CS778 Fall 2006

Primer on Generative vs. Discriminative Learning

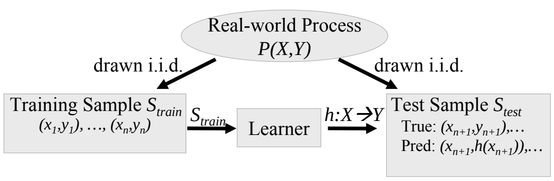
Thorsten Joachims

Cornell University
Department of Computer Science

Project

- **Project Topic**
  - Self-defined: comparison of methods, application of exiting method to new domain, explore modifications of existing method, etc.
  - Can be centered around a paper, but should envision some originality
  - I will help refine the topic
- **Project Proposal**
  - Outline general idea for project
  - Discuss resources and timeline
  - Due: September 19 (in class)

Supervised Learning



- **Learning Task: $P(X,Y) = P(X)\, P(Y|X)$**
  - Input Space: X (e.g. feature vectors, word sequence, etc.)
  - Output Space: Y (e.g. class label $1..k$)
  - Training Data: $S_{train}=((x_1,y_1), \ldots, (x_n,y_n)) \sim_{iid} P(X,Y)$
- **Goal: Find h: X → Y with low prediction error $Err_P(h)$**

Generalization Error and Sample Error

Definition: *The* **prediction error/generalization error/true error/expected loss/risk** $Err_P(h)$ *of a hypothesis* h *for a learning task* $P(X,Y)$ *is*

$$Err_P(h) = \sum_{\vec{x} \in X, y \in Y} \Delta(h(\vec{x}), y) P(X = \vec{x}, Y = y).$$

Definition: $\Delta(a,b)$ *is a loss function that measures the cost of making a wrong prediction. A commonly used loss function is the the 0/1-loss*

$$\Delta(a,b) = \begin{cases} 0 & if(a == b) \\ 1 & else \end{cases}$$

Definition: *The* **error on sample** S $Err_S(h)$ *of a hypothesis* h *is* $Err_S(h) = \frac{1}{n}\sum_{i=1}^{n} \Delta(h(\vec{x}_i), y_i)$.

Generative vs. Discriminative Models

Learning Task:
- Generator: Generate descriptions according to distribution $P(X)$.
- Teacher: Assigns a value to each description based on $P(Y|X)$.

Training Examples $(x_1,y_1), \ldots, (x_n,y_n) \sim P(X,Y)$

Discriminative Model
- Model $P(Y|X)$ with $P(Y|X,\omega)$
  - Find ω e.g. via MLE
  - Examples: Log. Reg., CRF
- Model discriminant functions
  - Find $h_\omega \in H$ with low train loss (e.g. Emp. Risk Min.)
  - Examples: SVM, Dec. Tree

Generative Model
- Model $P(X,Y)$ with distributions $P(Y,X|\omega)$
- Find ω that best matches $P(X,Y)$ on training data (e.g. MLE)
- Examples: naive Bayes, HMM

Generative Model: Model P(X,Y)

- **Bayes' Decision Rule:** Optimal Decision is

$$h(x) = \underset{y \in Y}{\text{argmin}} \left[\sum_{y' \in Y} \Delta(y',y) P(Y=y'|X=x) \right]$$

- **Equivalent Reformulations:** For 0/1-Loss Δ(y,y') = 1, if y ≠ y', 0 else

$$\begin{aligned} h(x) &= \underset{y \in Y}{\text{argmax}} [P(Y=y|X=x)] \\ &= \underset{y \in Y}{\text{argmax}} \left[\frac{P(X=x|Y=y)P(Y=y)}{P(X=x)} \right] \\ &= \underset{y \in Y}{\text{argmax}} [P(X=x|Y=y)P(Y=y)] \\ &= \underset{y \in Y}{\text{argmax}} [P(X=x, Y=y)] \\ &= \underset{y \in Y}{\text{argmax}} [P(Y=y|X=x)P(X=x)] \end{aligned}$$

- **Learning: maximum likelihood (or MAP, or Bayesian)**
  - Assume model class P(X,Y|ω) with parameters ω ∈ Ω
  - Find $\omega' = \underset{\omega \in \Omega}{\text{argmax}} \prod_{i=1}^{n} [P(Y=y_i, X=x_i|\omega)]$

Naïve Bayes' Classifier (Multivariate)

fever (3)	cough (2)	pukes (2)	flu?
high	yes	no	1
high	no	yes	1
low	yes	no	-1
low	yes	yes	1
high	no	yes	???

- **Input Space X: Feature Vector**
- **Output Space Y: {1,-1}**
- **Model:**
  - Prior class probabilities

    $$P(Y=+1) \quad P(Y=-1)$$

  - Class conditional model (one for each class)

    $$P(X=\vec{x}|Y=+1) = \prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=+1)$$

    $$P(X=\vec{x}|Y=-1) = \prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=-1)$$

- **Classification rule:**

$$h_{naive}(\vec{x}) = argmax_{y\in\{+1,-1\}} \left\{ P(Y=y) \prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=y) \right\}$$

Estimating the Parameters of Naïve Bayes

fever (3)	cough (2)	pukes (2)	flu?
high	yes	no	1
high	no	yes	1
low	yes	no	-1
low	yes	yes	1
high	no	yes	???

- **Count frequencies in training data**
  - n: number of training examples
  - n_+ / n_- : number of pos/neg examples
  - $\#(X^{(j)}=x^{(j)}, y)$: number of times feature $X^{(j)}$ takes value $x^{(j)}$ for examples in class y
  - $|X^{(j)}|$: number of values attribute of $X^{(j)}$
- **Estimating:** $\omega' = \operatorname{argmax}_{\omega\in\Omega} \prod_{i=1}^{n} \left[P(Y=y_i) \prod_{j=1}^{N} \left[P(X_i^{(j)}=x_i^{(j)}|Y=y_i) \right] \right]$
  - P(Y): Maximum Likelihood Estimate

    $$\hat{P}(Y=1) = \frac{n_+}{n} \quad \hat{P}(Y=-1) = \frac{n_-}{n}$$

  - P(X|Y): Maximum Likelihood Estimate

    $$\hat{P}(X^{(j)}=x^{(j)}|Y=y) = \frac{\#(X^{(j)}=x^{(j)},y)}{n_y}$$

  - P(X|Y): Smoothing with Laplace estimate

    $$\hat{P}(X^{(j)}=x^{(j)}|Y=y) = \frac{\#(X^{(j)}=x^{(j)},y)+1}{n_y+|X^{(j)}|}$$

Generative vs. Discriminative Models

Learning Task:
- Generator: Generate descriptions according to distribution *P(X)*.
- Teacher: Assigns a value to each description based on *P(Y|X)*.

Training Examples $(x_1,y_1),\dots,(x_n,y_n) \sim P(X,Y)$

Discriminative Model
- Model *P(Y|X)* with *P(Y|X,ω)*
  - Find *ω* e.g. via MLE
  - Examples: Log. Reg., CRF
- Model discriminant functions
  - Find $h_\omega \in H$ with low train loss (e.g. Emp. Risk Min.)
  - Examples: SVM, Dec. Tree

Generative Model
- Model *P(X,Y)* with distributions *P(Y,X|ω)*
- Find ω that best matches *P(X,Y)* on training data (e.g. MLE)
- Examples: naive Bayes, HMM

Discriminative Model: Model P(Y|X)

- **Bayes' Decision Rule:**
  - General: $h(x) = \operatorname{argmin}_{y\in Y} \left[\sum_{y'\in Y} \Delta(y',y) P(Y=y'|X=x) \right]$
  - Assume 0/1 Loss Δ(y,y') = 1, if y ≠ y', 0 else

    $$h(x) = \operatorname{argmax}_{y\in Y} [P(Y=y|X=x)]$$

- **Learning: maximum likelihood (or MAP, or Bayesian)**
  - Assume model class P(Y|X,ω) with parameters ω ∈ Ω
  - Find

    $$\omega' = \operatorname{argmax}_{\omega\in\Omega} \prod_{i=1}^{n} [P(Y=y_i|X=x_i,\omega)]$$

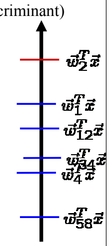
Logistic Regression/"Maximum Entropy"

- **Assume:**

$$P(Y=y|X=x,\omega=(w_1,\dots,w_k)) = \frac{e^{w_y^T x}}{\sum_{y'\in Y} e^{w_{y'}^T x}}$$

→Learn one weight vector w_y for each class $y \in Y$ (linear discriminant)

$$h(x) = argmax_{y\in Y} \left[w_y^T x \right]$$



- **Maximum Likelihood training:**

$$\begin{aligned} \omega' &= \operatorname{argmax}_{\omega\in\Omega} \left[\prod_{i=1}^{n} P(Y=y_i|X=x_i,\omega) \right] \\ &= \operatorname{argmax}_{\omega\in\Omega} \left[\sum_{i=1}^{n} \log(P(Y=y_i|X=x_i,\omega)) \right] \\ &= \operatorname{argmax}_{(w_1,\dots,w_k)} \left[\sum_{i=1}^{n} w_{y_i}^T x_i - \log\left(\sum_{y'\in Y} e^{w_{y'}^T x_i} \right) \right] \end{aligned}$$

Generative vs. Discriminative Models

Learning Task:
- Generator: Generate descriptions according to distribution *P(X)*.
- Teacher: Assigns a value to each description based on *P(Y|X)*.

Training Examples $(x_1,y_1),\dots,(x_n,y_n) \sim P(X,Y)$

Discriminative Model
- Model *P(Y|X)* with *P(Y|X,ω)*
  - Find *ω* e.g. via MLE
  - Examples: Log. Reg., CRF
- Model discriminant functions
  - Find $h_\omega \in H$ with low train loss (e.g. Emp. Risk Min.)
  - Examples: SVM, Dec. Tree

Generative Model
- Model *P(X,Y)* with distributions *P(Y,X|ω)*
- Find ω that best matches *P(X,Y)* on training data (e.g. MLE)
- Examples: naive Bayes, HMM

Discriminative Model: Model Discriminant Function *h* Directly

- **Discriminant Function: h_ω: X × Y → ℜ**

$$h(x) = \operatorname{argmin}_{y \in Y} \left[\sum_{y' \in Y} \Delta(y', y) P(Y = y' | X = x) \right]$$
$$= \operatorname{argmax}_{y \in Y} [h(x, y)] \quad (\text{e.g. } h(x, y) = [w_y^T x])$$

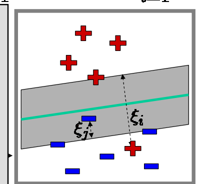
- **Consistency of Empirical Risk:**
  - Training Error (i.e. Empirical Risk): $Err_S(h) = \sum_{i=1}^{n} \Delta(y_i, h(x_i))$
  - For sufficiently "small" H_Ω and "large" S: Rule $h' \in H_\Omega$ with best $Err_S(h')$ has $Err_P(h')$ close to $min_{h \in H_\Omega}\{Err_P(h)\}$
- **Learning: Empirical Risk Minimization (ERM)**
  - Assume class H_Ω of discriminant functions $h_\omega: X \to Y$
  - Find
$$h'_\omega = \operatorname{argmin}_{h_\omega \in H_\Omega} \left[\sum_{i=1}^{n} \Delta(y_i, h_\omega(x_i)) \right]$$

Support Vector Machine

- **Training Examples:** $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ $\vec{x} \in \Re^N$ $y \in \{+1, -1\}$
- **Hypothesis Space:** $H_\Omega = \left\{ h(\vec{x}) = sgn\left[\vec{w}^T \vec{x} + b\right] : ||\vec{w}|| < \Omega \right\}$
- **Training Loss:** $Err_{S_{train}}(h) = \sum_{i=1}^{n} \Delta(y_i, h(\vec{x}_i)) \le \sum_{i=1}^{n} \xi_i$

Optimization Problem:

$$\min_{\vec{w}, \vec{\xi}, b} \quad \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{n} \xi_i$$
$$s.t. \quad y_1(\vec{w}^T \vec{x}_1 + b) \ge 1 - \xi_1$$
$$\ldots$$
$$y_n(\vec{w}^T \vec{x}_n + b) \ge 1 - \xi_n$$



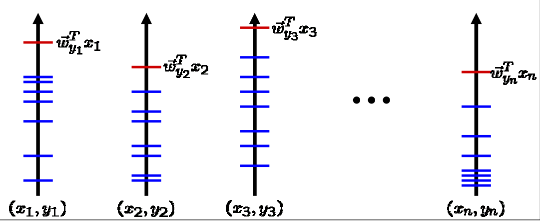
[Vapnik et al.]

[Crammer & Singer 02]

Training: Find $<\vec{w}_1, \ldots, \vec{w}_k>$ that solve

$$\min_{\vec{w}_1, \ldots, \vec{w}_n, \vec{\xi}} \quad \sum_{i=1}^{k} \vec{w}_i^2 + C \sum_{i=1}^{n} \xi_i$$
$$s.t. \quad \forall j \ne y_1 : \vec{w}_{y_1}^T \vec{x}_1 \ge \vec{w}_j^T \vec{x}_1 + 1 - \xi_1$$
$$\ldots$$
$$\forall j \ne y_n : \vec{w}_{y_n}^T \vec{x}_n \ge \vec{w}_j^T \vec{x}_n + 1 - \xi_n$$

- ..., $k\}$
-



So what about Complex Outputs?

- **Approach: view as multi-class classification task**
  - Every complex output y ∈ Y is one class

x The bear chased the cat	→	y Det→N→V→Det→N

- **Problem: Exponentially many classes!**
  - Generative Model: P(X,Y|ω)
  - Discriminative Model: P(Y|X,ω)
  - Discriminant Functions: h_ω: X × Y → ℜ
- **Challenges**
  - How to compactly represent model?
  - How to do efficient inference with model (i.e. $\operatorname{argmax}_{y \in Y} [h(x, y)]$)?
  - How to effectively estimate model from data? (e.g. compute $h'_\omega = \operatorname{argmin}_{h_\omega \in H_\Omega} \sum_{i=1}^{n} \Delta(y_i, h_\omega(x_i))$)

Predicting Sequences: Hidden Markov Model

- **Bayes rule:** $h(x) = \operatorname{argmax}_{y \in Y} [P(X = x | Y = y) P(Y = y)]$
- **Assumptions for compact representation**

$$P(Y = (y^{(1)}, \ldots, y^{(l)})) = \prod_{i=0}^{l} P(Y_c = y^{(i)} | Y_p = y^{(i-1)})$$
$$P(X = (x^{(1)}, \ldots, x^{(l)}) | Y = (y^{(1)}, \ldots, y^{(l)})) = \prod_{i=1}^{l} P(X_c = x^{(i)} | Y_c = y^{(i)})$$

y	Det	→ N	→ V	→ Det	→ N
	↓	↓	↓	↓	↓
x	The	bear	chased	the	cat

- **Prediction rule:**

$$h(x) = \operatorname{argmax}_{y \in Y} [P(X = x | Y = y) P(Y = y)]$$
$$= \operatorname{argmax}_{(y^{(1)}, \ldots, y^{(l)}) \in Y} \left[\prod_{i=1}^{l} P(Y_c = y^{(i)} | Y_p = y^{(i-1)}) P(X_c = x^{(i)} | Y_c = y^{(i)}) \right]$$

→Viterbi (dynamic prog) algorithm computes argmax efficiently

Predicting Sequences: HMM Training

- **Maximum Likelihood (or alternative estimator)**
  - Find $\omega' = \operatorname{argmax}_{\omega \in \Omega} \prod_{i=1}^{n} [P(Y = y_i, X = x_i | \omega)]$
$$= \operatorname{argmax}_{\omega \in \Omega} \left[\prod_{i=1}^{n} \prod_{j=1}^{l} P(Y_c = y_i^{(j)} | Y_p = y_i^{(j-1)}) P(X_c = x_i^{(j)} | Y_c = y_i^{(j)}) \right]$$
- **For Hidden Markov Model**
  - Closed-form solutions
$$P(Y_c = y_a | Y_p = y_b) = \frac{\#ofTimesStateAFollowsStateB}{\#ofTimesStateBOccurs}$$
$$P(X_c = x_a | Y_c = y_b) = \frac{\#ofTimesOutputAIsObservedInStateB}{\#ofTimesStateBOccurs}$$
  - Need for smoothing the estimates (e.g. max a posteriori)

Questions in this Class

- **Important applications for which conventional methods don't fit!**
  - Noun-phrase co-reference: two step approaches of pair-wise classification and clustering as postprocessing, e.g [Ng & Cardie, 2002]
  - Directly optimize complex loss functions (e.g. F1, AvgPrec)
- **Improve upon existing methods!**
  - Part-of-Speech Tagging: generative models like HMM
  - Is it possible to train discriminatively? Log Regression, SVMs, Boosting, etc.
  - SVM outperforms naïve Bayes for text classification [Joachims, 1998] [Dumais et al., 1998]
- **Mor**
  -
  -
- **Tra**
  -
  -
  - Support Vector Machines

Precision/Recall Break-Even Point	Naïve Bayes	Linear SVM
Reuters	72.1	87.5
WebKB	82.0	90.3
Ohsumed	62.4	71.6

Reading

- **Generative vs. Discriminative Training**
  - Tony Jebara, Machine Learning: Discriminative and Generative, Kluwer, 2004. (Chapters 1 and 2)
- **Hidden-Markov Models**
  - C. Manning and H. Schuetze, Foundations of Statistical NLP, MIT Press, 1999.
- **Logistic Regression**
  - T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer, 2001. (Section 4.4)
- **Support Vector Machines**
  - B. Schoelkopf and A. Smola, Learning with Kernels, MIT Press, 2001. (Chapters 1 and 7, available at http://www.learning-with-kernels.org/)