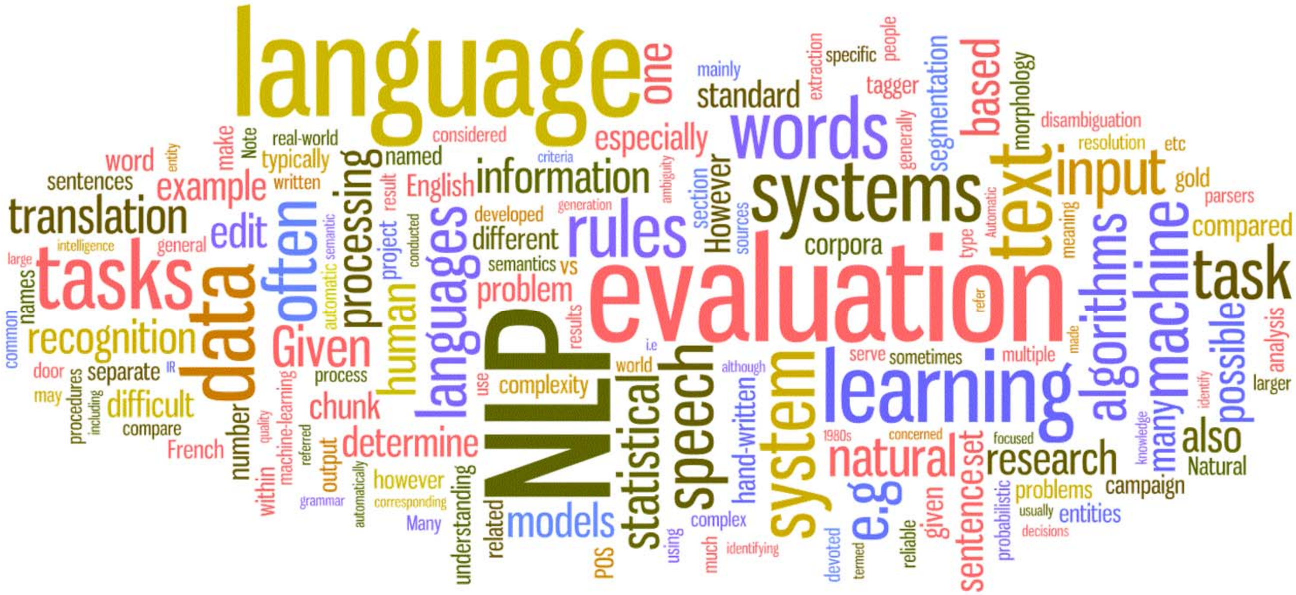


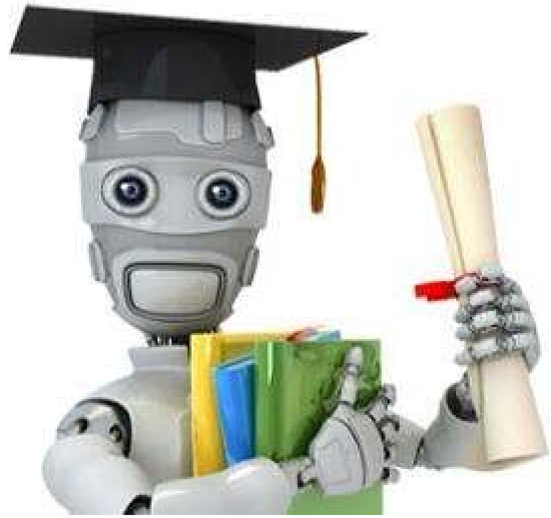
Lecture3: Machine Learning Foundations

Xu Ruifeng

Harbin Institute of Technology, Shenzhen

Today's class

- What is machine learning?
- Linear Classifier
    - Logistic Regression
    - Support Vector Machine
- Structured Learning
    - Hidden Markov Model
    - Condition Random Fields



What is Machine Learning?

- **Common theme** is to solve a prediction problem:
  - given an **input** x,
  - **predict** an “appropriate” **output** y.
- Let’s start with a few examples…

Example: Spam Detection

- **Input:** Incoming email
- **Output**: “SPAM” or “NOT SPAM”
- A **classification** problem

Example: Sentiment Analysis of Movie Reviews

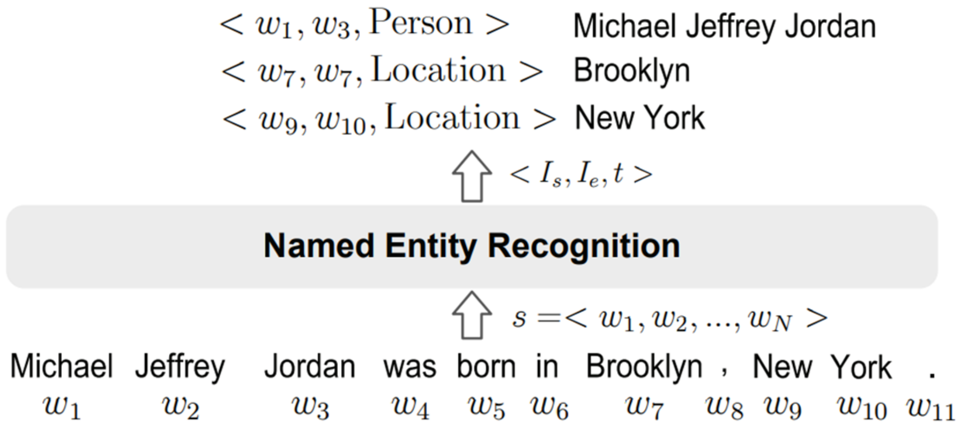
- **Input:** Movie Review

- **Output**: Recommended level (1~5)

- A **regression** problem

Example: Named Entity Recognition

- **Input:** Sentence



- **Output**: Names of people, location, organization, etc

- A **labeling** problem

The Prediction Function

- A **prediction function** takes input x and produces an output y.
- We're looking for prediction functions that solve particular problems.
- Rule-based system
  - labor-intensive
  - poor generalization ability
- Machine Learning： Learning from data.

Machine Learning Algorithm

- Learning from the "**training data**".
  - many examples of (input x, output y) pairs.
  - e.g. A set of emails, and whether or not each is SPAM.
  - e.g. A set of movie reviews, and corresponding sentiment scores.
- A **machine learning algorithm**:
  - **input**: Training Data
  - "learns" from the training data.
  - **output**: A "prediction function" that produces output y given input x.

Machine Learning Algorithm

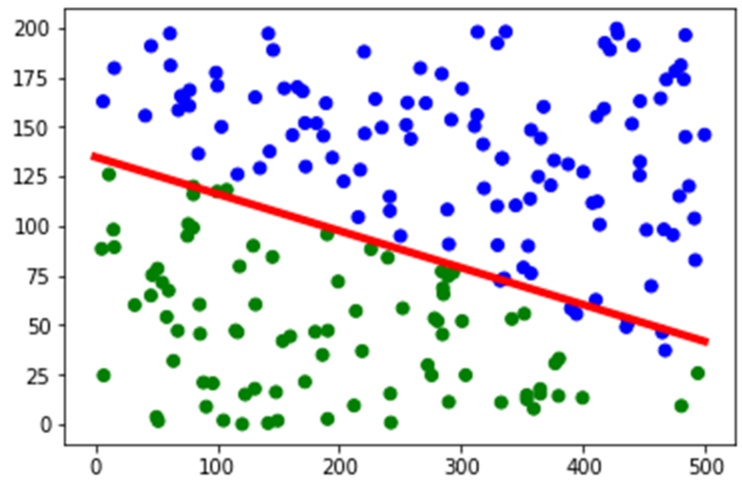
- Every machine learning algorithm has three components:
  - **representation**: How to model the relationship between input x and output y.
    - hypothesis space: all possible conditional probability distributions $p(y|x;\theta)$ or decision functions $g(x;\theta)$.
  - **evaluation**: The way to evaluate candidate hypothesis function.
    - loss Function, Empirical Risk Minimization
  - **optimization**: The algorithm to determine the parameters in the hypothesis function.

Machine Learning Algorithm

- Types of Learning
  - supervised learning
  - unsupervised Learning
  - reinforcement Learning

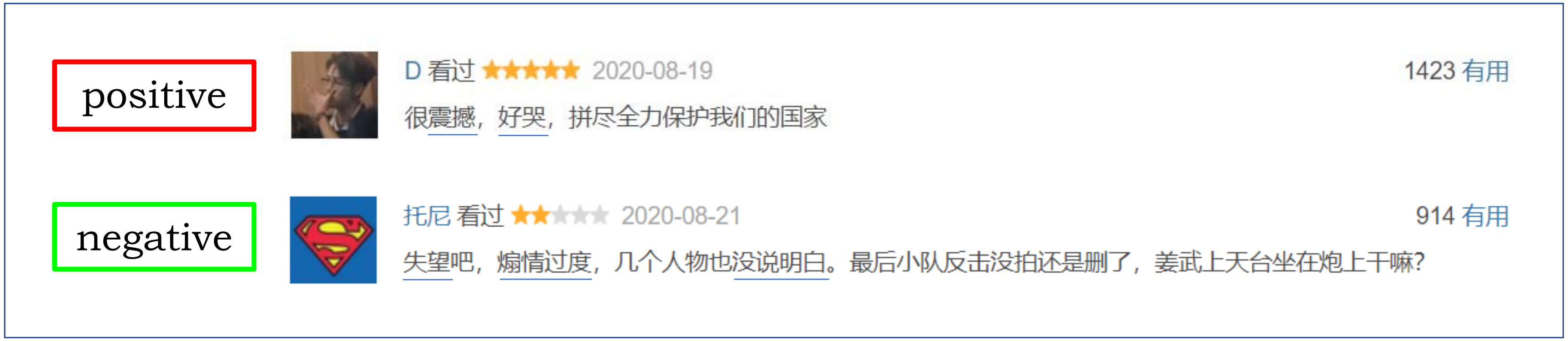
Today's class

- What is machine learning?
- **Linear Classifier**
  - Logistic Regression
  - Support Vector Machine
- Structured learning
  - Hidden Markov Model
  - Condition Random Field



Linear Classifier

- Task: Sentiment classification
  - to determine whether the sentiment polarity expressed by a movie review is positive or negative.



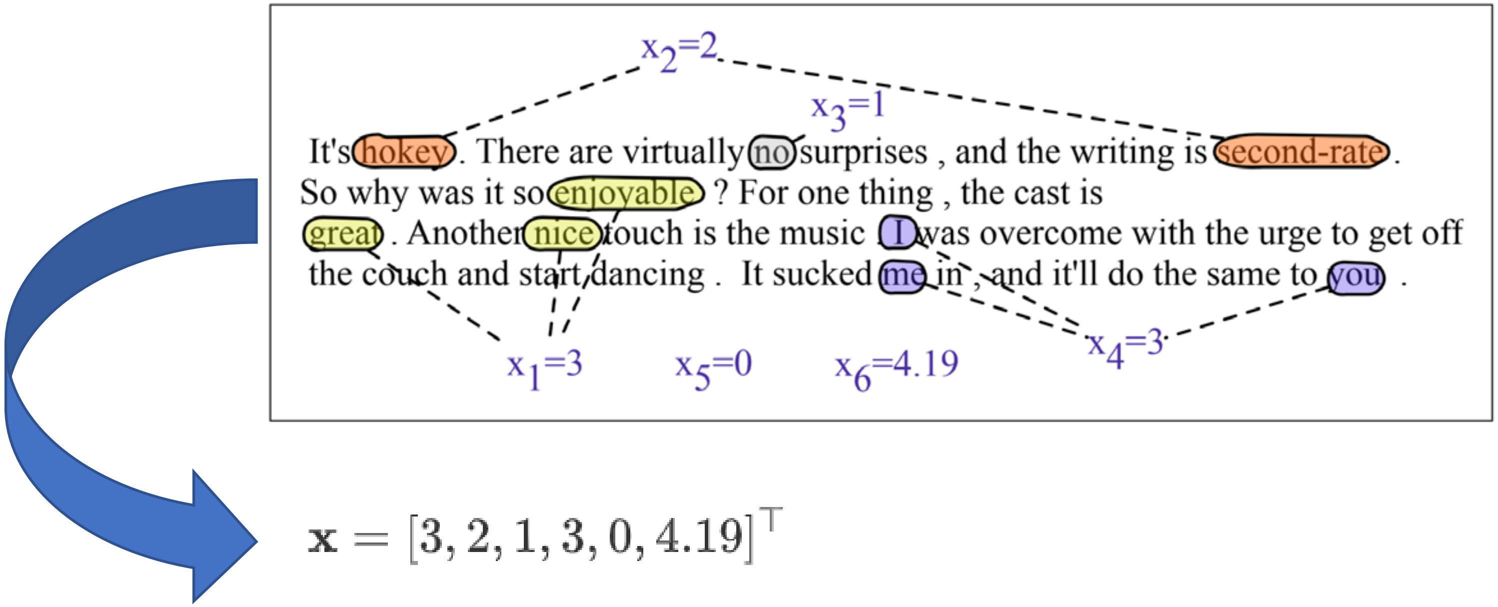
Linear Classifier

- Feature extraction：
  - to construct a feature vector, e.g.

Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Linear Classifier

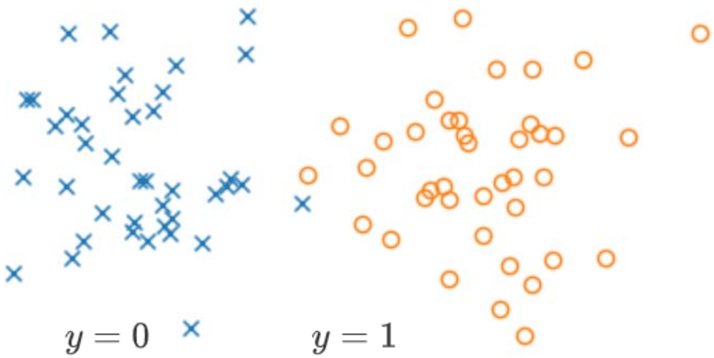
- Feature extraction：
  - to construct a feature vector, e.g.



$$\mathbf{x} = [3, 2, 1, 3, 0, 4.19]^\top$$

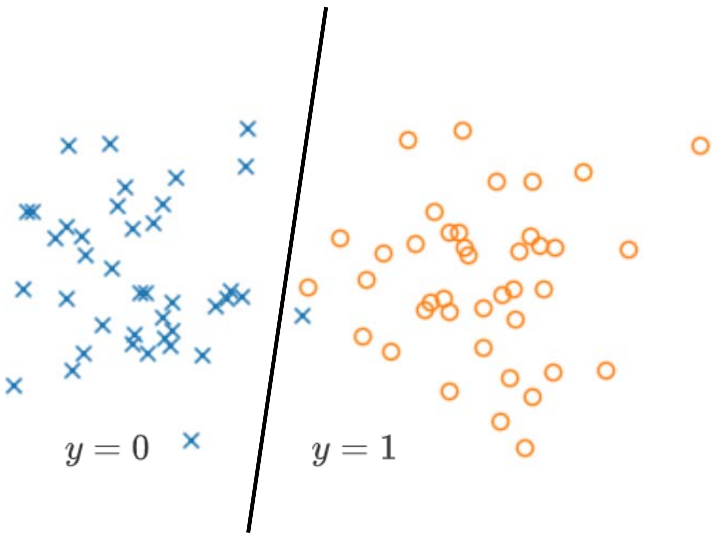
Linear Classifier

- Feature extraction：
  - an n-dimensional feature vector is a point in the n-dimensional space(n维空间).



Linear Classifier

- Feature extraction：
  - an n-dimensional feature vector is a point in the n-dimensional space(n维空间).



Linear Classifier

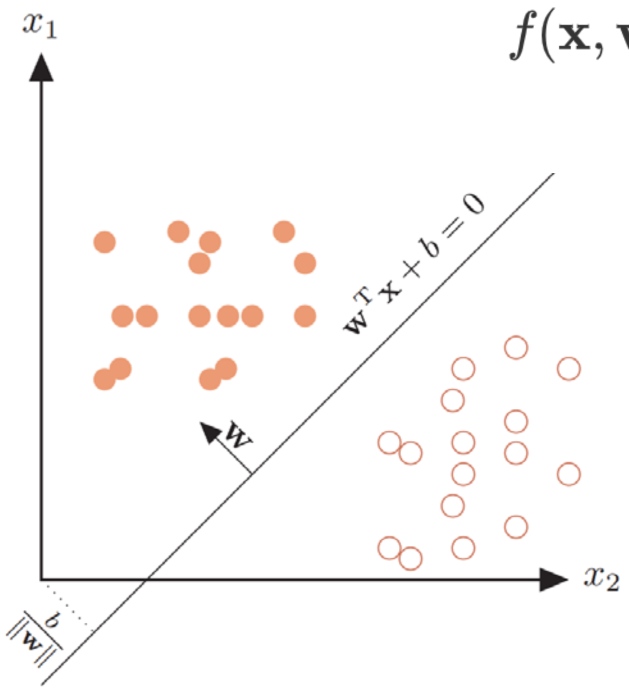
- Linear discriminant function.
  - given a sample $\mathbf{x} = [x_1, \dots, x_d]^\top$, let $f(\mathbf{x}, \mathbf{w})$ be

$$
\begin{aligned}
f(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^\top \mathbf{x} + b \\
&= w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b
\end{aligned}
$$

  - $\mathbf{w} = [w_1, \dots, w_d]^\top$ is a vector of **weights**, b is a **bias term**.

Linear Classifier

- Linear discriminant function.
  - decision boundary(决策边界), hyperplane(超平面)

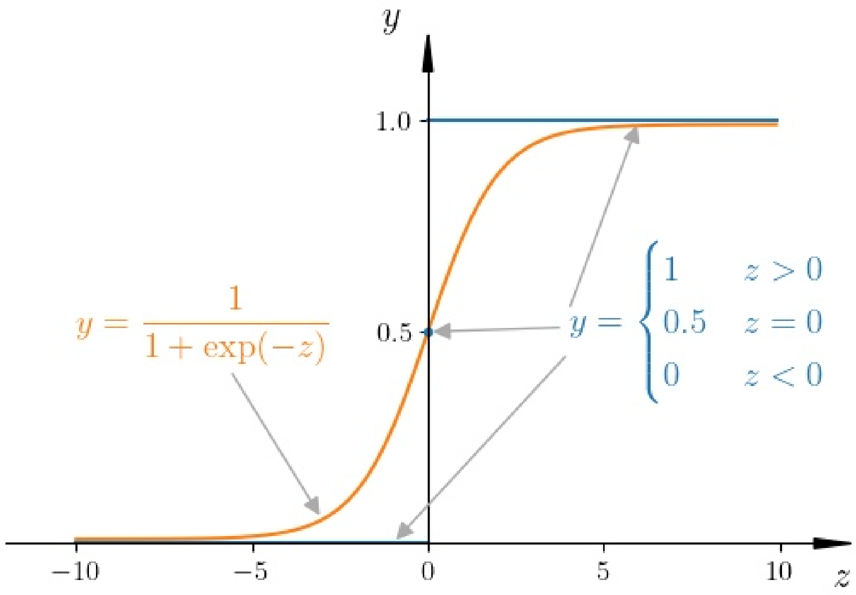


$$
\begin{aligned}
f(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^\top \mathbf{x} + b \\
&= w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b
\end{aligned}
$$

Logistic Regression

- For a binary classification, $y \in \{0, 1\}$, but $f(\mathbf{x}, \mathbf{w}) \in R$.
  - unit-step function 单位阶跃函数
  - logistic function (sigmoid)

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



Logistic Regression

- Conditional Probability

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x} + b)}$$

$$p(y=0|\mathbf{x}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

$$\hat{y} = \begin{cases} 1 & \text{if } \ p(y=1|x) > 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

Logistic Regression

- Conditional Probability
  - let's assume that we've already learned the weights and the bias term.

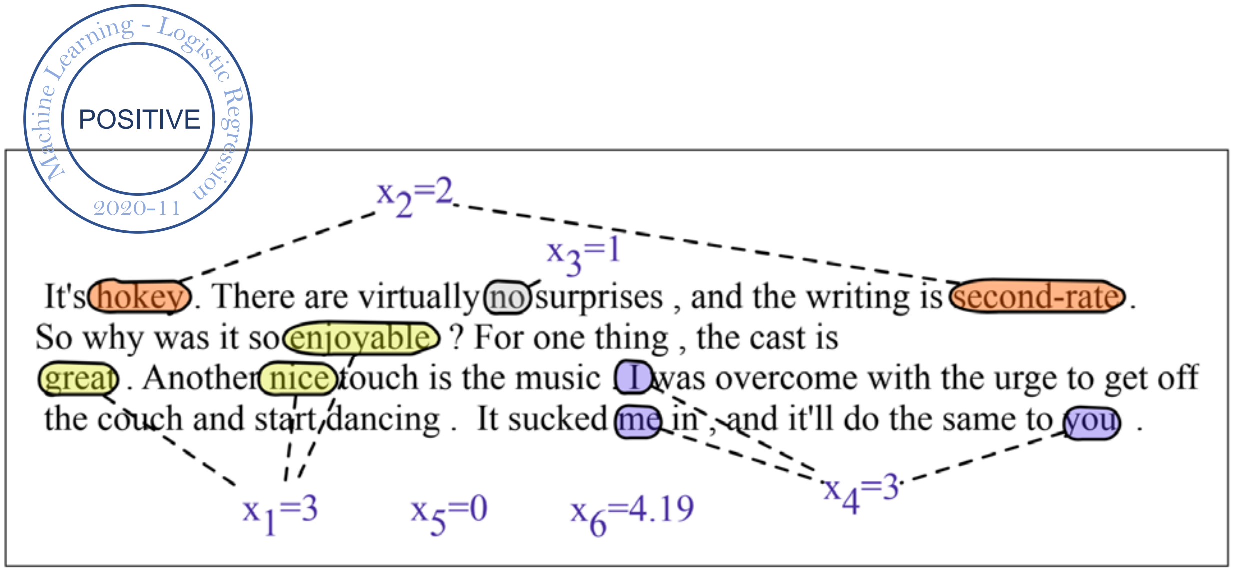
$$\mathbf{x} = [3, 2, 1, 3, 0, 4.19]^\top$$

$$\mathbf{w} = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]^\top, b = 0.1$$

$$\begin{aligned}
p(y=1|\mathbf{x}) &= \sigma(\mathbf{w}^T\mathbf{x} + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19]^\top + 0.1) \\
&= \sigma(0.833) \\
&= 0.70 \\
p(y=0|\mathbf{x}) &= 1 - \sigma(\mathbf{w}^\top\mathbf{x} + b) \\
&= 0.30
\end{aligned}$$

Logistic Regression

- Conditional Probability
  - $p(y = 1|\mathbf{x}) = 0.7, p(y = 0|\mathbf{x}) = 0.3.$



Logistic Regression

- Learning in Logistic Regression
  - to determine the parameters of the model, the weights $\mathbf{w}$ and bias b
  - we want to learn parameters that make $\hat{y}$ for each training sample as close as possible to the true y.
  - **loss function**: the distance between the system output $\hat{y}$ and the gold output y.
  - **gradient descent**: an optimization algorithm for iteratively updating the parameters so as to minimize the loss function

Logistic Regression

- Cross-entropy Loss Function(交叉熵损失函数)
  - given N train samples $\left\{(\mathbf{x}^{(n)}, y^{n})\right\}_{n=1}^{N}$
  - let $\hat{y} = \mathrm{p}(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^{\top}\mathbf{x} + b)$

$$L(\mathbf{w}, b) = -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\log\hat{y}^{(n)} + (1 - y^{(n)})\log(1 - \hat{y}^{(n)})\right).$$

Logistic Regression

- Cross-entropy loss function

$$L(\mathbf{w}, b) = -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\log\hat{y}^{(n)} + (1-y^{(n)})\log(1-\hat{y}^{(n)})\right).$$

$$L^{(n)}(\mathbf{w}, b) = -\left(y^{(n)}\log\hat{y}^{(n)} + (1-y^{(n)})\log(1-\hat{y}^{(n)})\right)$$

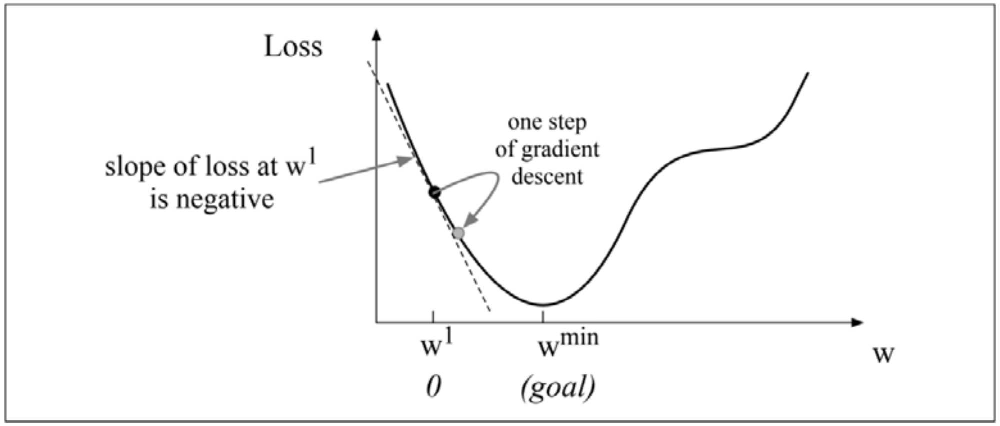
$$\text{if } \ y^{(n)} = 0, \ \ \min L^{(n)}(\mathbf{w}, b) \Longrightarrow \max\log(1-\hat{y}^{(n)}) \Longrightarrow \min\hat{y}^{(n)}$$

$$\text{if } \ y^{(n)} = 1, \ \ \min L^{(n)}(\mathbf{w}, b) \Longrightarrow \max\log\hat{y}^{(n)} \Longrightarrow \max\hat{y}^{(n)}$$

Logistic Regression

- Gradient Descent

$$\hat{\mathbf{w}}, \hat{b} = \arg\min_{\mathbf{w},b} L(\mathbf{w}, b)$$



- 沿着负梯度方向更新参数

Logistic Regression

- Gradient Descent
  - partial derivative

$$\frac{\partial L}{\partial \mathbf{w}} = -\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}(y^{(n)} - \hat{y}^{(n)}).$$

$$\frac{\partial L}{\partial b} = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)} - \hat{y}^{(n)}).$$

  - update **w**, b (沿着负梯度方向更新参数)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha\frac{\partial L}{\partial \mathbf{w}}$$

$$b_{t+1} \leftarrow b_t - \alpha\frac{\partial L}{\partial b}$$

  - α is the step (learning rate)

Logistic Regression

- Regularization
  - if the weights are learned too perfect that the model completely match the training data, noisy factors that just accidentally correlate with the class will also be modeled.
  - this problem is called **overfitting**.
  - one way to avoid overfitting is to add a **regularization term** $R(\theta)$ to the objective function to penalize large weights

$$\hat{\mathbf{w}}, \hat{b} = \arg\min_{\mathbf{w},b} L(\mathbf{w}, b) + \alpha R(\mathbf{w}, b)$$

Logistic Regression

- Regularization
  - let $\theta = [w_1, w_2, \dots, w_d, b]^\top$
  - l1 regularization

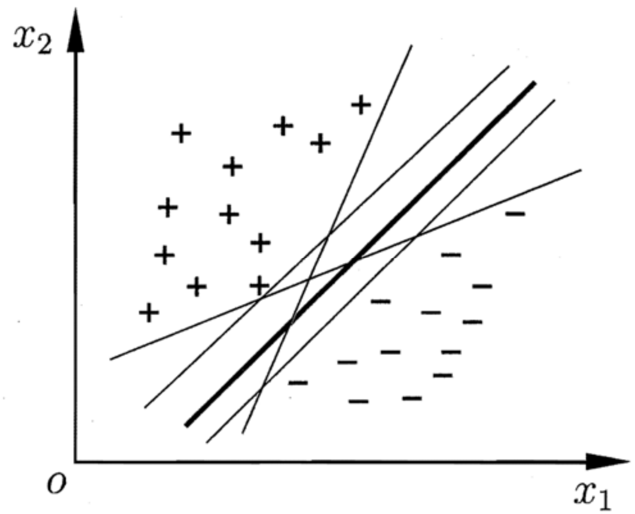
$$R(\theta) = \|\theta\|_1 = \sum_{i=1}^{d+1} \theta_i$$

  - l2 regularization

$$R(\theta) = \frac{1}{2}\|\theta\|_2^2 = \frac{1}{2}\sum_{i=1}^{d+1} \theta_i^2$$

Support Vector Machine

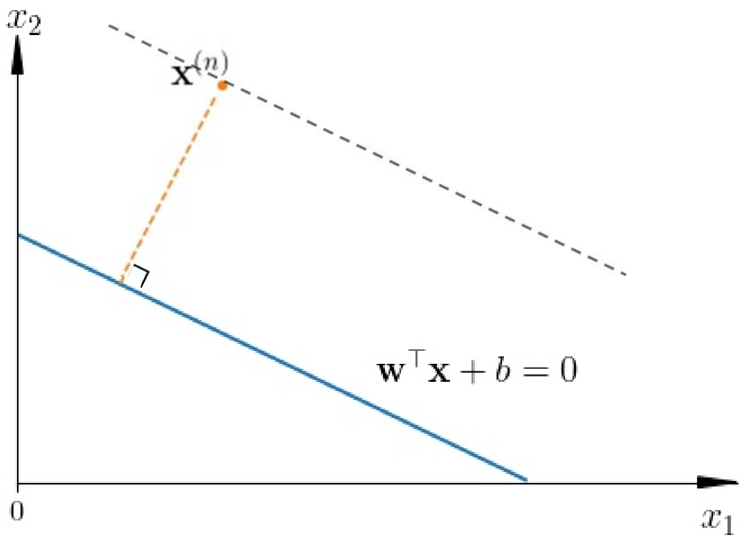
- Motivation
  - the decision boundary could separate samples of two classes
  - the **maximum margin** solution (间隔最大化准则)
    - the decision boundary is chosen to be the one for which the margin is maximized.



Support Vector Machine

- Motivation
  - margin:

$$\gamma^{(n)} = \frac{|\mathbf{w}^\top \mathbf{x}^{(n)} + b|}{\|\mathbf{w}\|}$$



Support Vector Machine

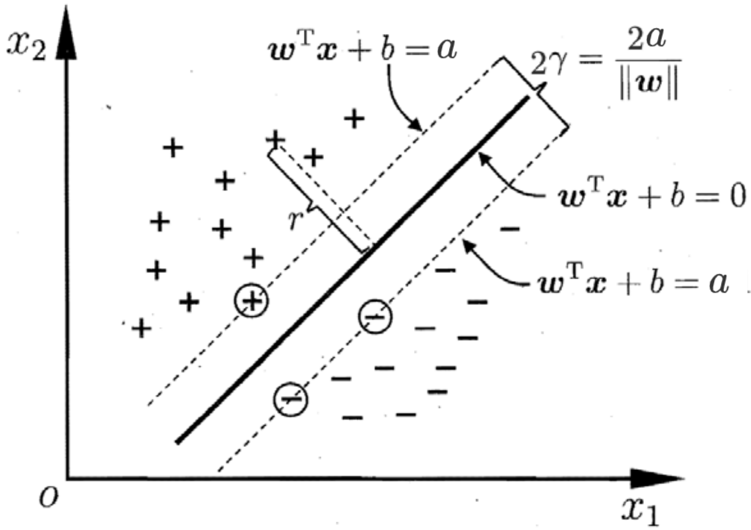
- Motivation

  - assuming that all samples are correctly classified, define $\gamma = \frac{a}{||w||}$ as the **minimum margin** between all samples and the decision boundary.

    $$y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} + b) \geq a$$

  - the **maximum margin** solution

    $$\max \gamma \Longrightarrow \max \frac{a}{\|\mathbf{w}\|}$$



Support Vector Machine

- Motivation
  - objective function

$$\max_{\mathbf{w},b} \frac{a}{\|\mathbf{w}\|}$$
$$\text{s.t.} \quad y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} + b) \geq a, \forall n$$

  - 等比例地缩放$\mathbf{w}$和b：

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|}$$
$$\text{s.t.} \quad y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} + b) \geq 1, \forall n$$

$$\Longrightarrow$$

$$\boxed{\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t.} \quad 1 - y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} + b) \leq 1, \forall n \end{aligned}}$$

Support Vector Machine

- Optimization
  - this is a convex quadratic programming(凸二次规划) problem
  - this could by solved by Lagrange multiplier method(拉格朗日乘子法)

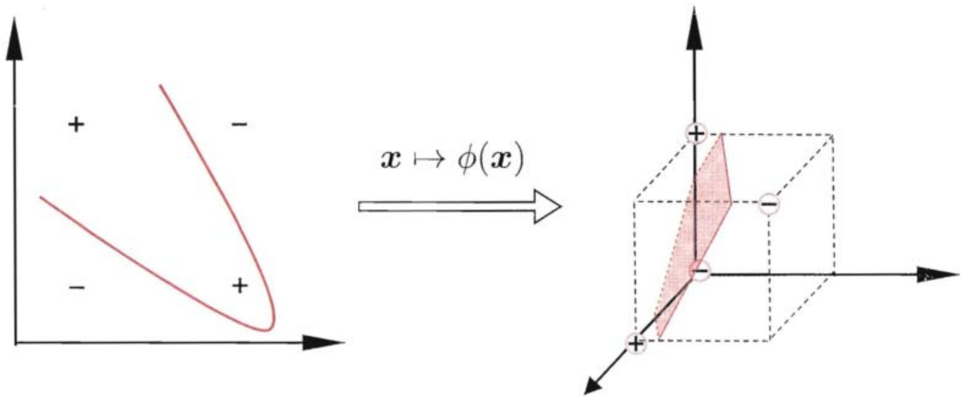
$$
\begin{aligned}
\min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\
\text{s.t.} \quad & 1 - y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} + b) \le 1, \forall n
\end{aligned}
$$

dual form

$$
\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^\top \mathbf{x}^{(j)} \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0, \\
& \alpha_i \ge 0, \forall n
\end{aligned}
$$

Support Vector Machine

- Kernel function(核函数)
  - to deal with linearly inseparability problem(线性不可分问题), it's necessary to map samples from the original feature space to a higher-dimensional space by **kernel function**

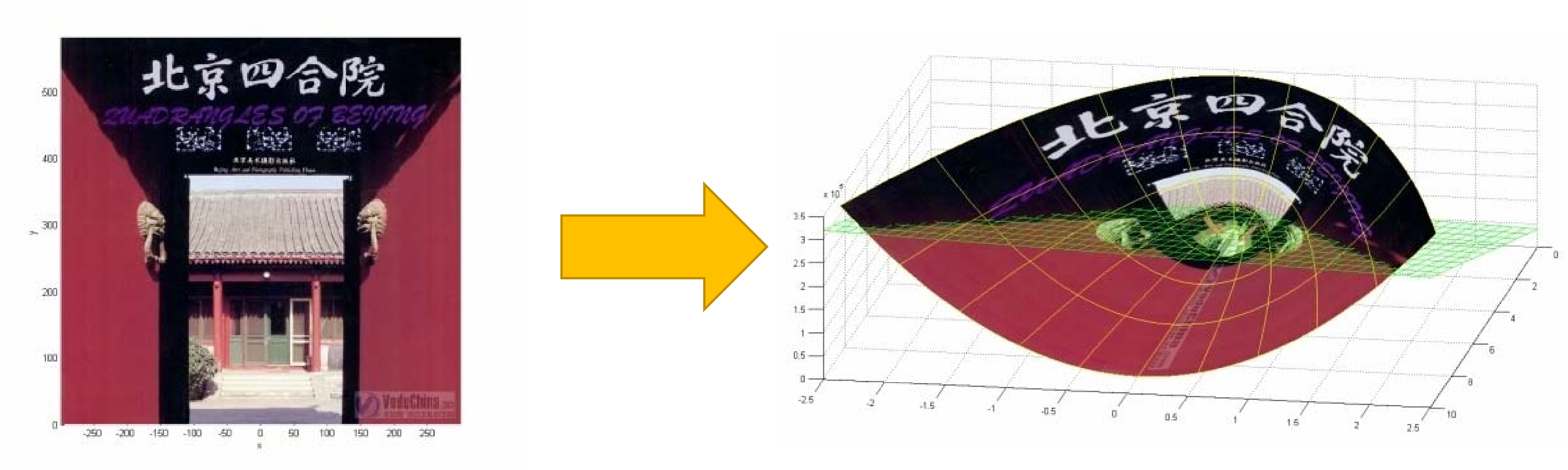


Support Vector Machine

The **inner product square kernel function** can map a picture from a 2-dimensional space to a 3-dimensional space.

$$K(v_1, v_2) = < v_1, v_2 >^2 = \left(x^2, \sqrt{2}xy, y^2\right)$$

$$v_1 = (x_1, y_1), v_2 = (x_2, y_2)$$



Support Vector Machine

- Kernel function
  - decision boundary

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

the cost of calculating $\phi(\boldsymbol{x})^\top \phi(\mathbf{z})$ is expensive, so define

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

kernel function

  - dual form of objective function

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \underline{\phi(\mathbf{x}^{(i)})^\top \phi(\mathbf{x}^{(j)})} \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0, \\
& \alpha_i \geq 0, \forall n
\end{aligned}$$

Support Vector Machine

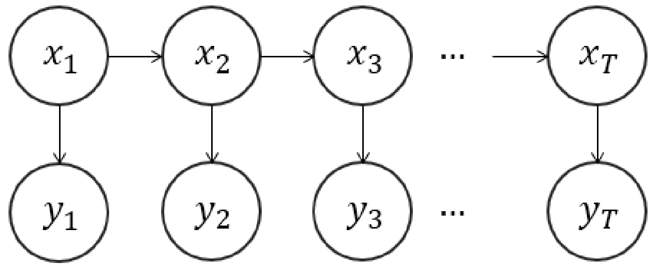
- kernel functions commonly used in machine learning

表 6.1 常用核函数

名称	表达式	参数
线性核	$\kappa(\mathbf{x},\mathbf{z}) = \mathbf{x}^\top\mathbf{z}$	
多项式核	$\kappa(\mathbf{x},\mathbf{z}) = (\mathbf{x}^\top\mathbf{z}+1)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\mathbf{x},\mathbf{z}) = \exp(-\frac{\lVert\mathbf{x}-\mathbf{z}\rVert^2}{2\sigma^2})$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\mathbf{x},\mathbf{z}) = \exp(-\frac{\lVert\mathbf{x}-\mathbf{z}\rVert}{\sigma})$	$\sigma > 0$
Sigmoid 核	$\kappa(\mathbf{x},\mathbf{z}) = \tanh(\beta\mathbf{x}^\top\mathbf{z}+\theta)$	tanh 为双曲正切函数, $\beta > 0$, $\theta < 0$

Today’s class

- What is machine learning?
- Linear Classifier
  - Logistic Regression
  - Support Vector Machine
- **Structured learning**
  - Hidden Markov Model
  - Condition Random Field



Structured learning

- A classifier predicts which class the input sample belongs to.
- The output of **structured learning**(结构化学习) is no longer discrete labels, but structured objects, such as sequences, trees, or graphs.
- e.g. Part-of-Speech(词性标注)

生活	的	理想	是	理想	的	生活
名词	助词	名词	动词	形容词	助词	名词

Sequence

- Let's observe the following word sequence and pos sequence.

生活	的	理想	是	理想	的	生活
名词	助词	名词	动词	形容词	助词	名词

Sequence

- Let's observe the following word sequence and pos sequence.
  - “是” 一般是名词，“的”一般是助词
  - “理想”可以做名词，也可以做形容词

生活	的	理想	是	理想	的	生活
↓	↓	↓	↓	↓	↓	↓
名词	助词	名词	动词	形容词	助词	名词

P(词性|单词)	名词	助词	动词	形容词
生活	1			
的		1		
理想	0.5			0.5
是			1	

Sequence

- Let's observe the following word sequence and pos sequence.
  - “是”一般是动词，“的”一般是助词
  - “理想”可以做名词，也可以做形容词

生活	的	理想	是	理想	的	生活
↑	↑	↑	↑	↑	↑	↑
名词	助词	名词	动词	形容词	助词	名词

P(单词|词性)	生活	的	理想	是
名词	0.5		0.5	
助词		1		
动词				1
形容词			1	

Sequence

- Let's observe the following word sequence and pos sequence.
  - “形容词”后面一般接“助词”
  - “助词”后面一般接“名词”

生活	的	理想	是	理想	的	生活

名词 → 助词 → 名词 → 动词 → 形容词 → 助词 → 名词

P(后一个词的词性|前一个词的词性)	名词	助词	动词	形容词
名词		0.5	0.5	
助词	1			
动词				1
形容词		1		

Hidden Markov Model

- Observation(观测) and State(状态)
  - observation x_t， observation sequence $\mathbf{x}$

$$\mathbf{x} = (x_1, x_2, \cdots, x_T)$$

生活 的 理想 是 理想 的 生活

  - Hidden state z_t， state sequence $\mathbf{z}$

$$\mathbf{y} = (y_1, y_2, \cdots, y_T)$$

名词 助词 名词 动词 形容词 助词 名词

  - the observation is visible, the state is invisible.

Hidden Markov Model

- Observation(观测) and State(状态)
  - observation space $O = \{o_1, o_2, \ldots, o_M\}$ Vocabulary
  - state space $S = \{s_1, s_2, \ldots, S_N\}$ Part-of-speech set

ag	adjective morpheme	绿色/n似/d锦/ag	a	adjective	重要/a 步伐/n
ad	adverb-adjective	积极/ad 谋求/v	an	adnoun	克服/v 困难/an
bg	distinguish morpheme	一个/m 次/bg 地区/n	b	distinguish word	女/b 司机/n
c	conjunction	合作/vn 与/c 伙伴/n	dg	adverb morpheme	了解/v 甚/dg 深/a
d	adverb	进一步/d 发展/v	e	exclamation	啊/e
f	position word	贵州/ns 南部/f	h	heading element	非/h 主角/n
i	idiom	一言一行/i	j	abbreviation	德/j 外长/n
k	tail element	朋友/n 们/k	l	habitual word	落到实处/l
mg	numeral morpheme	让/v 乙/mg 背上/v	m	numeral	三/m 个/q
ng	noun morpheme	出/v 两/m 天/q 差/ng	n	noun	科技/n 文献/n
nr	person's name	朴/nr 贞爱/nr	ns	toponym	安徽/ns
nt	organization proper noun	联合国/nt	nx	foreign character	24/m K/nx
nz	other proper noun	满族/nz	o	onomatopoeia	哈哈/o 笑/v
p	preposition	对/p 子孙/n 负责/v	q	quantifier	首/m 批/q
rg	pronoun morpheme	成长/v 于/p 斯/rg	r	pronoun	本/r 地区/n
s	location word	西部/s 交通/n	tg	time morpheme	3 日/t 晚/tg
t	time	下午/t 2时/t	u	auxiliary	填平/v 了/u
vg	verb morpheme	洗/v 了/u 澡/vg	v	verb	编辑/v文献/n
vd	adverb-verb	持续/vd 好转/v	vn	gerund	收费/vn 电话/n
w	punctuation	”/w	yg	modal morpheme	致/v 之/u 耳/Yg
y	modal word	又/d 何在/v 呢/y	z	state word	短短/z 几/m 年

Hidden Markov Model

- Application
  - **from observation to state:** given a model and some observations, can we estimate the unobservable sequence of states?

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} \underline{p(\mathbf{z}|\mathbf{x})}$$

how to calculate it?

Hidden Markov Model

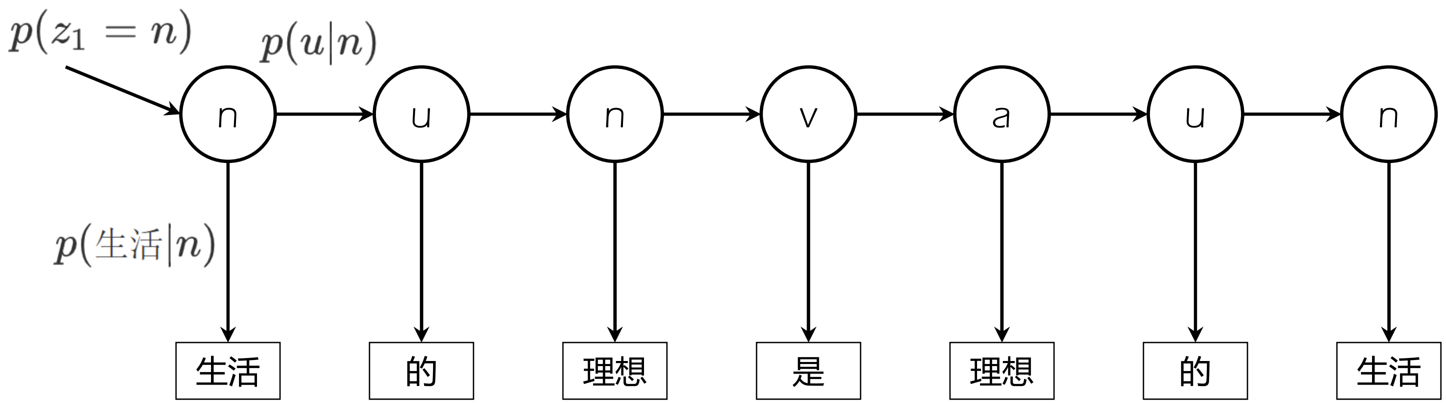
- Application
  - **from observation to state:** given a model and some observations, can we estimate the unobservable sequence of states?

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}) = \arg\max_{\mathbf{z}} \underline{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}$$

1. generate the sequence of states
2. generate the observations by the states

Hidden Markov Model

- Observation(观测) and State(状态)



Hidden Markov Model

- Two basic assumptions
  - observational independence hypothesis： the x_t of each moment only depends on its tag z_t.
  - homogeneous Markov property： z_t depends on z_{t-1} at the previous moment:

$$
\begin{aligned}
p(\mathbf{z}|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \\
&\propto p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \\
&= \left[\prod_{t=1}^{T} p(x_t|z_t)\right] p(z_1) \left[\prod_{t=2}^{T} p(z_t|z_{t-1})\right]
\end{aligned}
$$

Hidden Markov Model

- Two basic assumptions

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

$$\propto p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

$$= \left[\prod_{t=1}^{T} p(x_t|z_t)\right] p(z_1) \left[\prod_{t=2}^{T} p(z_t|z_{t-1})\right]$$

transition probabilities
状态转移概率

observation probabilities
观测概率

emission probabilities
发射概率

initial state probabilities
初始状态概率

Hidden Markov Model

- Example：
  - 观测：拼音；状态：汉字
    - ta'zhang'le'yi'tou'chang'tou'fa

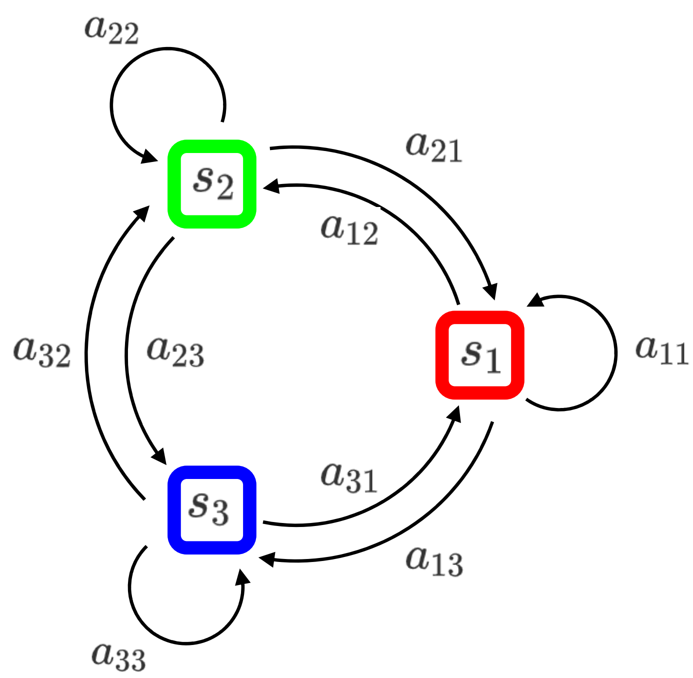
  - 如我要打“张”这个字，一般可能的输入是“zh”、“zhang”、“zhagn”等。另一方面，文字与文字之间也是有一些转移规律的。
  - 利用单个文字的输入统计规律、以及文字与文字之间的转移规律这两方面的信息，从一段字符序列推断对应的输入文字也不是什么难事了。

Hidden Markov Model

- HMM parameter representation $\lambda = (A, B, \pi)$
  - A is the state transition matrix 状态转移

$$A = [a_{ij}]_{N \times N}$$

  - $a_{ij} = p(z_{t+1} = s_j | z_t = s_i)$
  - $\sum_j a_{ij} = 1$

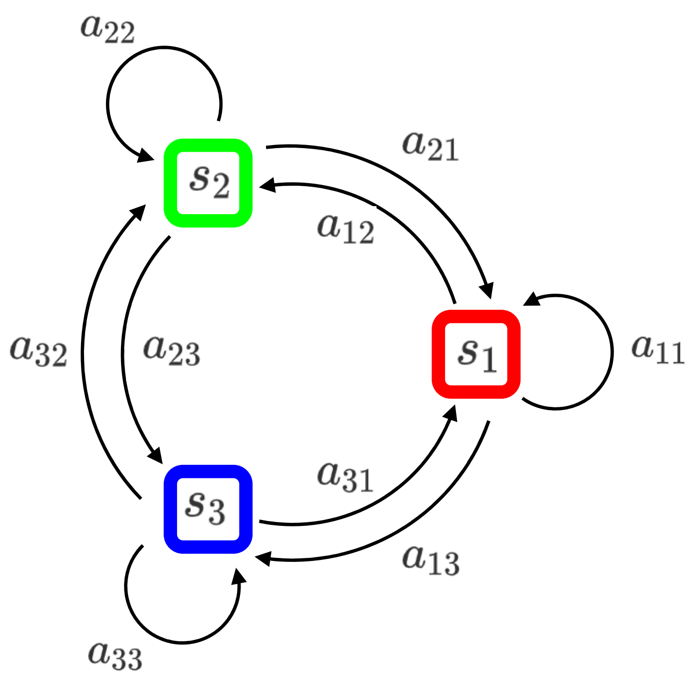


Hidden Markov Model

- HMM parameter representation $\lambda = (A, B, \pi)$
  - π is the vector of the initial state probabilities

$$\pi = (\pi_i)$$

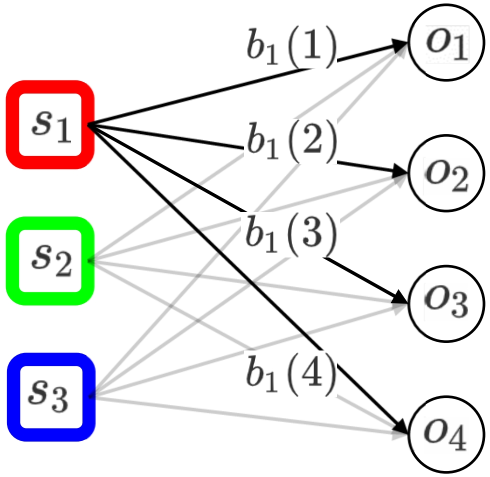
  - $\pi_i = p(z_1 = s_i)$, and $\sum_{i=1}^{N} \pi_i = 1$



Hidden Markov Model

- HMM parameter representation $\lambda = (A, B, \pi)$
  - B is the observation probability matrix

$$B = [b_j(k)]_{N \times M}$$



  - $b_j(k) = p(x_t = o_k | z_t = s_j)$

Hidden Markov Model

- HMM parameter representation
  - A and π determine how to generate the hidden state sequence
  - B determines how to generate observations from the state
  - Together they determine how to generate observation sequences.

Hidden Markov Model

- Three Basic HMM Problems
  - **Evaluation.** Given the observation sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$ and an HMM model $\lambda = (A, B, \pi)$, how to compute the probability of the observation sequence $P(\mathbf{x}|\lambda)$? 估值问题
  - **Decoding.** Given the observation sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$ and an HMM model $\lambda = (A, B, \pi)$, how to find the state sequence that best explains the observations $P(\mathbf{z}|\mathbf{x}, \lambda)$? 解码问题
  - **Learning.** How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\mathbf{x}|\lambda)$? 学习问题

Evaluation Problem

- Direct calculation

$$P(\mathbf{x}|\lambda) = \sum_{\mathbf{z}} \underline{P(\mathbf{x}|\mathbf{z}, \lambda) P(\mathbf{z}|\lambda)}$$

$$P(\mathbf{x}|\mathbf{z}, \lambda) = \prod_{t=1}^{T} b_{z_t}(x_t)$$

$$P(\mathbf{z}|\lambda) = \pi_{z_1} \prod_{t=2}^{T} a_{z_{t-1} z_t}$$

  - there are N^T possible state sequences, and the computational complexity is $O(TN^T)$.
  - N is the number of possible state, T is length of sequence
  - the solution is dynamic programming.

Evaluation Problem

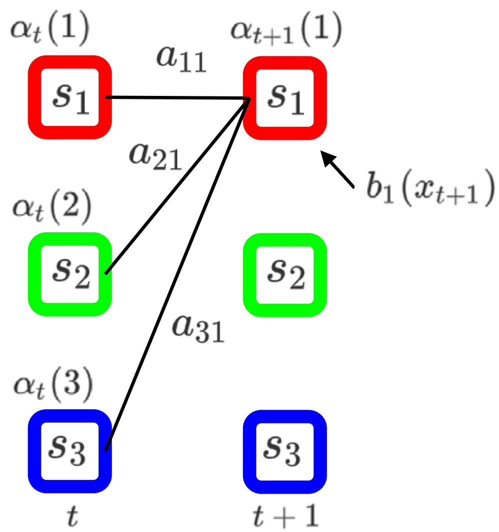
- Forward Algorithm
  - Forward probability

$$\alpha_t(i) = P(x_1, x_2, \cdots, x_t, z_t = s_i | \lambda)$$

$$\begin{aligned} P(\mathbf{x}|\lambda) &= P(x_1, x_2, \cdots, x_T | \lambda) \\ &= \sum_{i=1}^{N} P(x_1, x_2, \cdots, x_T, z_T = s_i | \lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned}$$

  - Recursive

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji} \right] b_i(x_{t+1})$$



Evaluation Problem

- Forward Algorithm
  - Derive the equation

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i(x_{t+1})$$

$$\begin{aligned}\alpha_{t+1}(i) &= p(x_1 \cdots, x_{t+1}, z_{t+1} = s_i) \\ &= p(x_{t+1} | z_{t+1} = s_i) p(x_1, \cdots, x_t, z_{t+1} = s_i)\end{aligned}$$

x_{t+1} is only determined by z_{t+1}

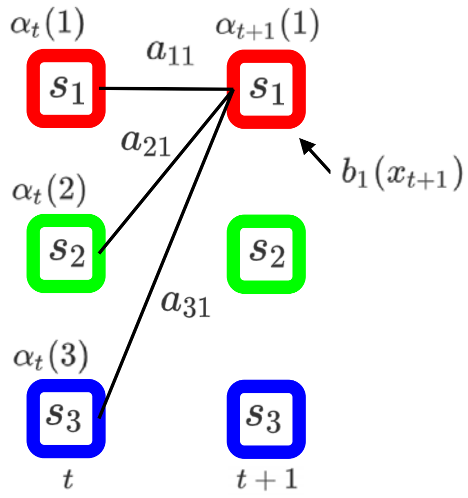
Evaluation Problem

- Forward Algorithm
  - Derive the equation

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji} \right] b_i(x_{t+1})$$

$$\begin{aligned}
\alpha_{t+1}(i) &= p(x_1 \cdots, x_{t+1}, z_{t+1} = s_i) \\
&= p(x_{t+1} | z_{t+1} = s_i) p(x_1, \cdots, x_t, z_{t+1} = s_i) \\
&= b_i(x_{t+1}) \sum_{j=1}^{N} p(x_1, \cdots, x_t, z_t = s_j, z_{t+1} = s_i) \\
&= b_i(x_{t+1}) \sum_{j=1}^{N} p(x_1, \cdots, x_t, z_t = s_j) p(z_{t+1} = s_i | z_t = s_j) \\
&= b_i(x_{t+1}) \sum_{j=1}^{N} \alpha_t(j) a_{ji}
\end{aligned}$$

z_{t+1} is only determined by z_t



Evaluation Problem

- Forward Algorithm
  - Algorithm Summary

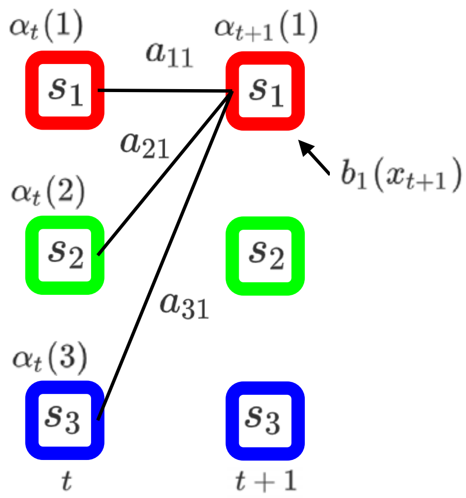
① Initialization $\alpha_1(i) = \pi_i b_i(x_1)$

② Forward recursion： 对$t = 1,2,\cdots,T-1$，

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i(x_{t+1})$$

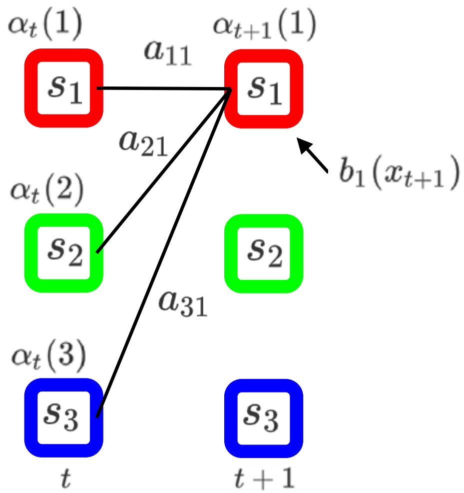
③ Termination

$$P(\mathbf{x}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$



Evaluation Problem

- Forward Algorithm
  - Computational complexity is $O(N^2T)$
  - Why? Every calculation directly refers to the calculation result at the previous moment, avoiding repeated calculations.



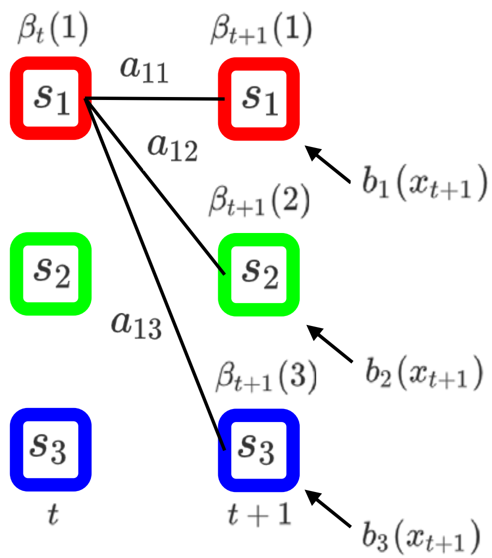
Evaluation Problem

- Backward Algorithm
  - Backward probability: $\beta_t(i) = P(x_{t+1}, x_{t+2}, \cdots, x_T | z_t = s_i, \lambda)$

$$P(\mathbf{x}|\lambda) = \sum_{i=1}^{N} b_i(x_1)\beta_1(i)\pi_i$$

  - Recursive:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1})\beta_{t+1}(j)$$



Evaluation Problem

- Backward Algorithm
  - Backward probability:

$$\beta_t(i) = P(x_{t+1}, x_{t+2}, \cdots, x_T | z_t = s_i, \lambda)$$

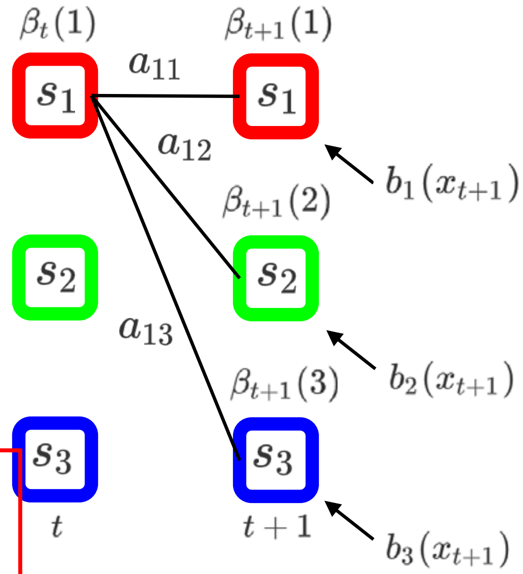
$$\begin{aligned}
P(\mathbf{x}|\lambda) &= P(x_1, x_2, \cdots, x_T) \\
&= \sum_{i=1}^{N} P(x_1, x_2, \cdots, x_T, z_1 = s_i) \\
&= \sum_{i=1}^{N} P(x_1 | z_1 = s_i) P(x_2, x_3, \cdots, x_T | z_1 = s_i) P(z_1 = s_i) \\
&= \sum_{i=1}^{N} b_i(x_1) \beta_1(i) \pi_i
\end{aligned}$$

Evaluation Problem

- Backward Algorithm
  - Derive the equation:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$

$$\begin{aligned}
\beta_t(i) &= P(x_{t+1}, \cdots, x_T | z_t = s_i) \\
&= \sum_{j=1}^{N} P(x_{t+1}, \cdots, x_T, z_{t+1} = s_j | z_t = s_i) \\
&= \sum_{j=1}^{N} P(x_{t+1}, \cdots, x_T | z_{t+1} = s_j) P(z_{t+1} = s_j | z_t = s_i) \\
&= \sum_{j=1}^{N} P(x_{t+2}, \cdots, x_T | z_{t+1} = s_j) P(x_{t+1} | z_{t+1} = s_j) P(z_{t+1} = s_j | z_t = s_i) \\
&= \sum_{j=1}^{N} \beta_{t+1}(j) b_j(x_{t+1}) a_{ij}
\end{aligned}$$



State transition probability

Evaluation Problem

- Backward Algorithm
  - Algorithm Summary

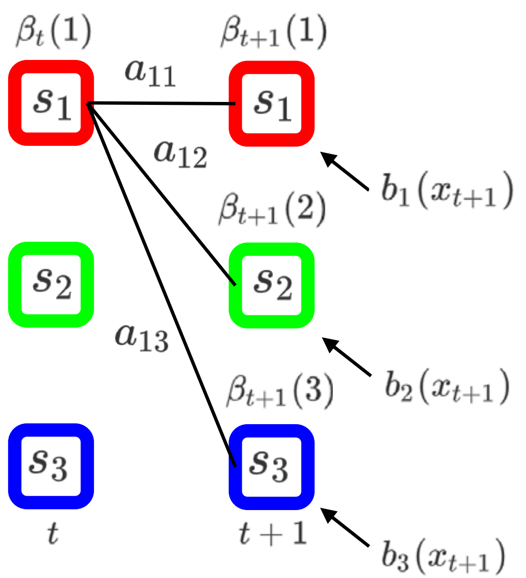
① Initialization $\beta_T(i) = 1$

② backward recursion： 对$t = T-1, T-2, \ldots, 1$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$

③ Termination

$$P(\mathbf{x}|\lambda) = \sum_{i=1}^{N} b_i(x_1)\beta_1(i)\pi_i$$



Evaluation Problem

- Incorporate Forward Recursion and Backward Recursion to estimate the probability of an observation sequence.

$$P(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji}\right] b_i(x_{t+1})$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$

Decoding Problem

- Decoding Problem
  - Given the observation sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$ and an HMM model $\lambda = (A, B, \pi)$, how to find the state sequence that best explains the observations $P(\mathbf{z}|\mathbf{x}, \lambda)$?
- Viterbi algorithm
  - define Viterbi variable $\delta_t(i)$ as the maximum probability of producing observation sequence $x_1, x_2, \cdots, x_t$ when moving along any hidden state sequence $z_1, z_2, \cdots, z_{t-1}$ and getting into $z_t = s_i$.

$$\delta_t(i) = \max_{z_1, z_2, \cdots, z_{t-1}} P(x_1, x_2, \cdots, x_t, z_1, z_2, \cdots, z_{t-1}, z_t = s_i | \lambda)$$

Decoding Problem

- Viterbi algorithm
  - find the global optimal results through find the stage optimal
  - general idea:
    - if best path ending in $z_t = s_j$ goes through $z_{t-1} = s_i$ then it should coincide with best path ending in $z_{t-1} = s_i$

$$\delta_{t+1}(i) = \max_{1 \leq j \leq N} [\delta_t(j) a_{ji}] b_i(x_{t+1})$$

    - to backtrack best path keep info that predecessor of s_j was s_i

Decoding Problem

- Viterbi algorithm
  - Algorithm Summary

① Initialization

$$\delta_1(i) = \pi_i b_i(x_1)$$
$$\Psi_1(i) = 0$$

② Forward recursion：对$t = 1,2,\cdots,T-1$，

$$\delta_{t+1}(i) = \max_{1\le j\le N}[\delta_t(j)a_{ji}]b_i(x_{t+1})$$
$$\Psi_{t+1}(i) = \arg\max_{1\le j\le N}[\delta_t(j)a_{ji}]$$

③ Termination

$$P^* = \max_{1\le i\le N}\delta_T(i)$$
$$z_T^* = \arg\max_{1\le i\le N}\delta_T(i)$$

④ Backtrack：对 $t = T-1,\ldots,2,1$,

$$z_t^* = \Psi_{t+1}(z_{t+1}^*)$$

Decoding Problem

- Example: <u>Initialization</u>

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.09409*0.061 = 0.00573949				
s_2: v	0				
s_3: vn	0				

$$\delta_1(1) = \pi_1 b_1(我们)$$

$$\delta_1(2) = \pi_2 b_2(我们)$$

$$\delta_1(3) = \pi_3 b_3(我们)$$

B

	我们	_要_	加强	合作	交流
r	06100	00000	00000	00000	00000
v	00000	01571	00440	00095	00020
vn	00000	00000	00051	01315	00363

π

r	09409
v	07744
vn	00633

A

	r	v	vn
r	01615	25140	02394
v	03836	16499	02563
vn	00204	06264	07944

Decoding Problem

- Example: Forward recursion

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	r->r r->要 0			
s_2: v	0	r->v v->要 2.26680813 1806e-05			
s_3: vn	0	r->vn vn->要 0			

$$\delta_2(1) = \max_i\,[\delta_1(i)a_{i1}]b_1(要)$$

$$\delta_2(2) = \max_i\,[\delta_1(i)a_{i2}]b_2(要)$$

$$\delta_2(3) = \max_i\,[\delta_1(i)a_{i3}]b_3(要)$$

B

	我们	_要_	加强	合作	交流
r	06100	00000	00000	00000	00000
v	00000	01571	00440	00095	00020
vn	00000	00000	00051	01315	00363

π

r	09409
v	07744
vn	00633

A

	r	v	vn
r	01615	25140	02394
v	03836	16499	02563
vn	00204	06264	07944

Decoding Problem

- Example: Forward recursion

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0		
s_2: v	0	2.26680813 1806e-05	v->v v->加强 8.672723739 895865e-15		
s_3: vn	0	0	0		

$$\delta_3(1) = \max_i [\delta_2(i) a_{i1}] b_1(\text{加强})$$

$$\delta_3(2) = \max_i [\delta_2(i) a_{i2}] b_2(\text{加强})$$

$$\delta_3(3) = \max_i [\delta_2(i) a_{i3}] b_3(\text{加强})$$

B

	我们	_要_	加强	合作	交流
r	06100	00000	00000	00000	00000
v	00000	01571	00440	00095	00020
vn	00000	00000	00051	01315	00363

π

r	09409
v	07744
vn	00633

A

	r	v	vn
r	01615	25140	02394
v	03836	16499	02563
vn	00204	06264	07944

Decoding Problem

- Example: Forward recursion

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	
s_2: v	0	2.26680813 1806e-05	8.672723739 895865e-15	v->v v->合作 3.553006399 833383e-09	
s_3: vn	0	0	0	v->vn vn->合作 7.639925452 991693e-09	

$$\delta_4(1) = \max_i\,[\delta_3(i)a_{i1}]b_1(合作)$$

$$\delta_4(2) = \max_i\,[\delta_3(i)a_{i2}]b_2(合作)$$

$$\delta_4(3) = \max_i\,[\delta_3(i)a_{i3}]b_3(合作)$$

B

	我们	_要_	加强	合作	交流
r	06100	00000	00000	00000	00000
v	00000	01571	00440	00095	00020
vn	00000	00000	00051	01315	00363

π

r	09409
v	07744
vn	00633

A

	r	v	vn
r	01615	25140	02394
v	03836	16499	02563
vn	00204	06264	07944

Decoding Problem

- Example: Forward recursion

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.26680813 1806e-05	8.672723739 895865e-15	3.553006399 833383e-09	1.17242105181 70199e-13
s_3: vn	0	0	0	7.639925452 991693e-09	2.20310391108 7946e-12

$$\delta_5(1) = \max_i [\delta_4(i)a_{i1}]b_1(交流)$$

$$\delta_5(2) = \max_i [\delta_4(i)a_{i2}]b_2(交流)$$

$$\delta_5(3) = \max_i [\delta_4(i)a_{i3}]b_3(交流)$$

B

	我们	_要_	加强	合作	交流
r	06100	00000	00000	00000	00000
v	00000	01571	00440	00095	00020
vn	00000	00000	00051	01315	00363

π

r	09409
v	07744
vn	00633

A

	r	v	vn
r	01615	25140	02394
v	03836	16499	02563
vn	00204	06264	07944

Decoding Problem

- Example: Termination

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.26680813 1806e-05	8.672723739 895865e-15	3.553006399 833383e-09	1.17242105181 70199e-13
s_3: vn	0	0	0	7.639925452 991693e-09	2.20310391108 7946e-12

√

Decoding Problem

- Example: Backtrack

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.266808131806e-05	8.672723739895865e-15	3.553006399833383e-09	1.1724210518170199e-13
s_3: vn	0	0	0	7.639925452991693e-09	2.203103911087946e-12 √

Decoding Problem

- Example: Backtrack

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.266808131806e-05	8.672723739895865e-15	3.553006399833383e-09	1.1724210518170199e-13
s_3: vn	0	0	0	7.639925452991693e-09	2.203103911087946e-12

√

Decoding Problem

- Example: <u>Backtrack</u>

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.266808131806e-05	8.672723739895865e-15	3.553006399833383e-09	1.1724210518170199e-13
s_3: vn	0	0	0	7.639925452991693e-09	2.203103911087946e-12

√

Decoding Problem

- Example: Backtrack

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.266808131806e-05	8.672723739895865e-15	3.553006399833383e-09	1.1724210518170199e-13
s_3: vn	0	0	0	7.639925452991693e-09	2.203103911087946e-12

√

Decoding Problem

- Example: Backtrack

$\delta_t(i)$	我们	要	加强	合作	交流
s_1: r	0.00573949	0	0	0	0
s_2: v	0	2.266808131806e-05	8.672723739895865e-15	3.553006399833383e-09	1.1724210518170199e-13
s_3: vn	0	0	0	7.639925452991693e-09	2.203103911087946e-12 √

我们/r 要/v 加强/v 合作/vn 交流/vn

Learning Problem

Learning. How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\mathbf{x}|\lambda)$?

- Given $\left\{(\mathbf{x}^{(n)}, \mathbf{z}^{n})\right\}_{n=1}^{N}$
  - transition probabilities: A_{ij} is the number of transitions from state s_i to state s_j
    $$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}}$$
  - Observation probabilities: B_{ij} is the number of observation o_k occurs in state s_j
    $$\hat{b}_j(k) = \frac{B_{jk}}{\sum_{k=1}^{M} B_{jk}}$$
  - initial state probabilities
    $$\hat{\pi}_i = \frac{\sum_{n=1}^{N} \mathbb{1}\left(z_1^{(n)} = s_i\right)}{N}$$

Learning Problem

Learning. How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\mathbf{x}|\lambda)$?

- Given $\left\{\mathbf{x}^{(n)}\right\}_{n=1}^{N}$ $\boxed{\text{Baum-Welch algorithm}}$
  - initialize the parameters
  - repeat
    - E step: estimate state $\hat{\mathbf{z}}^{(n)}$ for observation $\mathbf{x}^{(n)}$ by the forward-backward algorithm
    - M step: update parameters by $\mathbf{x}^{(n)}, \hat{\mathbf{z}}^{(n)}$

Learning Problem

- E step
  - not directly estimated $\mathbf{z}^{(n)}$
  - Instead, it estimates the probability of $z_t^{(n)} = s_i$, So, $\gamma_t(i) = P(z_t = s_i | \mathbf{x}, \lambda)$:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \underbrace{\alpha_t(j)}_{\text{Forward probability}} \underbrace{\beta(j)}_{\text{Backward probability}}}$$

Learning Problem

- E step:
  - One variable derivation:

$$\alpha_t(i) = P(x_1, x_2, \cdots, x_t, z_t = s_i|\lambda)$$

$$\beta_t(i) = P(x_{t+1}, x_{t+2}, \cdots, x_T|z_t = s_i, \lambda)$$

$$\begin{aligned}\gamma_t(i) &= p(z_t = s_i|\mathbf{x}) \\ &= \frac{p(\mathbf{x}|z_t = s_i)p(z_t = s_i)}{p(x)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta(j)}\end{aligned}$$

- If $\xi_t(i,j) = p\big(z_t = s_i, z_{t+1} = s_j\big|\mathbf{x}, \lambda\big)$, So

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)}$$

Learning Problem

- E step:
  - Two variables derivation:

$$\alpha_t(i) = P(x_1, x_2, \cdots, x_t, y_t = s_i | \lambda)$$
$$\beta_{t+1}(j) = P(x_{t+2}, x_{t+3}, \cdots, x_T | z_{t+1} = s_j, \lambda)$$

$$\begin{aligned}\xi_t(i,j) &= p(z_t = s_i, z_{t+1} = s_j | \mathbf{x}) \\ &= \frac{p(z_t = s_i, z_{t+1} = s_j, \mathbf{x})}{p(\mathbf{x})} \\ &= \frac{\alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}\end{aligned}$$

Learning Problem

- M step
  - update parameters

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$b_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot \mathbb{1}(x_t = o_k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\pi_i = \gamma_1(i)$$

Hidden Markov Model

- For Part-of-speech Tagging problem
  - Tagging Task is corresponding to Decoding Problem
  - Training Task is corresponding to Learning Problem
  - Case Study/Explaination is corresponding to Evaluation Problem

Condition Random Field

- HMM is a generative model that models the generation process of the observation sequence:

$$\begin{aligned}\hat{\mathbf{z}} &= \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}) \\ &= \arg\max_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\end{aligned}$$

- CRF is a discriminant model that directly models the mapping relationship from input to output:

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} F(\mathbf{z}|\mathbf{x})$$

Condition Random Field

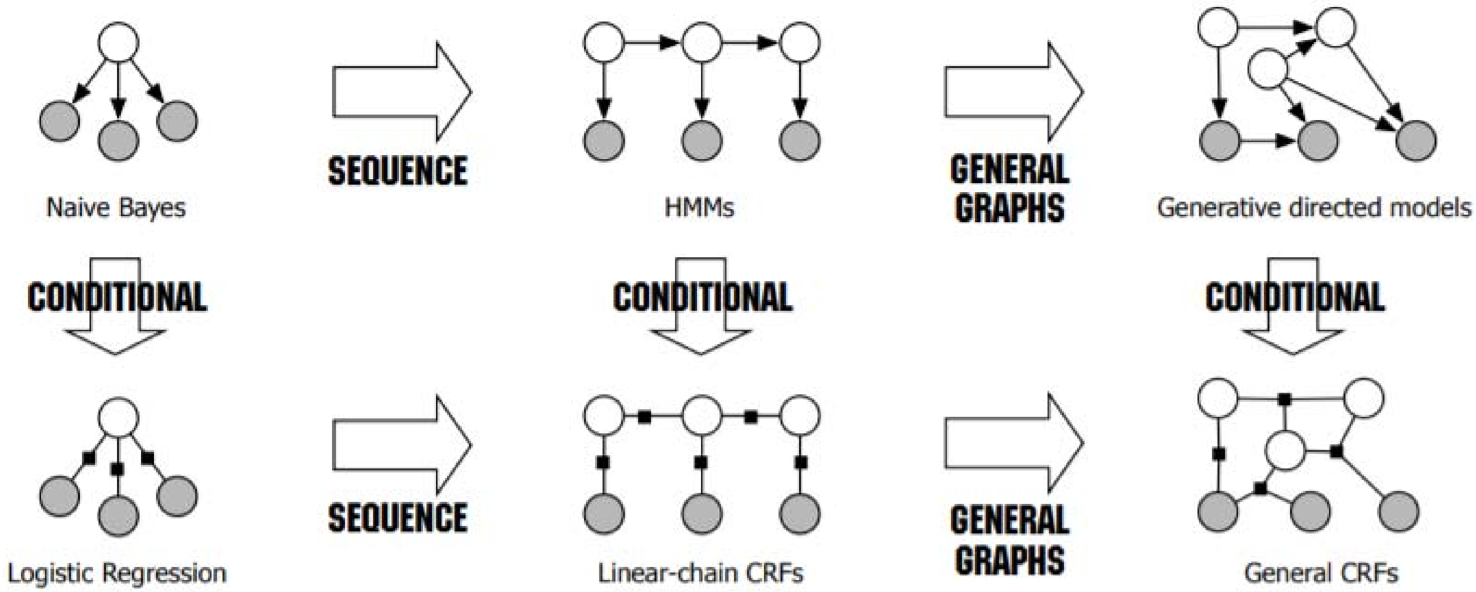
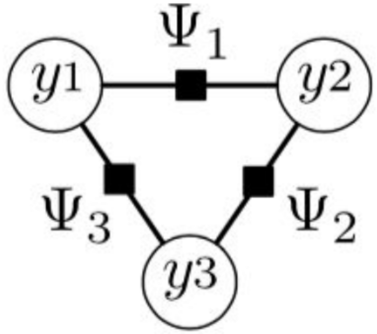


Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

Condition Random Field

- Undirected Graph Model:
  - calculate the probability distribution $P(y_1, y_2, \ldots, y_n)$:



$$p(y_1, y_2, y_3) = Z^{-1}\Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi_3(y_1, y_3) = Z^{-1}\prod_{a=1}^{A}\Psi_a y_a$$

$$Z = \sum_{Y}^{\Omega}\prod_{a=1}^{A}\Psi_a y_a$$

Condition Random Field

- CRF is a probabilistic undirected graph model:
  - Let's let the largest group set in CRF be C, so the conditional probability is:

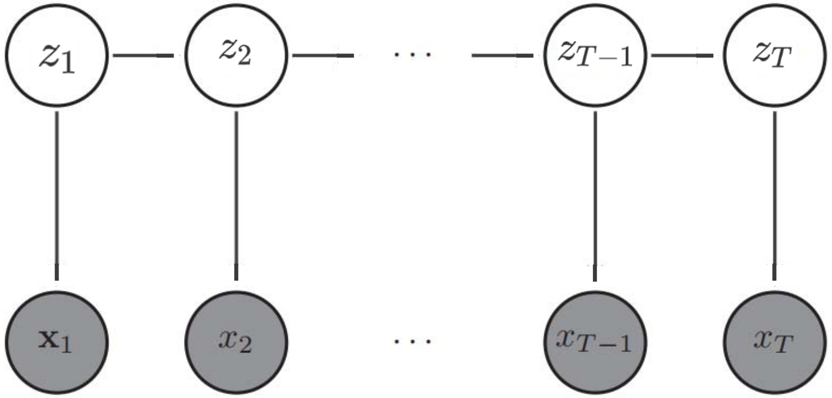
$$p(\mathbf{z}|\mathbf{x},\theta) = \frac{1}{Z(\mathbf{x},\theta)} \exp\left(\sum_{c\in\mathcal{C}} \theta_c^\top f_c(\mathbf{x},\mathbf{z_c})\right)$$

  - Among them, $Z(\mathbf{x},\theta)$ is the normalization factor, also known as the partition function:

$$Z(\mathbf{x},\theta) = \sum_{\mathbf{z}} \exp(\sum_{c\in\mathcal{C}} \theta_c^\top f_c(\mathbf{x},\mathbf{z_c}))$$

Condition Random Field

- Linear chain CRF



$$p(\mathbf{z}|\mathbf{x},\theta) = \frac{1}{Z(\mathbf{x},\theta)} \exp\left(\sum_{t=1}^{T} \theta_1^\top \underline{f_1(x_t, y_t)} + \sum_{t=2}^{T} \theta_2^\top \underline{f_2(y_{t-1}, y_t)} \right)$$

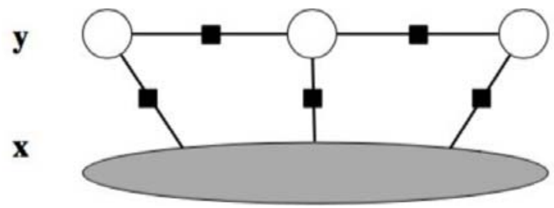
State characteristics

Transition characteristics, generally represented by a state transition matrix

Condition Random Field

- Linear chain conditional random field in general:
  - Let X, Y be random variables, $F = \{f_k(y, y', x)\}_{\{k=1\}}^{K}$ is a set of real-valued characteristic functions:

$$p(y,x) = Z^{-1} \prod_{t=1}^{T} \exp\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, X)\}$$



Condition Random Field

- Linear chain CRF
  - The calculation of the partition function is generally calculated by the dynamic programming method.
  - The parameters are generally learned through the maximum conditional log-likelihood function.
  - The decoding process is similar to HMM, and it is calculated quickly through the Viterbi algorithm.

HMM vs. CRF

- Some Comparisons between HMM and CRF:

	Speed	**Discrim vs. Generative**	**Normalization**
HMM	Very fast	Generative	Local
CRF	Kinda slow	Discriminative	Global

Open-Source Package

- SVM
  - LIBSVM -- A Library for Support Vector Machines
    https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- HMM
  - GHMM Library http://www.ghmm.org/
- CRF
  - CRF++
    https://drive.google.com/drive/folders/0B4y35FiV1wh7fngteFhHQUN2Y1B5eUJBNHZUemJYQV9VWlBUb3JlX0xBdWVZTWtSbVBneU0?usp=drive_web#list
  - Pocket CRF
  - FlexCRF

Reference

1. 统计学习方法(第2版).
2. Pattern Recognition and Machine Learning.
3. Black Box Machine Learning https://davidrosenberg.github.io/mlcourse/Archive/2017Fall/Lectures/01.black-box-ML.pdf
4. Logistic Regression https://web.stanford.edu/~jurafsky/slp3/5.pdf

The next lecture

- Lecture 4: Deep Learning Foundations