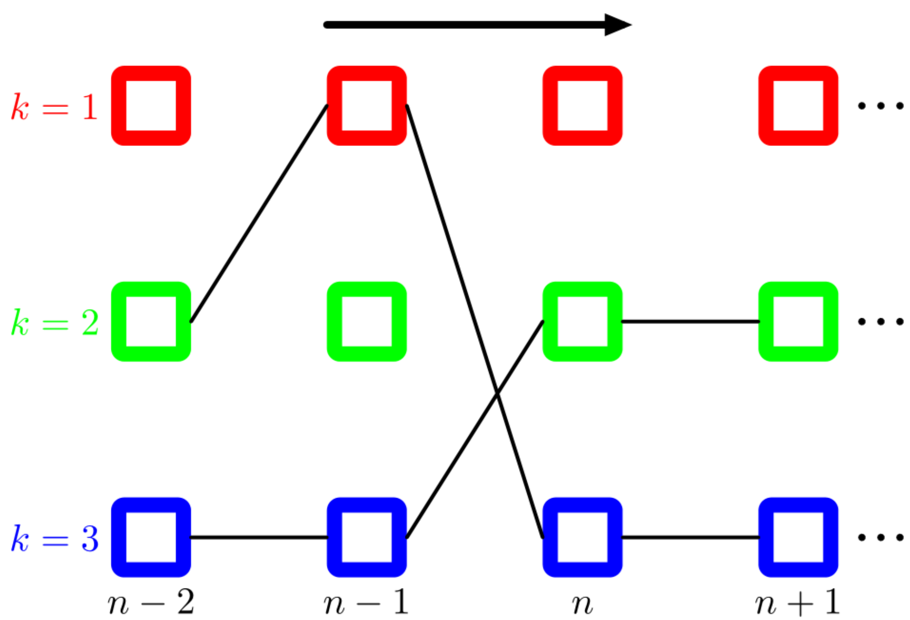
$$\boxed{p(\mathbf{z}_n, \mathbf{X}) = \mu_{f_n\rightarrow \mathbf{z}_n}(\mathbf{z}_n)\mu_{f_{n+1}\rightarrow \mathbf{z}_n}(\mathbf{z}_n) = \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}$$

$$\boxed{\gamma(\mathbf{z}_n) = \frac{p(\mathbf{z}_n, \mathbf{X})}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}}$$

Latent Sequence Estimation

- **Decoding**:

  Given a HMM model $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$, what is the most likely latent sequence $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ for an observation sequence $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$?



Viterbi Algorithm

$$\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}} \ln p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{z}_1,\ldots,\mathbf{z}_n)$$

$$\omega(\mathbf{z}_{n+1}) = \max_{\mathbf{z}_1,\ldots,\mathbf{z}_n} \ln p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{x}_{n+1},\mathbf{z}_1,\ldots,\mathbf{z}_n,\mathbf{z}_{n+1})$$

$$\omega(\mathbf{z}_{n+1}) = \ln p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) + \max_{\mathbf{z}_n}\{\ln p(\mathbf{z}_{n+1}|\mathbf{z}_n) + \omega(\mathbf{z}_n)\}$$

$$\omega(\mathbf{z}_1) = \ln p(\mathbf{z}_1) + \ln p(\mathbf{x}_1|\mathbf{z}_1)$$

Viterbi Algorithm

- Note that maximization over $\mathbf{z}_n$ must be performed for each of K possible values of $\mathbf{z}_{n+1}$
- Denote this function by $\psi(k_n)$, where $k \in \{1, \ldots, K\}$
- Once we find the most probable value of $\mathbf{z}_N$, we can trackback along the chain

$$k_n^{\max} = \psi(k_{n+1}^{\max})$$

- Reduce the computational cost from O(K^N) to O(KN)

Example

- Given an HMM and an observation sequence, how to perform evaluation and decoding

Transition A	Emission B	Hidden States Z	Observations X
$\begin{bmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{bmatrix}$	$\begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix}$	{bull, bear}	{up, down}

If Z is stationary, then π = [3/7, 4/7]. We also can assume π = [1/2, 1/2]

An observation sequence: {up, up, down}

Evaluation (Sum-Product)

$$\alpha(z_1) = p(z_1, x_1) = p(x_1|z_1)p(z_1)$$

$\boxed{x_1\text{=up, } z_1\text{=bull or bear}}$

$$= \begin{bmatrix}0.8\\0.1\end{bmatrix} . \begin{bmatrix}0.5\\0.5\end{bmatrix} = \begin{bmatrix}0.8 \times 0.5\\0.1 \times 0.5\end{bmatrix} = \begin{bmatrix}0.4\\0.05\end{bmatrix}$$

$$\alpha(z_2) = p(z_2, x_1, x_2) = p(x_2|z_2) \sum_{z_1} p(z_2|z_1)\,\alpha(z_1)$$

$\boxed{x_2\text{=up, } z_2\text{=bull or bear}}$

$$= \begin{bmatrix}0.8\\0.1\end{bmatrix} . \begin{bmatrix}0.6 \times \boxed{0.4} + 0.3 \times \boxed{0.05}\\0.4 \times \boxed{0.4} + 0.7 \times \boxed{0.05}\end{bmatrix} = \begin{bmatrix}0.204\\0.0195\end{bmatrix}$$

$$\alpha(z_3) = p(z_3, x_1, x_2, x_3) = p(x_3|z_3) \sum_{z_2} p(z_3|z_2)\,\alpha(z_2)$$

$\boxed{x_3\text{=down, } z_2\text{=bull or bear}}$

$$= \begin{bmatrix}0.2\\0.9\end{bmatrix} . \begin{bmatrix}0.6 \times \boxed{0.204} + \boxed{0.3 \times 0.0195}\\0.4 \times \boxed{0.204} + \boxed{0.7 \times 0.0195}\end{bmatrix} = \begin{bmatrix}0.02565\\0.085725\end{bmatrix}$$

$$p(x_1, x_2, x_3) = \sum_{z_3} \alpha(z_3) = 0.111375$$

Decoding (Max-Product)

$$\delta(z_1) = p(z_1, x_1) = p(x_1|z_1)p(z_1)$$

$\boxed{x_1\text{=up, } z_1\text{=bull or bear}}$

$$= \begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.8 \times 0.5 \\ 0.1 \times 0.5 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.05 \end{bmatrix}$$

$$\delta(z_2) = p(z_2, x_1, x_2) = p(x_2|z_2) \max_{z_1} p(z_2|z_1)\delta(z_1)$$

$\boxed{x_2\text{=up, } z_2\text{=bull or bear}}$

$$= \begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \times 0.4 \\ 0.4 \times 0.4 \end{bmatrix} = \begin{bmatrix} 0.192 \\ 0.016 \end{bmatrix} \leftarrow$$

$$\phi(z_2) = \arg \max_{z_1} p(z_2|z_1)\delta(z_1) = \begin{bmatrix} bull \rightarrow bull \\ bull \rightarrow bear \end{bmatrix} \leftarrow$$

$$\delta(z_3) = p(z_3, x_1, x_2, x_3) = p(x_3|z_3) \max_{z_2} p(z_3|z_2)\delta(z_2)$$

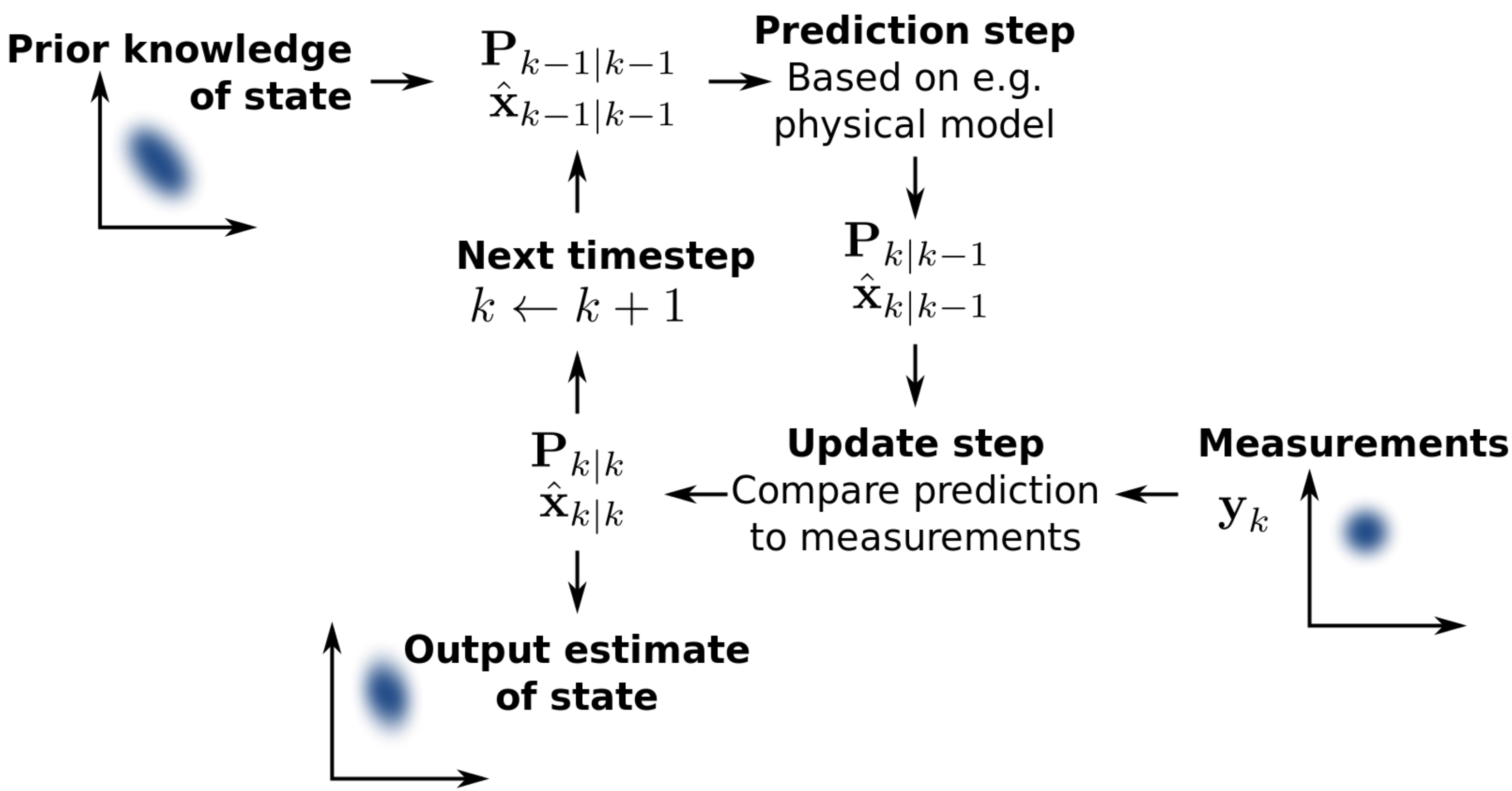
$\boxed{x_3\text{=down, } z_2\text{=bull or bear}}$

$$= \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \times 0.192 \\ 0.4 \times 0.192 \end{bmatrix} = \begin{bmatrix} 0.02304 \\ 0.06912 \end{bmatrix} \leftarrow$$

$$\phi(z_3) = \arg \max_{z_2} p(z_3|z_2)\delta(z_2) = \begin{bmatrix} bull \rightarrow bull \\ bull \rightarrow bear \end{bmatrix} \leftarrow$$

Optimal solution：
bull→bull → bear

Kalman Filtering

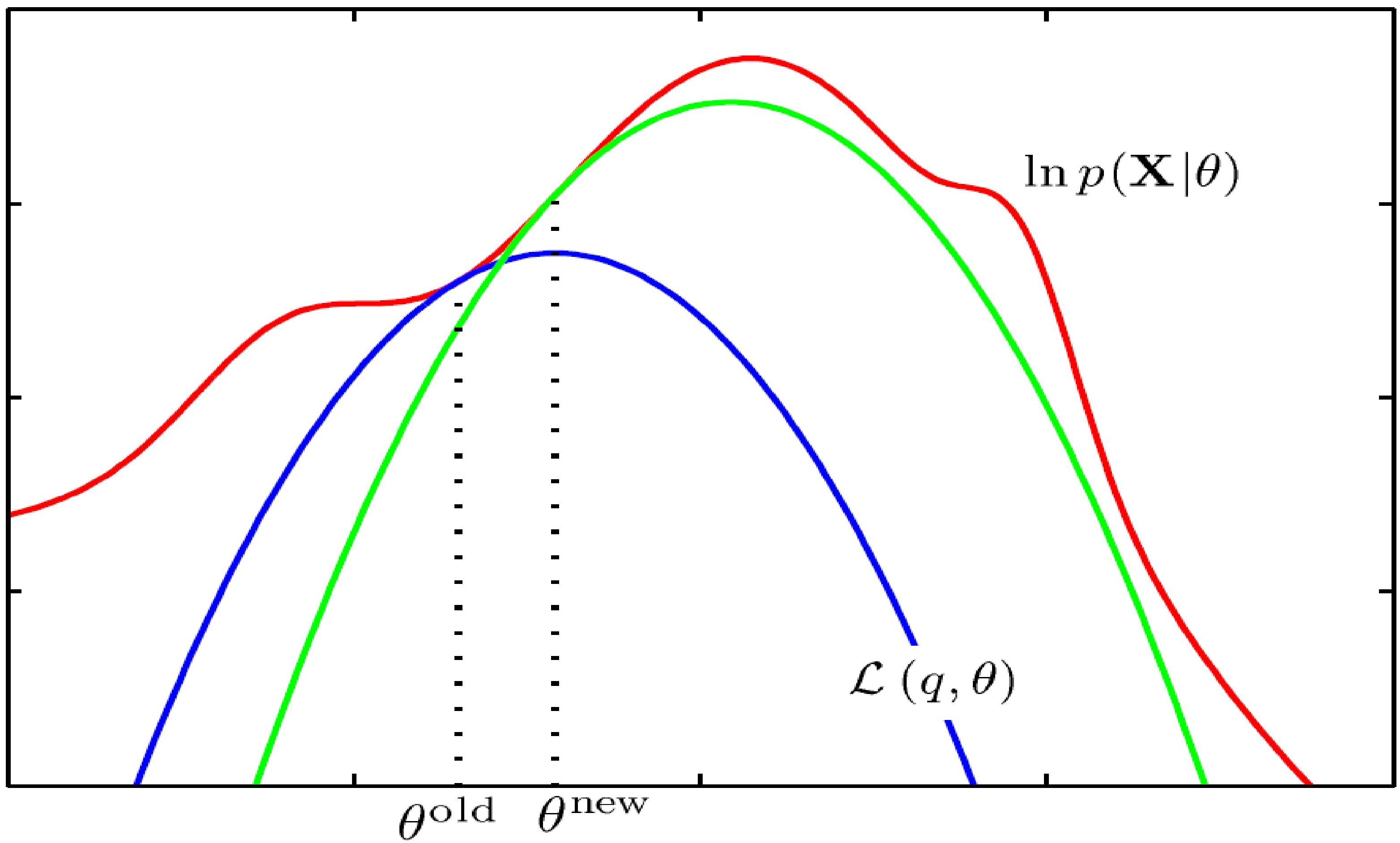


EM for LDS

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \ln p(\mathbf{z}_1|\boldsymbol{\mu}_0, \mathbf{V}_0) + \sum_{n=2}^{N} \ln p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}, \boldsymbol{\Gamma}) + \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{C}, \boldsymbol{\Sigma})$$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\mathbf{Z}|\boldsymbol{\theta}^{\text{old}}} [\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})]$$

EM Algorithm in General



The EM Algorithm in General (I)

- Direct optimization of p (X|θ) is difficult while optimization of complete data likelihood p (X, Z|θ) is significantly easier.
- Decomposition of the likelihood p (X|θ)

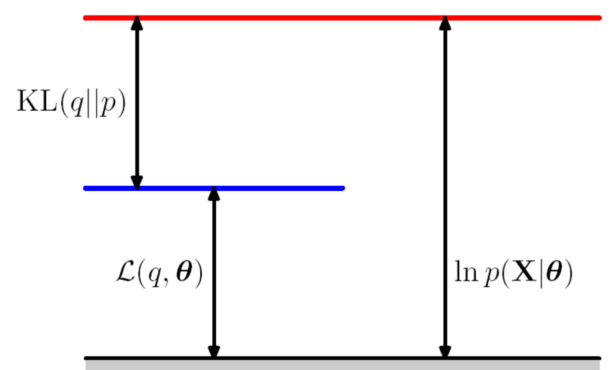
$$\boxed{\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X}|\boldsymbol{\theta})}$$

$$\boxed{\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p)}$$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

$$\mathrm{KL}(q\|p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$$

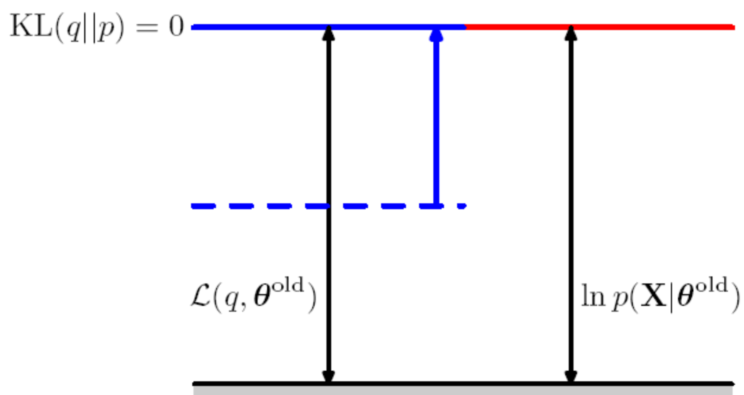
$$\mathrm{KL}(q\|p) \geqslant 0 \quad \Rightarrow \quad \mathcal{L}(q, \boldsymbol{\theta}) \leqslant \ln p(\mathbf{X}|\boldsymbol{\theta})$$

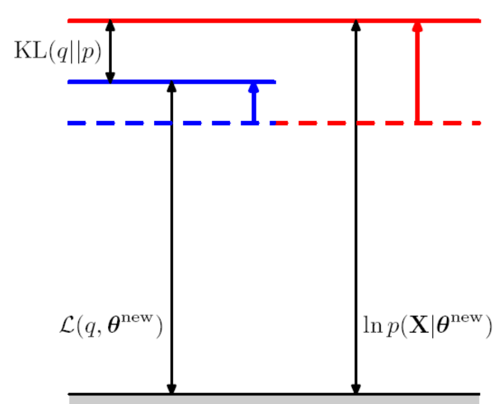


The EM Algorithm in General (II)

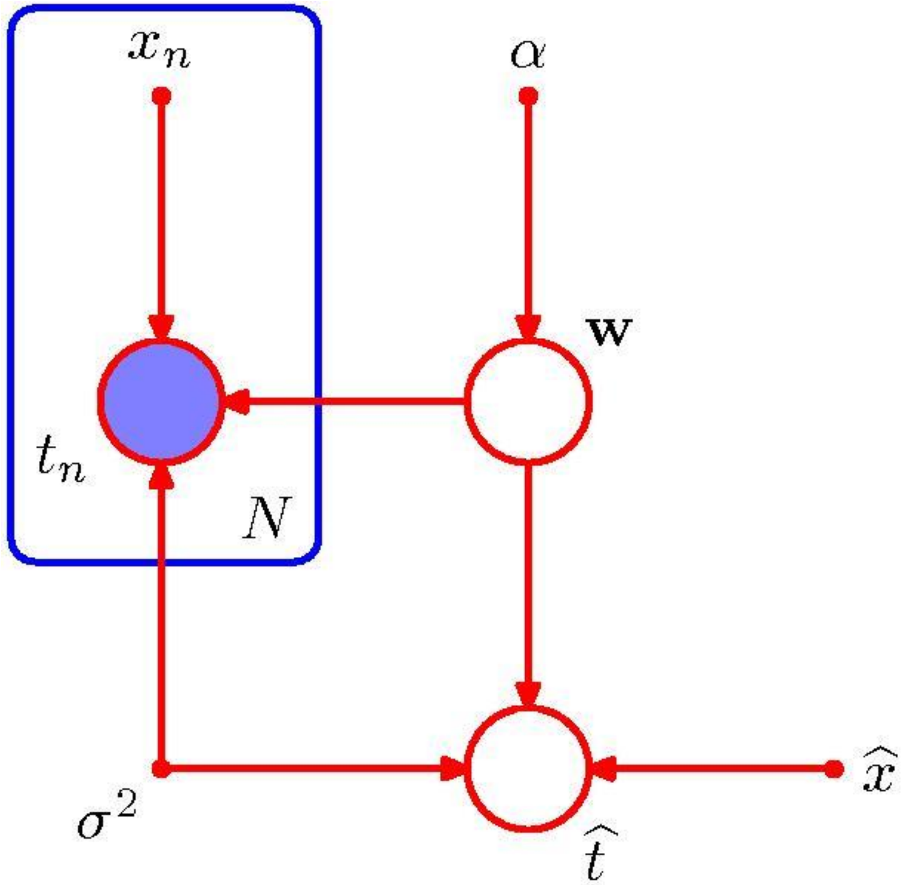
- (**E step**) The lower bound $\mathcal{L}(q, \theta_{old})$ is maximized while holding θ_{old} fixed. Since $\ln p(X|\theta)$ does not depend on $q(Z)$, $\mathcal{L}(q, \theta_{old})$ will be the largest when $KL(q||p)$ vanishes (i.e. when $q(Z)$ is equal to the posterior distribution $p(Z|X, \theta_{old})$)
- (**M step**) $q(Z)$ is fixed and the lower bound $\mathcal{L}(q, \theta_{old})$ is maximized wrt. θ to θ_{new}. When the lower bound is increased, θ is updated making $KL(q||p)$ greater than 0. Thus the increase in the log likelihood function is greater than the increase in the lower bound.
- In the M step, the quantity being maximized is the expectation of the complete-data log-likelihood

$$\begin{aligned}\mathcal{L}(q,\boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}}) \\ &= \mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}}) + \text{const} \quad \Leftarrow \quad \boxed{q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}})}\end{aligned}$$

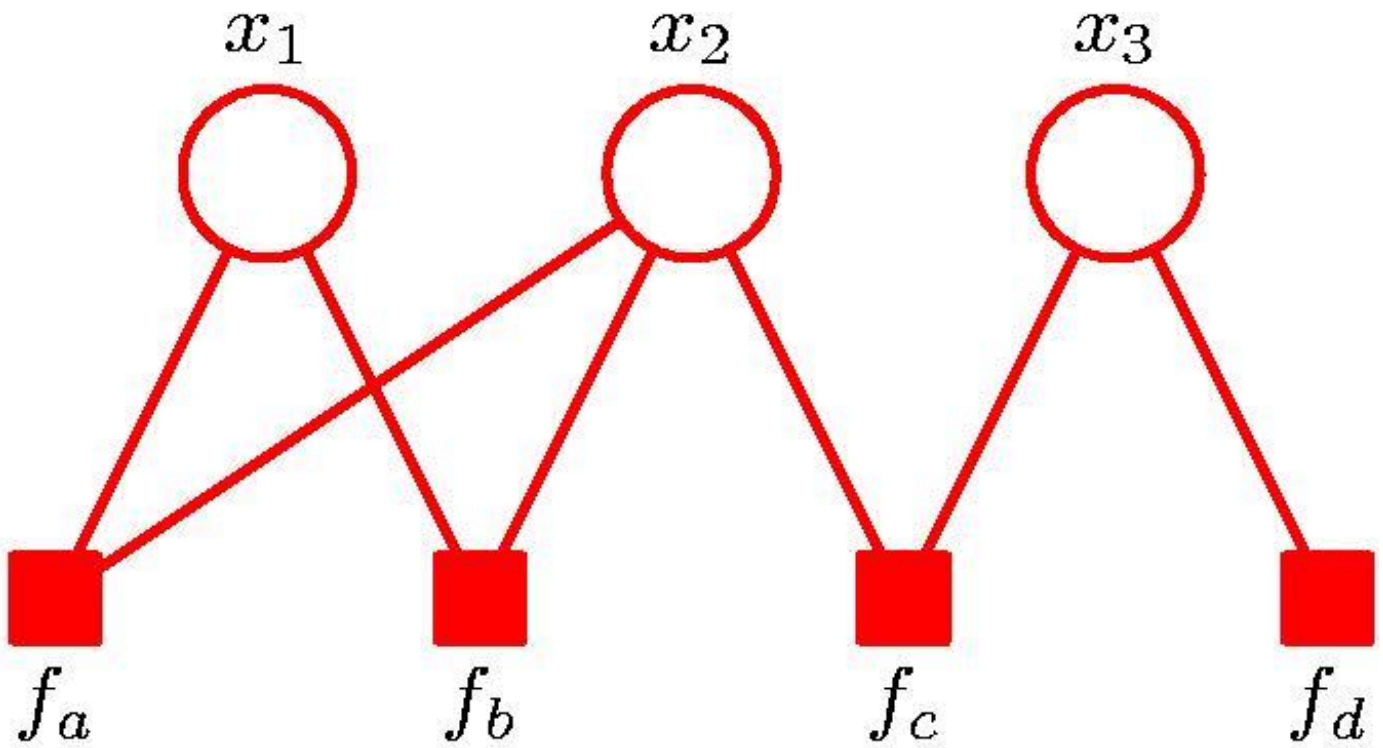




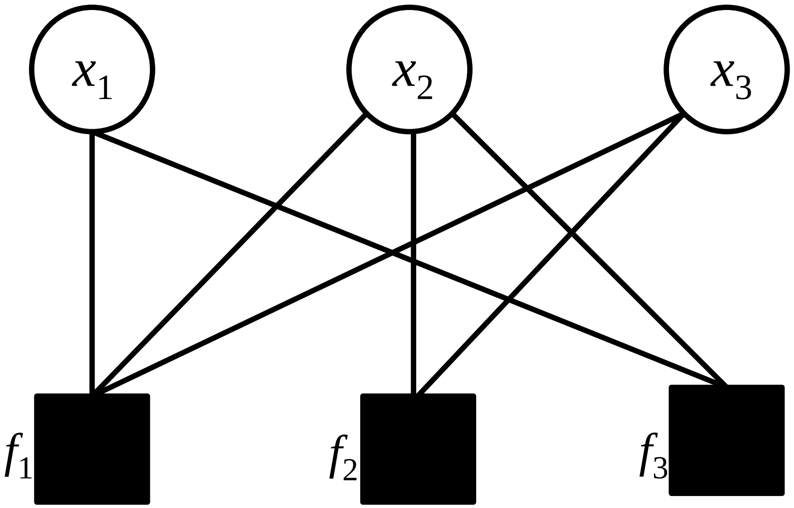
Graphical Model – Bayesian Regression



Graphical Model – Factor Graph



Factor Graph for Solving Equations



$x_1 + 2x_2 + x_3 = 4$ $\quad x_2 + 2x_3 = 3$ $\quad x_1 + x_2 = 2$

(1) $x_1 = 1 \quad x_2 = 1 \quad x_3 = 0$

(2) $f_1 \to x_1$: $x_1 = 4 - 2x_2 - x_3 = 2$

$f_1 \to x_2$: $x_2 = (4 - x_1 - x_3)/2 = 1.5$

$f_1 \to x_3$: $x_3 = 4 - 2x_2 - x_1 = 1$

$f_2 \to x_2$: $x_2 = 3 - 2x_3 = 3$

$f_2 \to x_3$: $x_3 = 3 - 2x_2 = 1$

$f_3 \to x_1$: $x_1 = 2 - x_2 = 1$

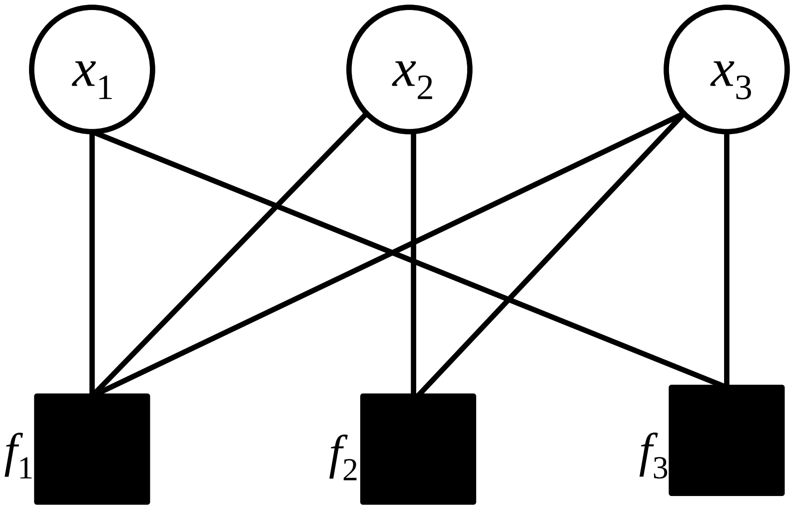
$f_3 \to x_2$: $x_2 = 2 - x_1 = 1$

(3) $x_1 = (1+2+1)/3 = 4/3 \quad x_2 = (1+1.5+3+1)/4 = 6.5/4 \quad x_3 = (0+1+1)/3 = 2/3$

Factor Graph for Computing Means

$S_i = x_i N_\mathrm{i}$

(1) $S_1 = 11 \quad N_1 = 10 \qquad S_2 = 10 \quad N_2 = 10 \qquad S_3 = 18 \quad N_3 = 20$



$\mathrm{x}_1 = x_2 = x_3 \qquad x_2 = x_3 \qquad x_1 = x_3$

(2)

$f_1 \rightarrow x_1$: $S_1 = 28 \quad N_1 = 30$

$f_1 \rightarrow x_2$: $S_2 = 29 \quad N_2 = 30$

$f_1 \rightarrow x_3$: $S_3 = 21 \quad N_3 = 20$

$f_2 \rightarrow x_2$: $S_2 = 18 \quad N_2 = 20$

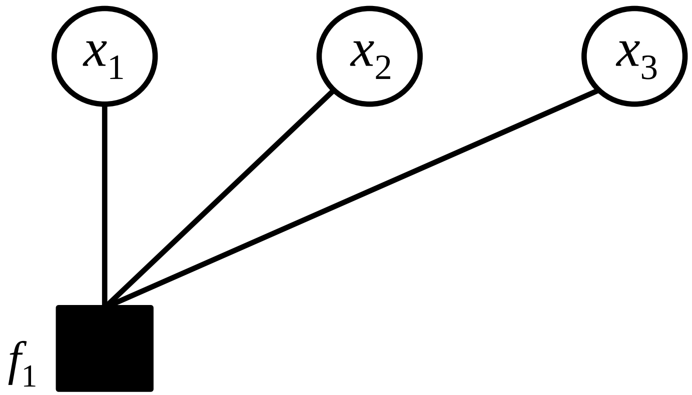
$f_2 \rightarrow x_3$: $S_3 = 10 \quad N_3 = 10$

$f_3 \rightarrow x_1$: $\mathrm{S}_1 = 18 \quad N_1 = 20$

$f_3 \rightarrow x_3$: $\mathrm{S}_3 = 11 \quad N_3 = 10$

(3) $S_1 = 57 \quad N_1 = 60 \qquad S_2 = 57 \quad N_2 = 60 \qquad S_3 = 60 \quad N_3 = 60$

Factor Graph for Belief Aggregation



$x_1 = x_2 = x_3$

(1) $x_1 \sim \mathcal{N}(m_1, \Sigma_1)$ $x_2 \sim \mathcal{N}(m_2, \Sigma_2)$

$x_3 \sim \mathcal{N}(m_3, \Sigma_3)$

(2) $f_1 \rightarrow x_1$:

$\hat{\Sigma}_1^{-1} = \Sigma_2^{-1} + \Sigma_3^{-1}$ $\hat{\Sigma}_1^{-1}\hat{m}_1 = \Sigma_2^{-1}m_2 + \Sigma_3^{-1}m_3$

$f_1 \rightarrow x_2$:

$\hat{\Sigma}_2^{-1} = \Sigma_1^{-1} + \Sigma_3^{-1}$ $\hat{\Sigma}_2^{-1}\hat{m}_2 = \Sigma_1^{-1}m_1 + \Sigma_3^{-1}m_3$

$f_1 \rightarrow x_3$:

$\hat{\Sigma}_3^{-1} = \Sigma_1^{-1} + \Sigma_3^{-1}$ $\hat{\Sigma}_2^{-1}\hat{m}_2 = \Sigma_1^{-1}m_1 + \Sigma_3^{-1}m_3$

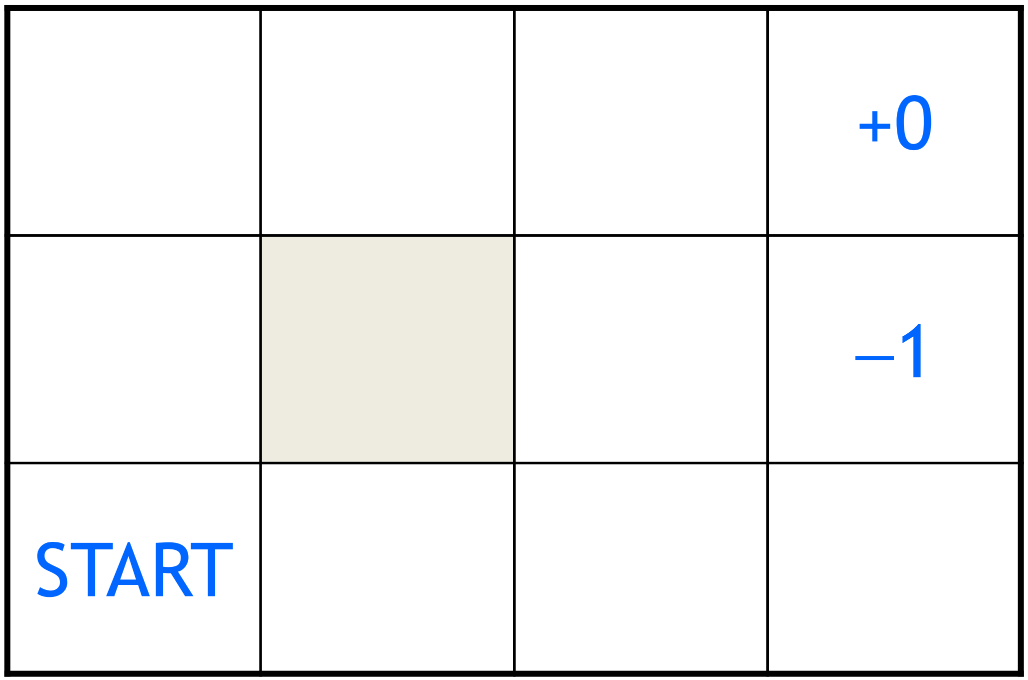
(3)

$\bar{\Sigma}_1^{-1} = \Sigma_1^{-1} + \hat{\Sigma}_1^{-1}$ $\bar{\Sigma}_1^{-1}\bar{m}_1 = \Sigma_1^{-1}m_1 + \hat{\Sigma}_1^{-1}\hat{m}_1$

$\bar{\Sigma}_2^{-1} = \Sigma_2^{-1} + \hat{\Sigma}_2^{-1}$ $\bar{\Sigma}_2^{-1}\bar{m}_2 = \Sigma_2^{-1}m_2 + \hat{\Sigma}_2^{-1}\hat{m}_2$

$\bar{\Sigma}_3^{-1} = \Sigma_3^{-1} + \hat{\Sigma}_3^{-1}$ $\bar{\Sigma}_3^{-1}\bar{m}_3 = \Sigma_3^{-1}m_3 + \hat{\Sigma}_3^{-1}\hat{m}_3$

Markov Decision Process



MDP Model

0	1	2	3
4	5	6	7
8	9	10	11

a) Position 3: reward = 0 for any action
b) Positions 5 and 7: wall, reward = -1
c) reward = - 0.1 for each step in other states
d) action = {up/0, down/1, left/2, right/3}

<u>transition probabilities:</u>

$$\{x\text{: }\{u_1\text{: }(x', p(x'|x, u_1), r), u_2\text{: }(x', p(x'|x, u_2), r),$$
$$u_3\text{: }(x', p(x'|x, u_3), r), u_4\text{: }(x', p(x'|x, u_4), r)\ \}\}$$

```
{0: {0: (0, 1.0, -0.1), 1: (4, 1.0, -0.1), 3: (1, 1.0, -0.1), 2: (0, 1.0, -0.1)}, 1: {0: (1,
1.0, -0.1), 1: (1, 1.0, -1), 3: (2, 1.0, -0.1), 2: (0, 1.0, -0.1)}, 2: {0: (2, 1.0, -0.1), 1:
(6, 1.0, -0.1), 3: (3, 1.0, -0.1), 2: (1, 1.0, -0.1)}, 3: {0: (3, 1.0, 0), 1: (3, 1.0, 0), 3:
(3, 1.0, 0), 2: (3, 1.0, 0)}, 4: {0: (0, 1.0, -0.1), 1: (8, 1.0, -0.1), 3: (4, 1.0, -1), 2: (4,
1.0, -0.1)}, 5: {0: (1, 1.0, -0.1), 1: (9, 1.0, -0.1), 3: (6, 1.0, -0.1), 2: (4, 1.0, -0.1)}, 6:
{0: (2, 1.0, -0.1), 1: (10, 1.0, -0.1), 3: (6, 1.0, -1), 2: (6, 1.0, -1)}, 7: {0: (3, 1.0,
-0.1), 1: (11, 1.0, -0.1), 3: (7, 1.0, -1), 2: (6, 1.0, -0.1)}, 8: {0: (4, 1.0, -0.1), 1: (8,
1.0, -0.1), 3: (9, 1.0, -0.1), 2: (8, 1.0, -0.1)}, 9: {0: (9, 1.0, -1), 1: (9, 1.0, -0.1), 3:
(10, 1.0, -0.1), 2: (8, 1.0, -0.1)}, 10: {0: (6, 1.0, -0.1), 1: (10, 1.0, -0.1), 3: (11, 1.0,
-0.1), 2: (9, 1.0, -0.1)}, 11: {0: (11, 1.0, -1), 1: (11, 1.0, -0.1), 3: (11, 1.0, -0.1), 2:
(10, 1.0, -0.1)}}
```

Value Iteration (I)

Value Function V^0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$\boxed{V^0(0) = 0.0}$ $\boxed{V^1(0) = -0.1}$

$r(0, up) + V^0(0)*p(0|0,up) = -0.1 + (-0.0)*1 = -0.1$

$r(0, do) + V^0(4)*p(4|0,do) = -0.1 + (-0.0)*1 = -0.1$

$\boxed{r(0, rig) + V^0(1)*p(1|0,rig) = -0.1 + (-0.0)*1 = -0.1}$

$r(0, lef) + V^0(0)*p(0|0,lef) = -0.1 + (-0.0)*1 = -0.1$

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V^0(1) = 0.0}$ $\boxed{V^1(1) = -0.1}$

$r(1, up) + V^0(1)*p(1|1,up) = -0.1 + (-0.0)*1 = -0.1$

$r(1, do) + V^0(1)*p(1|1,do) = -1.0 + (-0.0)*1 = -1.0$

$\boxed{r(1, rig) + V^0(2)*p(2|1,rig) = -0.1 + (-0.0)*1 = -0.1}$

$r(1, lef) + V^0(0)*p(0|1,lef) = -0.1 + (-0.0)*1 = -0.1$

Value Iteration (II)

Value Function V^1

– 0.1	– 0.1	– 0.1	0.0
– 0.1	0.0	– 0.1	0.0
– 0.1	– 0.1	– 0.1	– 0.1

$\boxed{V^1(0) = -0.1}$ $\boxed{V^2(0) = -0.2}$

$r(0, up) + V^1(0)*p(0|0,up) = -0.1 + (-0.1)*1 = -0.2$

$r(0, do) + V^1(4)*p(4|0,do) = -0.1 + (-0.1)*1 = -0.2$

$\boxed{r(0, rig) + V^1(1)*p(1|0,rig) = -0.1 + (-0.1)*1 = -0.2}$

$r(0, lef) + V^1(0)*p(0|0,lef) = -0.1 + (-0.1)*1 = -0.2$

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V^1(1) = -0.1}$ $\boxed{V^2(1) = -0.2}$

$r(1, up) + V^1(1)*p(1|1,up) = -0.1 + (-0.1)*1 = -0.2$

$r(1, do) + V^1(1)*p(1|1,do) = -1.0 + (-0.1)*1 = -1.1$

$\boxed{r(1, rig) + V^1(2)*p(2|1,rig) = -0.1 + (-0.1)*1 = -0.2}$

$r(1, lef) + V^1(0)*p(0|1,lef) = -0.1 + (-0.1)*1 = -0.2$

Value Iteration (III)

Value Function V^2

– 0.2	– 0.2	– 0.1	0.0
– 0.2	0.0	– 0.2	0.0
– 0.2	– 0.2	– 0.2	– 0.2

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V^2(0) = -0.2}$ $\boxed{V^3(0) = -0.3}$

$r(0, up) + V^2(0)*p(0|0,up) = -0.1 + (-0.2)*1 = -0.3$

$r(0, do) + V^2(4)*p(4|0,do) = -0.1 + (-0.2)*1 = -0.3$

$\boxed{r(0, rig) + V^2(1)*p(1|0,rig) = -0.1 + (-0.2)*1 = -0.3}$

$r(0, lef) + V^2(0)*p(0|0,lef) = -0.1 + (-0.2)*1 = -0.3$

$\boxed{V^2(1) = -0.2}$ $\boxed{V^3(1) = -0.2}$

$r(1, up) + V^2(1)*p(1|1,up) = -0.1 + (-0.2)*1 = -0.3$

$r(1, do) + V^2(1)*p(1|1,do) = -1.0 + (-0.2)*1 = -1.2$

$\boxed{r(1, rig) + V^2(2)*p(2|1,rig) = -0.1 + (-0.1)*1 = -0.2}$

$r(1, lef) + V^2(0)*p(0|1,lef) = -0.1 + (-0.2)*1 = -0.3$

Value Iteration (IV)

Value Function V^3

– 0.3	– 0.2	– 0.1	0.0
– 0.3	0.0	– 0.2	0.0
– 0.3	– 0.3	– 0.3	– 0.3

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V^3(0) = -0.2}$ $\boxed{V^4(0) = -0.3}$

$r(0, up) + V^3(0)*p(0|0,up) = -0.1 + (-0.3)*1 = -0.4$

$r(0, do) + V^3(4)*p(4|0,do) = -0.1 + (-0.3)*1 = -0.4$

$\boxed{r(0, rig) + V^3(1)*p(1|0,rig) = -0.1 + (-0.2)*1 = -0.3}$

$r(0, lef) + V^3(0)*p(0|0,lef) = -0.1 + (-0.3)*1 = -0.4$

$\boxed{V^3(1) = -0.2}$ $\boxed{V^4(1) = -0.2}$

$r(1, up) + V^3(1)*p(1|1,up) = -0.1 + (-0.2)*1 = -0.3$

$r(1, do) + V^3(1)*p(1|1,do) = -1.0 + (-0.2)*1 = -1.2$

$\boxed{r(1, rig) + V^3(2)*p(2|1,rig) = -0.1 + (-0.1)*1 = -0.2}$

$r(1, lef) + V^3(0)*p(0|1,lef) = -0.1 + (-0.3)*1 = -0.4$

Value Iteration (V)

Value Function V^4

– 0.3	– 0.2	– 0.1	0.0
– 0.4	0.0	– 0.2	0.0
– 0.4	– 0.4	– 0.3	– 0.4

$\boxed{V^4(0) = -0.2}$ $\boxed{V^5(0) = -0.3}$

$r(0, up) + V^1(0)*p(0|0,up) = -0.1+ (-0.3)*1= -0.4$

$r(0, do) + V^1(4)*p(4|0,do) = -0.1+ (-0.4)*1= -0.5$

$\boxed{r(0, rig) + V^1(1)*p(1|0,rig) = -0.1+ (-0.2)*1= -0.3}$

$r(0, lef) + V^1(0)*p(0|0,lef) = -0.1+ (-0.3)*1= -0.4$

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V^4(1) = -0.2}$ $\boxed{V^5(1) = -0.2}$

$r(1, up) + V^1(1)*p(1|1,up) = -0.1+ (-0.2)*1= -0.3$

$r(1, do) + V^1(1)*p(1|1,do) = -1.0+ (-0.2)*1= -1.2$

$\boxed{r(1, rig) + V^1(2)*p(2|1,rig) = -0.1+ (-0.1)*1= -0.2}$

$r(1, lef) + V^1(0)*p(0|1,lef) = -0.1+ (-0.3)*1= -0.4$

Stationary Value Function

Stationary Value Function

– 0.3	– 0.2	– 0.1	0.0
– 0.4	0.0	– 0.2	0.0
– 0.5	– 0.4	– 0.3	– 0.4

0	1	2	3
4	5	6	7
8	9	10	11

$\boxed{V(0) = -0.3}$

r(0, up) + V(0)*p(0|0,up) = – 0.1+ (– 0.3)*1= – 0.4

r(0, do) + V(4)*p(4|0,do) = – 0.1+ (– 0.4)*1= – 0.5

$\boxed{r(0, rig) + V(1)*p(1|0,rig) = -0.1+ (-0.2)*1= -0.3}$

r(0, lef) + V(0)*p(0|0,lef) = – 0.1+ (– 0.3)*1= – 0.4

$\boxed{V(1) = -0.2}$

r(1, up) + V(1)*p(1|1,up) = – 0.1+ (– 0.2)*1= – 0.3

r(1, do) + V(1)*p(1|1,do) = – 1.0+ (– 0.2)*1= – 1.0

$\boxed{r(1, rig) + V(2)*p(2|1,rig) = -0.1+ (-0.1)*1= -0.2}$

r(1, lef) + V(0)*p(0|1,lef) = – 0.1+ (– 0.3)*1= – 0.4

Optimal Policy for Value Iteration

Stationary Value Function

–0.3	–0.2	–0.1	0.0
–0.4	–0.0	–0.2	–0.0
–0.5	–0.4	–0.3	–0.4

V(0) = – 0.3

Optimal Action: right →

r(0, up) + V(0)*p(0|0,up) = – 0.1+ (– 0.3)*1= – 0.4

r(0, do) + V(4)*p(4|0,do) = – 0.1+ (– 0.4)*1= – 0.5

r(0, rig) + V(1)*p(1|0,rig) = – 0.1+ (– 0.2)*1= – 0.3

r(0, lef) + V(0)*p(1|0,lef) = – 0.1+ (– 0.3)*1= – 0.4

Optimal Policy

→	→	→	●
↑	□	↑	□
↑ →	→	↑	←

V(1) = – 0.2

Optimal Action: right →

r(1, up) + V(1)*p(1|1,up) = – 0.1+ (– 0.2)*1= – 0.3

r(1, do) + V(1)*p(1|1,do) = – 1.0+ (– 0.0)*1= – 1.0

r(1, rig) + V(2)*p(2|1,rig) = – 0.1+ (– 0.1)*1= – 0.2

r(1, lef) + V(0)*p(0|1,lef) = – 0.1+ (– 0.3)*1= – 0.4

Policy Iteration

- Often the optimal policy has been reached long before the value function has converged.
- Policy iteration (1) calculates a new policy based on the current value function and (2) then calculates a new value function based on this policy.

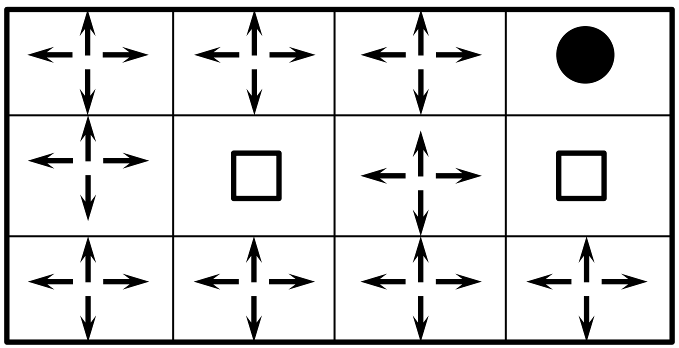
(1) Policy improvement $\pi^* = \underset{\pi}{\operatorname{argmax}} \quad R_T^\pi(x_t)$

(2) Policy evaluation $R_T^\pi(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau} \mid u_{t+\tau} = \pi(z_{1:t+\tau-1} u_{1:t+\tau-1})\right]$

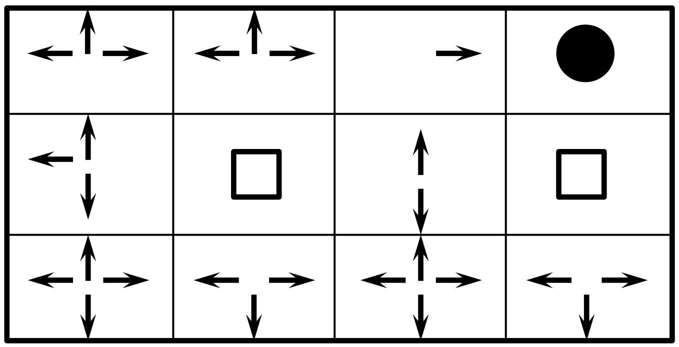
- Often converges faster to the optimal policy.

Policy Iteration (I)

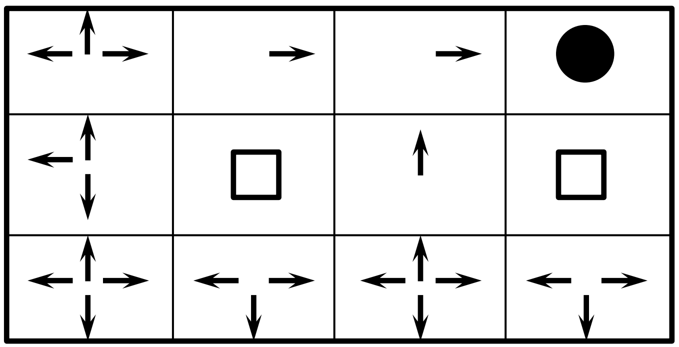
Policy π^0



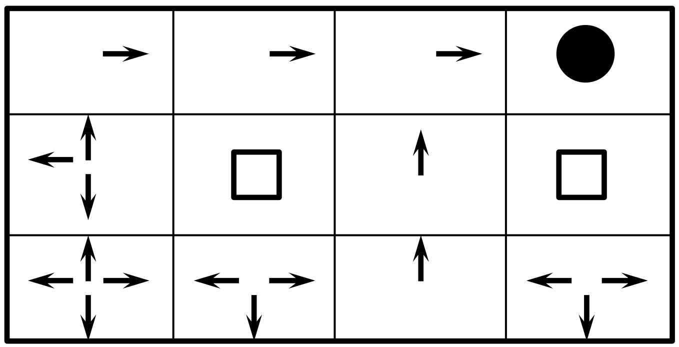
Policy π^1



Policy π^2

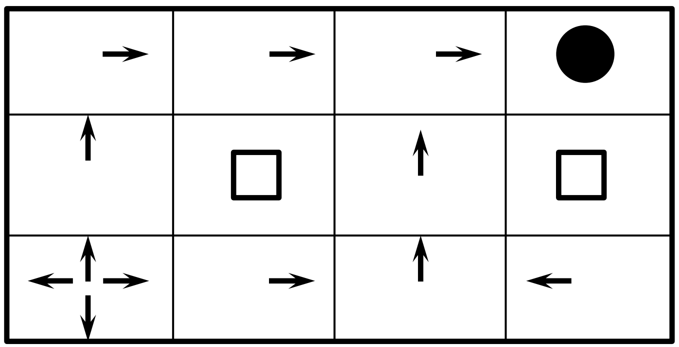


Policy π^3

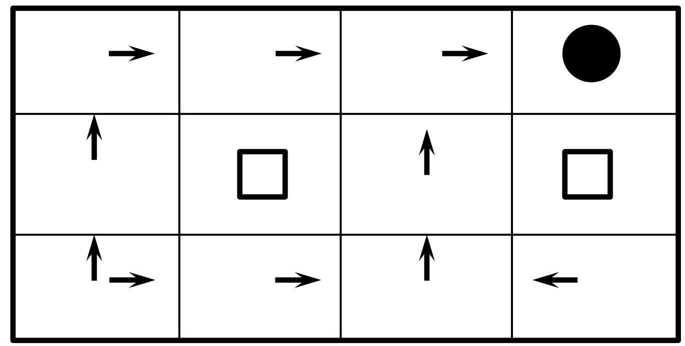


Policy Iteration (II)

Policy π^4



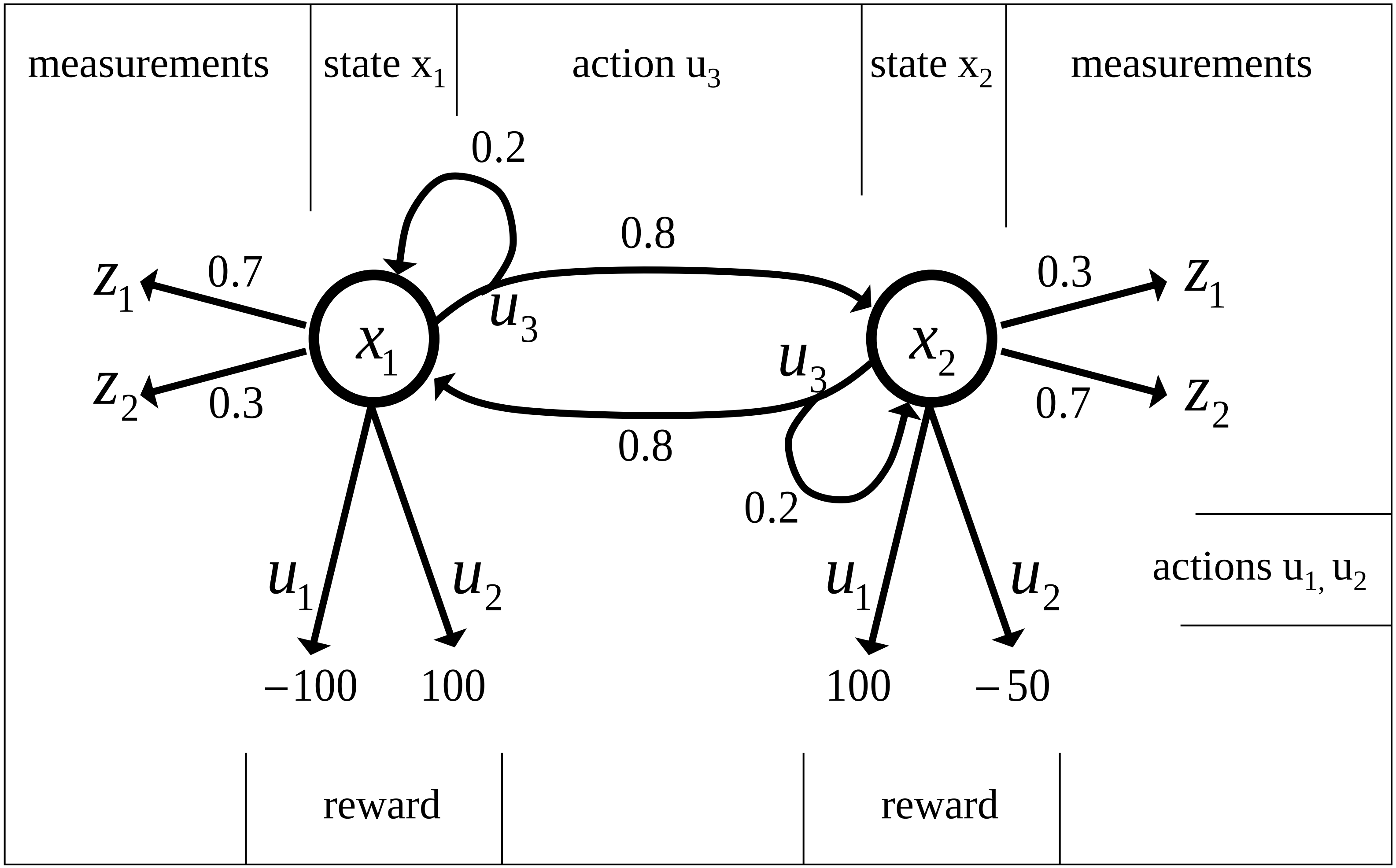
Policy π^5



Value Function

–0.3	–0.2	–0.1	0.0
–0.4	–0.0	–0.2	–0.0
–0.5	–0.4	–0.3	–0.4

Partially Observable MDP



More Course Links

Stanford Machine Learning：
https://see.stanford.edu/Course/CS229/47

MIT Machine Learning： https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/index.htm

Stanford CNN for Vision： http://cs231n.stanford.edu

Stanford Deep Learning: http://cs230.stanford.edu/syllabus.html

MIT Deep Learning: http://introtodeeplearning.com/