Part VII: Related Techniques

- Parameter Estimation
  - Basic Concepts
  - Maximum Likelihood Estimation
- Language Modeling
  - Feature Selection Methods
  - Clustering for Class-based Modeling
- Parameter Smoothing
  - Good-Turing
  - Back-off
- Adaptive Learning
  - Discrimination Enhancement
    - Parameter Tying
  - Robustness Enhancement
  - An NLP Example
- Bootstrapping

Parameter Learning via Estimation: Estimator Characteristics

- Statistic:
  - A *statistic* is any real or vector-valued function of the observation (e.g., $T(\mathbf{X})$).
    - E.g. X: Head/Tail of a fair coin; T: number of heads
- Estimators:
  - An estimator is a statistic calculated from sample data that provide either point estimates or interval estimates for some population parameter.
- Unbiased:
  - An estimator $\hat{\theta}$ is unbiased if its mean is equal to the population parameter being estimated θ , i.e., $E\left[\hat{\theta}\right] = \theta$
- Efficiency:
  - An estimator $\hat{\theta}$ of θ is said to be more efficient than any other unbiased estimator $\hat{\hat{\theta}}$ if $\text{var}(\hat{\hat{\theta}}) \leq \text{var}(\hat{\theta})$
  - An estimator is a minimum variance unbiased estimator if the variance of its sampling distribution is the smallest of all other unbiased estimators.

Parameter Learning via Estimation: Estimator Characteristics

- Consistency: $\lim_{n \to \infty} P\left(\left|\hat{\theta}(n) - \theta\right| \geq \varepsilon\right) = 0$
  - An estimator is said to be a consistent estimator if it converges to the parameter to be estimated in the probability sense. An estimator is a consistent one if the following conditions are met:
    - Asymptotically unbiased
    - Variance converges to 0 when sample size *n* gets large
  - Example:
    - Mean:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$E\left[\hat{\mu}\right] = \mu$$

$$Var\left[\hat{\mu}\right] = \frac{\sigma^2}{n}, \qquad \lim_{n \to \infty} Var\left[\hat{\mu}\right] \to 0$$

    - Probability estimator: relative frequencey
    - MLE estimator for mean and variance for Gaussian

Parameter Learning via Estimation: Estimator Characteristics

- Sufficiency:
  - A way for doing data compression in estimation process: by only keeping necessary information required for estimation
  - Definition: A statistic $T(X)$ is called *sufficient* for a parameter θ, if and only if, the conditional distribution of X given $T(X) = t$ does not involve θ (Bickel and Doksum, page 63).
    - Once the value of a sufficient statistic T is known, the sample $X = (X_1, \ldots, X_n)$ does not contain any further information about θ.
  - Example:
    - Sum of observations is a sufficient statistic for estimating mean
    - Event count is a sufficient statistic for estimating the associated probability

Parameter Estimation and Performance

- All probabilistic parameters are estimated from a finite set of samples.
  - Some criteria of a good estimator: unbias, consistent, efficient.
  - Estimation Criteria is not directly linked to Performance Criteria (e.g., error rate, joint precision-recall)
- Some frequently used estimation methods:
  - Maximum Likelihood Estimation (MLE)
  - Bayesian Estimation
  - Least Square Estimation

Maximum Likelihood Estimation

- To choose a set of parameters θ in a way that maximizes the likelihood function $L(\theta)$: $L(\theta) = f(x_1, x_2, \cdots, x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$
  - where $x_1, x_2, \ldots, x_n$ is a set of random samples from the distribution of a random variable ***X*** with density *f* and associated parameter θ.
- The ML estimation $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_k)$ is the set of estimated values that maximizes $L(\theta)$, or values that satisfies the simultaneous equations $\frac{\partial L(\theta)}{\partial \theta_i} = 0, \ i = 1, \cdots, k$
- Properties:
  - Maximum-likelihood estimates are (1) consistent, and (2) asymptotically efficient.
  - Let $\hat{\theta}_{ML}$ be an MLE of θ , then $g(\hat{\theta}_{ML})$ is an MLE of g(θ) , i.e., $[\hat{g}(\theta)]_{ML} = g(\hat{\theta}_{ML})$ if g(.) is a monotonic function.

Examples:

- The MLE for the "success" probability *p* of Bernoulli distribution is $\hat{p}_{ML} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
  where x_i is the outcome of the *i*-th Bernoulli trial.

- $\hat{p}_{ML}$ can be interpreted as the relative frequency of success over the *n* trials.

- The MLEs for the mean and variance of the normal density are:

$$\hat{\mu}_{MLE} = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n} \left(x_i - \overline{x} \right)^2$$

Bayesian Estimation

- To choose the parameters which maximizes the likelihood function $L(\pi(\theta), \theta)$: $L(\pi(\theta),\theta) = \pi(\theta) f(x_1, x_2, \cdots, x_n|\theta)$
- Where $\pi(\theta)$ is the prior probability density of θ before sampling.
- The Bayesian estimation $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_k)$ is the set of estimated values that satisfies the equations $\frac{\partial L(\theta)}{\partial \theta_i} = 0, \quad i = 1, \cdots, k.$
- The Bayesian estimation $\hat{\theta}$ of parameter θ is the expected value of the parameter taken with respect to the posterior distribution of θ given the outcome of the random sample x, i.e.,

$$\hat{\theta} = E[\theta|x]$$

Bayesian Decision vs. Maximum-Likelihood Decision

- Bayesian Decision Rule: Find the most probable source model (M) for a given observation (O) by choosing the one with the maximum conditional probability P(M|O): $\hat{M}_B = \arg\max_M P(M|O)$
  - It is the optimal classifier that minimizes the error rate.

- Maximum Likelihood Decision Rule: Find the model (M) which is most likely to generate the observation (O): $\hat{M}_M = \arg\max_M P(O|M)$.
  - It is the same as the Bayesian Decision if prior probability, i.e., P(M), is uniformly distributed (since P(M|O) = P(O|M) x P(M) / P(O))

$$P(M|O) = \frac{P(O|M) \cdot P(M)}{P(O)}$$

Feature Selection

- Goal: select the best subset of *d* features which optimizes a criterion function from a large set of features.
  - select the most discriminative features for processing
  - reduce the dimension of the feature space and the size of the parameter space.
  - reduce redundant information without degrading system performance
  - eliminate irrelevant or noisy features to reduce their effects on performance

Sequential Forward Selection [Devijver 82]

- Procedures:
  - Initially the feature set contains no feature.
  - Add one feature to the current feature set to form an enlarged feature set.
    - the one being selected is the one that maximizes some criterion function (e.g., accuracy rate) when used jointly with the current feature set.
  - Repeat until the feature set contains *d* features.

Class-Based Modeling

- Goal:
  - To reduce the number of parameters such that the parameters can be estimated more reliably.
  - To improve statistical language modeling:
    - to provide a partial solution in dealing with the estimation of parameters for unseen events.
- Model Form

$$
\begin{aligned}
&P(F_1,F_2,F_3|C_1)\\
&=P(F_1|F_2,F_3,C_1)\times P(F_2|F_3,C_1)\times P(F_3|C_1)\\
&=P(F_1,CF_1|F_2,CF_2,F_3,\cdots,C_1)\times P(F_2,CF_2|F_3,\cdots,C_1)\times P(F_3,CF_3|C_1)\\
&=P(F_1|CF_1,\cdots,C_1)\times P(CF_1|F_2,CF_2,F_3,\cdots,C_1)\times\cdots\\
&\approx P(F_1|CF_1,C_1)\times P(CF_1|CF_2,C_1)\times\cdots
\end{aligned}
$$

- Example

$$P(w_1,w_2,\cdots,w_n)\approx P(w_1|t_1)\times P(t_1|t_2)\times\cdots\times P(w_{n-1}|t_{n-1})\times P(t_{n-1}|t_n)\times P(w_n|t_n)\times P(t_n)$$

Clustering for forming Classes

- Goal:
  - To collect different tokens having similar behavior into various equivalent classes to improve the robustness of the adopted statistical language model:
- Applications:
  - Part-of-speech Tagging, Word sense disambiguation, Machine Translation, etc.
- Typical Procedures
  - Define feature space (e.g., frequent words occurrence vector, etc.)
  - Define similarity measure (e.g., distance, angle difference, etc.)
  - Cluster data iteratively according to the given criterion (e.g., minimum mean square error, maximin-distance, etc.)
  - Stop when desired criteria are matched.
- Commonly used Clustering Algorithms
  - Dynamic Clustering: find best K clusters (for given K)
  - Hierarchical Clustering: #clusters decrement/increment by one per iteration
    - agglomerative (Bottom-up/Clumping approach): n singletons => K clusters
    - divisive (Top-Down/Splitting approach): single cluster => K clusters

Dynamic Clustering

- Procedure:
  - Employs an iterative algorithm to optimize a clustering criterion function.
  - At each iteration, data points are assigned to clusters.
  - Then, the cluster representatives are updated to reflect any change in the data point assignment.
  - The new cluster models are used in the next iteration.
  - Continue until a stable partition is obtained.
  - The number of clusters is known beforehand.

- Example: K-means clustering
  1. Choose the number of classes, K
  2. Choose initial $\mu_1, \mu_2, \ldots, \mu_K, .$
  3. Classify each data x_i to one of the K classes.
  4. Recompute the estimates for μ_i using the results of 3.
  5. If the μ_i are consistent then STOP; otherwise go to 1, 2, or 3.

Hierarchical Clustering [Bottom-Up]

- Advantage: cheaper computation
- Procedure:
  - Initially, every point in the data set is considered as a separate cluster.
  - At any stage of a hierarchical clustering algorithm the two of the existing clusters which are most *similar* are merged to create a new cluster, thus reducing the number of potential clusters by one.
  - Terminate when the desired criteria are matched.
  - The number of clusters is unknown beforehand.

- Examples [Brown 92]
  - Friday Monday Thursday ...
  - people guys folks fellows ...
  - water gas coal liquid acid ...
  - man woman boy girl ...
  - head body hands eyes ...

Parameter Smoothing

- WHY: Not Enough Data to Train Parameters
  - IBM use 81 million parameters from 40,000 sentence pairs, about 800,000 words in each language.
  - In general, use training set with size over 10 x N_p to achieve good generalization, where N_p is the number of parameters.
  - When data is not enough, smoothing technique must be used to achieve robustness
- Parameter Smoothing Techniques:
  - Good-Turing estimate [Good 53]
  - Back-Off procedures [Katz 87]
  - Adding a flattening constant [Su 89]
  - Clipping with a floor value
  - Deleted Interpolation [Bahl 83]
    - Use the information from correlated and less restricted parameters, thus better estimated.

Smoothing Techniques

- Good-Turing Method [Good 53]:

$$C^*(x) = r^* \approx (r+1)\frac{N_{r+1}}{N_r} \qquad P^*_{GT}(X = x_i) \approx \frac{C^*(X = x_i)}{\sum_j C^*(X = x_j)}$$

  - where $C(x)=r$ is the number of occurrence of event $X=x$; $C^*(x)=r^*$ is the estimated frequency count that x would occur, and N_r is the number of events that occurs r times.

- Back-off Method [Katz 87]:
  - recursively reduce contextual window if the frequency is zero for larger window size

$$P_{BF}(c_i | c_{i-m}^{i-1}) = \begin{cases} P_{GT}(c_i | c_{i-m}^{i-1}) & C(c_{i-m}^{i}) > 0 \\ \alpha \cdot P_{BF}(c_i | c_{i-(m-1)}^{i-1}) & C(c_{i-m}^{i}) = 0, C(c_{i-m}^{i-1}) > 0 \\ P_{BF}(c_i | c_{i-(m-1)}^{i-1}) & otherwise \end{cases}$$

Adaptive Learning (Supervised)

- Directly minimizing error rate *via* gradient-descending search
- Basic Concepts:
  - Loss Function: the loss associated with an instance of recognition error
    - e.g., one recognition error $\Leftrightarrow$ loss: 1
    - (or approximate it within the range [0...1], depending on how bad it was mis-recognized)
  - Risk Function: expected value of the loss function
  - Minimizing risk function *via* gradient-descending
    - Approximate error function by an analytical loss function (such as arctan or sigmoid)
    - Find adjusting direction that might reduce risk: adjust the parameter vector in the *reverse* direction of the gradient of the Risk function ($-\nabla R$), so that the adjustment leads to least risk
    - Adjust parameter set iteratively: by adjusting a small step size of the parameter vector iteratively; the risk will then be reduced in *average*

Adaptive Learning (cont.)

- Adaptive learning is required to adjust the estimated parameters according to misjudged instances or unreliably recognized instances

- Why adaptive learning
  - Discrimination Issues
    - System recognition rate is related to the ranks of the candidate. Maximum likelihood approaches achieve the performance indirectly through the estimation process.
    - Hence, the criterion of maximizing likelihood is not equivalent to maximizing the recognition rate in the training corpus
  - Robustness Issues
    - Statistic variation between training corpora and unseen text is not considered in estimation
    - Hence, minimizing the error rate in the training corpora is not equivalent to maximizing the recognition rate in unseen text

Discrimination Enhancement [Su and Lee 1994]

- Find a discrimination function $g_j\left(O_j^{'},\lambda_j^{'}\right)$ which can well preserve the correct ranking orders.
  - Three ways to search for a good discrimination function, starting from a preliminary parameter $\hat{\lambda}$:
  - (1) change $\hat{\lambda}\rightarrow\lambda'$
  - (2) transform $O\rightarrow O'$
  - (3) adopt a good measuring function (e.g., $P(\cdot)\rightarrow g_j(\cdot)$).
  - Find a measuring function $g_j\left(O_j^{'},\lambda_j^{'}\right)$ for the transformed observation vector $\boldsymbol{O_j'}$ and adjusted parameter set λ_j' that maximize the probability of getting the correct ranks

Robustness Enhancement [Su and Lee 1994]

- Enlarge the inter-cluster distance and reduce intra-cluster variance to achieve maximum separation in a measure space
- Discard unreliable parameters
- Enlarge the margin between the correct analysis and the competing candidates in its confusing set

Learning Procedure

- **Initialization:** Initialize the parameters with maximum likelihood estimation (+ smoothing).
- **Calculate the miss-recognition distance:** Let the highest two scores and the correct score be

$$^{1}SC_z, ^{2}SC_z, ^{c}SC_z, \quad z \in \{lex, syn, sem\}$$

then the distance *d* for incorrect recognition is defined as follows:

$$d = \begin{cases} ^{1}SC_z - ^{c}SC_z & if\ error \\ ^{1}SC_z - ^{2}SC_z & if\ correct\ \& \dfrac{^{1}SC_z - ^{2}SC_z}{\max\left\{\left|^{1}SC_z\right|, \left|^{2}SC_z\right|\right\}} < \beta\% \end{cases}$$

Learning Procedure (cont.)

- Adjust the parameters:
  - Decide the amount of adjustment
    - A loss *l(d),* which is a function of the distance *d*, is defined for miss-recognition.
    - The amount of adjustment of parameters $\Delta\Lambda^{(t)}$ in the *t*-th iteration is determined such that the risk function $R=E[l(d)]$ (the expected loss) decrease:

$$R = E\left[l(d)\right] \approx \frac{1}{N}\sum_{i=1}^{N} l(d_i), \quad l(d) = \tan^{-1}\left(\frac{d}{d_0}\right), \quad l'(d) = \frac{d_0}{d_0^2 + d^2}$$

$$\Lambda^{(t+1)} = \Lambda^{(t)} + \Delta\Lambda^{(t)}$$

$$\Delta\Lambda^{(t)} = -\varepsilon(t) U \nabla R$$

    - $\varepsilon(t)$ is the learning rate function which is a mono-decreasing function of the iteration number *t*.
    - d_0 is a small positive constant.
    - *U* is a positive definite matrix controlling convergent speed of parameters.
  - The parameters are adjusted such that the score of the correct candidate is increased while the score of the top rank candidate is decreased.
  - The learning procedure would converge in mean, which means the *average risk* would decrease as the learning procedure proceeds.

Learning Procedure (cont.)

- **Robustness enhancement:** the learning process continuously proceeds until ${}^{c}SC \geq {}^{2}SC+\delta$; that is, the margin between the correct analysis (${}^{c}SC = {}^{1}SC$) and the second highest candidate exceeds a preset threshold δ.

Example of Learning

- Sentence: "Press the left button"
- Correct tag sequence: "v art *adj* n"
  - Score:
    - ^{c}SC =S(v|Press)+S(art|the)+S(**adj**|left)*+S(n|button)
      +S(v|@)+S(art|v)+S(**adj**|art)*+S(n|**adj**)*
- Tag sequence with the highest score: "v art *v* n"
  - Score:
    - ^{1}SC =S(v|Press)+S(art|the)+S(**v** |left)*+S(n|button)
      +S(v|@)+S(art|v)+S(**v** |art)*+S(n|**v**)*

Example of Learning (cont.)

- Parameters before learning

		Press	the	left	button	subtotal	total
candidate 1	@	v	art	v	n		
LS		0	0	-0.3	0	-0.3	-2.38
CS		-0.7	-0.52	-0.7	-0.16	-0.28	
candidate 2	@	v	art	n	n		
LS		0	0	-0.7	0	-0.7	-2.92
CS		-0.7	-0.52	-0.3	-0.7	-2.22	
candidate 3	@	v	art	adj	n		
LS		0	0	-0.52	0	-0.52	-2.42
CS		-0.7	-0.52	-0.52	-0.16	-1.9	

- Parameters after learning

		Press	the	left	button	subtotal	total
candidate 1	@	v	art	v	n		
LS		0	0	-0.35	0	-0.35	-2.51
CS		-0.7	-0.52	-0.74	-0.2	-2.16	
candidate 2	@	v	art	n	n		
LS		0	0	-0.7	0	-0.7	-2.92
CS		-0.7	-0.52	-0.3	-0.7	-2.22	
candidate 3	@	v	art	adj	n		
LS		0	0	-0.48	0	-0.48	-2.29
CS		-0.7	-0.52	-0.48	-0.11	-1.81	

  - LS: Lexical Score CS: Context Score @: beginning of sentence marker

Discrimination Enhancement: Parameter Tying

- Why Parameter Tying ?
  - Unseen (or rarely occurred) events could be smoothed before adaptive learning
  - BUT, they cannot be *adjusted* during adaptive learning (since they are not in annotated training corpus or seed corpus).
- Solution for reliable estimation: tied to other *highly correlated parameters*
  - so that adaptive learning process has more chance to adjust them

Discrimination Enhancement: Parameter Tying

- Example: (POS Tagging) [Lin 95]
  - 3-gram contextual probability (MLE): $$P(c_i|c_{i-1},c_{i-2}) = \frac{C(c_{i-2},c_{i-1},c_i)}{\sum_{c_i} C(c_{i-2},c_{i-1},c_i)} \equiv \frac{N_i}{D_i}$$
    - D_i (denominator) < Q_d (threshold) $\Rightarrow$ insufficient sample size $\Rightarrow$ unreliably estimated
    - N_i (numerator) < Q_n $\Rightarrow$ not well trainable (i.e., has no good chance to be adjusted by adaptive learning)
  - If a probability is both *unreliably estimated* (D_i<Q_d) and *not well trainable* (N_i< Q_n) $\Rightarrow$ tied to reliable 2-gram probability:
  - 3-grams with the same 2-gram context (c_i,c_{i-1}) are assigned with the same 2-gram contextual probability $p(c_i|c_{i-1})$

Parameters tied to P(CD | IN)		
Parameter	Ni	Di
P(CD | IN, PN)	4	372
P(CD | IN, PPS)	0	24
P(CD | IN, PPSS)	0	21
P(CD | IN, WPS)	3	27

  - Qd and Qn are quite robust (in the ranges Qd=415~1245, Qn=1~10)
  - Tied 3-gram has slightly larger number of parameters than 2-gram, and much smaller than pure 3-gram; and has the best test set performance
  - A good compromise between estimation errors and modeling errors

Bootstrapping

- Goal:
  - Estimate parameters with a small corpus
  - Automatic tagging of a large untagged corpus
- Why:
  - The performance of supervised learning is better than that of unsupervised learning.
  - However, annotating (e.g., tagging) the corpus is a tedious, boring and time consuming task.

Bootstrapping With a Seed Corpus

- Procedure
  - Use a small annotated corpus as a seed to estimate the initial parameter values.
  - Smoothed the parameter before going to the next stage of bootstrapping (for handling the cases that have not seen in previous stages)
  - Enlarge the training set by adding more un-annotated data
  - Use EM or Viterbi training algorithms to re-estimate the parameters, from the enlarged corpora (however, the annotation associated with the seed corpus must be remained untouched).
  - Incremental bootstrapping: repeat the above process stage by stage by adding more unannotated data each time
    - Avoiding the hints, brought in from previous iterations, been overriden by the perturbation resulted from un-supervised guessing; thus we will have a better initial starting parameter set in each stage.