Statistical Learning Models for Text and Graph Data
Text Categorization 1: Classification

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

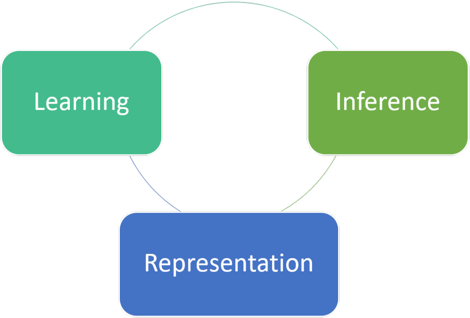
October 16, 2019

∗Contents are based on materials created by Noah Smith, Xiaojin (Jerry) Zhu, Eric Xing, Vivek Srikumar, Dan Roth

Reference Content

- Noah Smith. CSE 517: Natural Language Processing https://courses.cs.washington.edu/courses/cse517/16wi/
- Xiaojin (Jerry) Zhu. CS 769: Advanced Natural Language Processing. http://pages.cs.wisc.edu/~jerryzhu/cs769.html
- Eric Xing. 10715 Advanced Introduction to Machine Learning. https://www.cs.cmu.edu/~epxing/Class/10715/lectures/lecture1.pdf
- Vivek Srikumar. CS 6355 Structured Prediction. https://svivek.com/teaching/structured-prediction/spring2018/
- Dan Roth. CS546: Machine Learning and Natural Language . http://l2r.cs.uiuc.edu/~danr/Teaching/CS546-16/

Course Organization



- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

Text Classification

- Input: a piece of text $\mathbf{x} \in \mathcal{V}$, usually a document, e.g., a row vector of $\mathbf{X}$
- Output: a label from a finite set
- Standard line of attack:
  - Human experts label some data
  - Feed the data to a supervised machine learning algorithm that constructs an automatic classifier $f\colon \mathbf{x} \rightarrow \mathcal{L}$
  - Apply f to as much data as you want!
- Note: we assume the texts are segmented already, even the new ones

Text Classification: Examples

- Library-like subjects (e.g., the Dewey decimal system)
- News stories: politics vs. sports vs. business vs. technology ...
- Reviews of films, restaurants, products: positive vs. negative
- Author attributes: identity, political stance, gender, age, ...
- Email, arXiv submissions, etc.: spam vs. not
- What is the language of an article?

Closely related: relevance to a query

Features in Text Classification

Example (Running Example)

$\mathbf{x}$ = "The vodka was great, but don't touch the hamburgers."

- A different representation of the text sequence r.v. $\mathbf{X}$: feature vector
- For $j \in \{1, 2, \ldots, d\}$, let x^j be a discrete random variable taking a value in $\mathcal{F}$
  - Often, these are term (word and perhaps n-gram) frequencies

    e.g., $x^{\text{hamburgers}} = 1,\ x^{\text{the}} = 2,\ x^{\text{delicious}} = 1,\ x^{\text{don't touch}} = 1,$
  - Can also be word "presence" features

    e.g., $x^{\text{hamburgers}} = 1,\ x^{\text{the}} = 1,\ x^{\text{delicious}} = 1,\ x^{\text{don't touch}} = 1,$
  - Transformations on word frequencies: logarithm, idf weighting

$$\text{idf}(v) = \log \frac{N}{|i \in \{1, \ldots, N\}, c_{\mathbf{x}_i}(v) > 0|}$$

  - Disjunctions of terms
    - Clusters
    - Task-specific lexicons

Overview

Probabilistic Classification

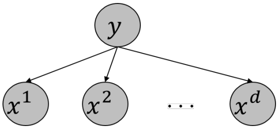
- Classification rule

$$\begin{aligned}\hat{y}(x) &= \arg\max_{y\in\mathcal{Y}} P(y|f(\mathbf{x})) \\ &= \arg\max_{y\in\mathcal{Y}} \frac{P(y,f(\mathbf{x}))}{P(f(\mathbf{x}))} \\ &= \arg\max_{y\in\mathcal{Y}} P(y,f(\mathbf{x}))\end{aligned}$$

Naive Bayes Classifier

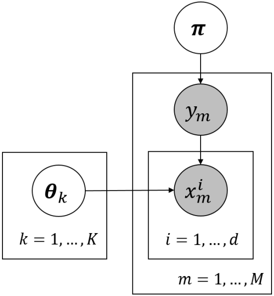
$$P(\mathbf{x}, y) = P(y) \prod_{j=1}^{d} P(X^j = x^j | y) = \pi_y \prod_{j=1}^{d} (\theta^j_{*|y})^{x^j}$$

- Parameters:
  - $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)^T$ is the "class prior" (sums to one): $\pi_k = P(y = k)$
  - For each feature j and label y, a distribution over values $\boldsymbol{\theta}_{*|y=y_k} = \boldsymbol{\theta}_k$ (sums to one for each y)
  - $K + K \times d$ parameters



Conditional independence assumption: given label observed, all the features are conditionally independent

Naive Bayes Classifier: A Generative View



Both y_m and $\mathbf{x}_m = (x_m^1, \ldots, x_m^d)^T$ are observed variables; $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$ are parameters

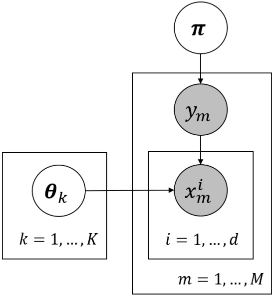
Naive Bayes from Class Conditional Unigram Model

- For $m = 1, \ldots, M$
  - Choose $y_m \sim Multinomial(y_m|1, \boldsymbol{\pi})$
  - Choose $N_m = \sum_j^d x_m^j \sim Poisson(\xi)$
  - For $n = 1, \ldots, N_m$
    - Choose $v \sim Multinomial(v|1, \boldsymbol{\theta}_{*|y_m}) = \prod_{j=1}^d (\theta^j_{*|y_m})^{v=j}$

Alternative views

- Choose $\mathbf{x}_m \sim Multinomial(\mathbf{X}|N_m, \boldsymbol{\theta}_{*|y_m}) = \begin{pmatrix} N_m \\ \mathbf{x}_m \end{pmatrix} \prod_{j=1}^d (\theta^j_{*|y_m})^{x_m^j}$
- Choose $x_m^d \sim Binomial(X|N_m, \theta^j_{*|y_m}) = \begin{pmatrix} N_m \\ x_m^j \end{pmatrix} (\theta^j_{*|y_m})^{x_m^j} (1 - \theta^j_{*|y_m})^{N_m - x_m^j}$

Parameter Estimation (based on Multinomial)



Both y_m and $\mathbf{x}_m = x_m^1, \dots, \mathbf{x}_m^d$ are observed variables; $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$ are parameters

Maximum likelihood of the training set:

$$
\begin{aligned}
\mathcal{J} &= \log \prod_{m=1}^{M} P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\
&= \sum_{m=1}^{M} \log P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\
&= \sum_{m=1}^{M} \log P(y_m|\boldsymbol{\pi}) P(\mathbf{x}_m|y_m, \boldsymbol{\theta}_{*|y_m})
\end{aligned}
$$

We can formulate a constrained optimization problem

$$
\begin{aligned}
\max \quad & \mathcal{J} \\
s.t. \quad & \sum_{k=1}^{K} \pi_k = 1 \\
& \sum_{j=1}^{d} \theta_k^j = 1 (k = 1, \dots, K)
\end{aligned}
$$

It's easy to solve with Lagrange multiplier and arrive at:

$$
\begin{aligned}
\pi_k &= \frac{|\{y_m = k\}|}{M} \\
\theta_k^j &= \frac{\sum_{m, y_m = k} x_m^j}{\sum_{m, y_m = k} \sum_{j=1}^{d} x_m^j}
\end{aligned}
$$

Maximum Likelihood Estimation: $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{W}|\boldsymbol{\theta})$

$$P(\mathcal{W}|\boldsymbol{\theta}) = \prod_i^d \theta_i^{u_i}$$

(log likelihood)

$$\Rightarrow \log P(\mathcal{W}|\boldsymbol{\theta}) = \sum_i^d u_i \log \theta_i$$

(Lagrange multiplier to make $\boldsymbol{\theta}$ be a distribution)

$$\Rightarrow L(\mathcal{W}, \boldsymbol{\theta}) = \log P(\mathcal{W}|\boldsymbol{\theta}) = \sum_i^d u_i \log \theta_i + \lambda(\sum_i \theta_i - 1)$$

(Set partial derivatives to zero)

$$\Rightarrow \frac{\partial L}{\partial \theta_i} = \frac{u_i}{\theta_i} + \lambda$$

Since $\sum_i^d \theta_i = 1$, we have $\lambda = -\sum_i^d u_i$

$$\Rightarrow \theta_i = \frac{u_i}{\sum_i^d u_i} = \frac{u_i}{N} \; (Maximum\ Likelihood\ Estimation\ , MLE)$$

Naive Bayes as a Linear Classifier

- Given

$$P(\mathbf{x}, y) = P(y)\prod_{j=1}^{d} P(X^j = x^j|y) = \pi_y \prod_{j=1}^{d}(\theta^j_{*|y})^{x^j}$$

- Consider a binary classification problem where $y = \{1, -1\}$, the prediction probability of the first class is

$$P(y = 1|\mathbf{x}) = \frac{\exp(\log \boldsymbol{\theta}_1^\top \mathbf{x} + \log \pi_1)}{\exp(\log \boldsymbol{\theta}_1^\top \mathbf{x} + \log \pi_1) + \exp(\log \boldsymbol{\theta}_{-1}^\top \mathbf{x} + \log \pi_{-1})}$$

- Classification rule with arg max can equivalently be expressed with log odds ratio:

$$\begin{aligned} f(\mathbf{x}) &= \log \tfrac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})} \\ &= \log P(y = 1|\mathbf{x}) - \log P(y = -1|\mathbf{x}) \\ &= (\log \boldsymbol{\theta}_1 - \log \boldsymbol{\theta}_{-1})^\top \mathbf{x} + (\log \pi_1 - \log \pi_{-1}) \end{aligned}$$

Naive Bayes: The Predictive Distribution

- Classification rule with arg max can equivalently be expressed with log odds ratio:

$$f(\mathbf{x}) = (\log \boldsymbol{\theta}_1 - \log \boldsymbol{\theta}_{-1})^\top \mathbf{x} + (\log \pi_1 - \log \pi_{-1})$$

- The decision rule is to classify $\mathbf{x}$ with $y = 1$ if $f(\mathbf{x}) > 0$, and $y = -1$ otherwise
- The Naive Bayes classifier induces a linear decision boundary in feature space $\mathcal{X}$; The boundary takes the form of a hyperplane, defined by $f(\mathbf{x}) = 0$

Naive Bayes: Remarks

- Estimation by (smoothed) relative frequency estimation: easy!
- For continuous or integer-valued features, use different distributions
- The bag of words version equates to building a conditional language model for each label

Generative vs. Discriminative Classification

- Naive Bayes is the prototypical generative classifier
  - It describes a probabilistic process: "generative story" for $\mathbf{x}$ and y
  - Models $P(\mathbf{x}, y) = P(\mathbf{x}|y)P(y)$ to interpret the data generation for each class
  - Assumes conditional independence on the features given class label
  - But why model $\mathbf{x}$? It's always observed? What if our goal is just classification $P(y|\mathbf{x})$?
- Discriminative models instead:
  - seek to optimize a performance measure, like accuracy, or a computationally convenient surrogate
  - do not worry about $P(\mathbf{X})$
  - directly model $P(y|\mathbf{x})$
  - tend to perform better when you have reasonable amounts of data

Discriminative Text Classifiers

- (Multinomial) logistic regression (also known as "max ent" and "log-linear" model)
- Support vector machines
- Neural networks

Logistic Regression

- Consider binary classification with $y \in \{-1, 1\}$, find a parameter vector to map **w**:

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

- Linear decision rule:

$$\begin{aligned} f(\mathbf{x}) &= \log \frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})} \\ &= \log \frac{\frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})}}{\frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1+\exp(-\mathbf{w}^\top \mathbf{x})}} \\ &= \log \exp(\mathbf{w}^\top \mathbf{x}) \\ &= \mathbf{w}^\top \mathbf{x} \end{aligned}$$

Theoretic Analysis (Ng and Jordan (2001))

Theorem (Compare Two Models)

Let h_D and h_G be any generative-discriminative pair of classifier, and $h_{D,\infty}$ and $h_{G,\infty}$ be their asymptotic/population versions. Then for $\varepsilon(h_{D,\infty}) \leq \varepsilon(h_{G,\infty}) + \epsilon_0$ to hold with high probability, it suffices to pick $m = \Omega(\log d)$, where d is dimensionality and m is number of training examples.

- When model assumption correct
  - NB and LR produce identical classifiers
- When model assumption incorrect
  - LR is less biased: does not assume conditional independence
  - Therefore expect to outperform NB

Results on UCI datasets (Ng and Jordan (2001))

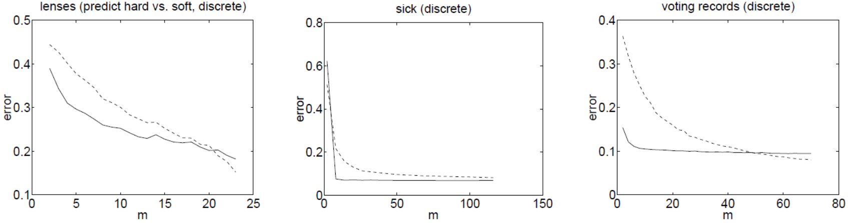
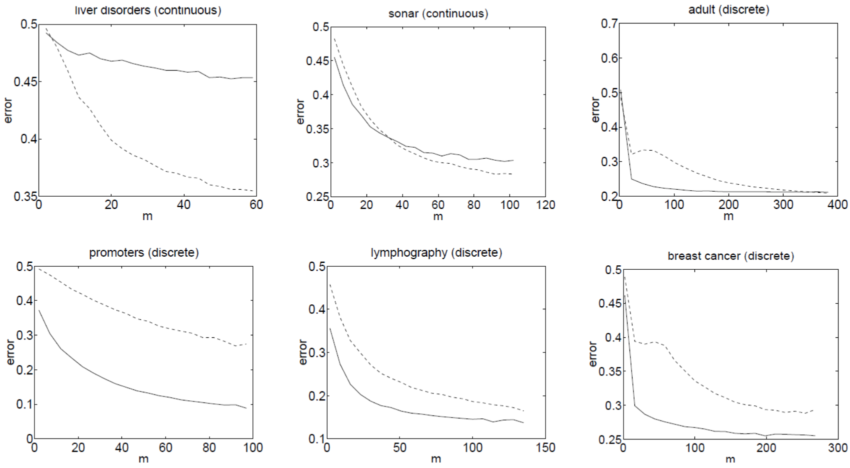
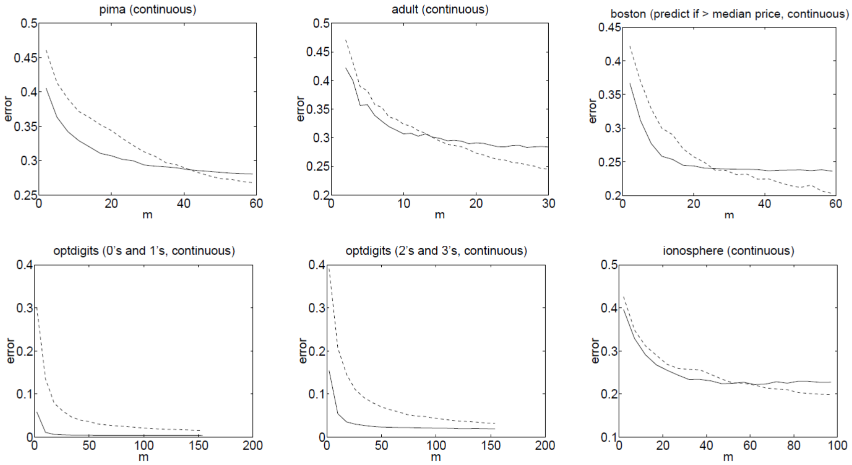


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. m (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

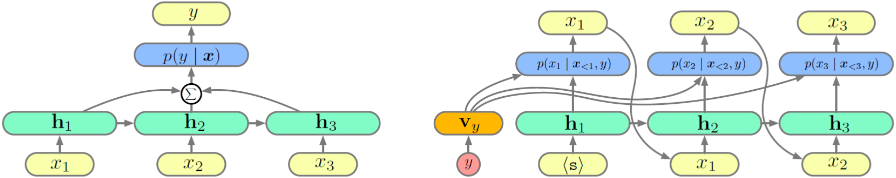
Results on UCI datasets (Ng and Jordan (2001))



Results on UCI datasets (Ng and Jordan (2001))



Generative vs. Discriminative Neural Network Text Classifiers
(Yogatama et al. (2017))



(a) Discriminative (b) Generative

Figure 1: Illustrations of our discriminative (left) and generative (right) LSTM models.

Neural Network Text Classifiers Results

(Yogatama et al. (2017))

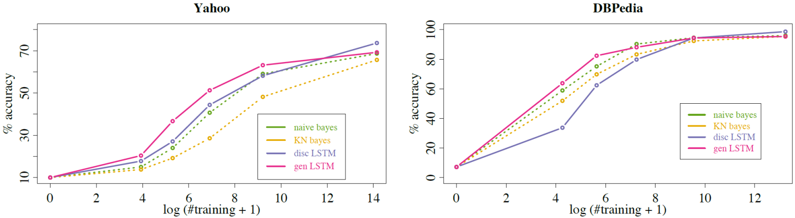
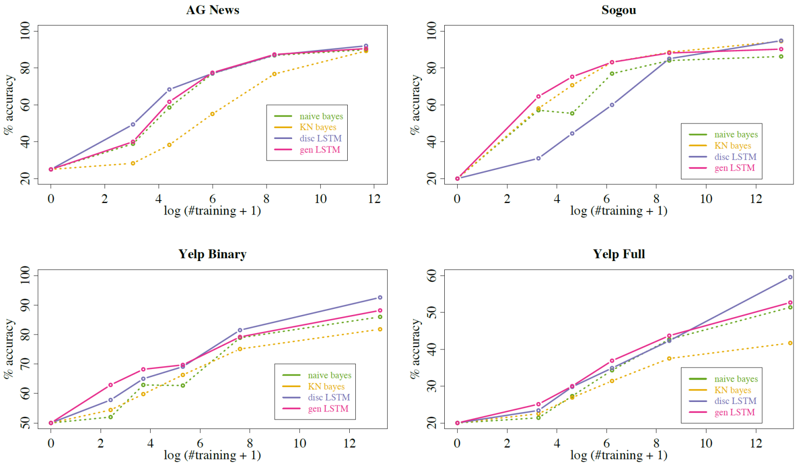


Figure 2: Accuracies of generative and discriminative models with varying training size.

Neural Network Text Classifiers Results

(Yogatama et al. (2017))



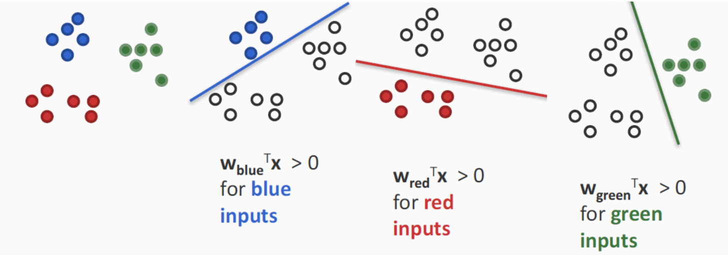
Overview

Binary to Multi-class

- Can we use a binary classifier to construct a multi-class classifier
  - Decompose the prediction into multiple binary decisions
    - One-vs-all
    - All-vs-all

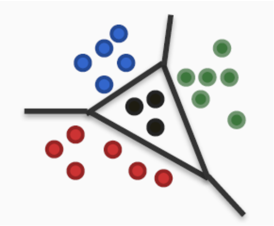
One-vs-all Classification

- Assumption: Each class individually separable from all the others
- Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_K$ using any binary classification algorithm we have seen
- Prediction: "Winner Takes All": $label = \arg\max_i \mathbf{w}_i^\top \mathbf{x}$



One-vs-all Classification

- Easy to learn
  - Use any binary classifier learning algorithm
- Problems
  - No theoretical justification
  - Calibration issues: We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  - Might not always work: Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized



All-vs-all Classification

Sometimes called one-vs-one

- Assumption: Every pair of classes is separable
- Train $\frac{K(K-1)}{2}$ classifiers to separate every pair of labels from each other
- Prediction: More complex, each label get K-1 votes
    - How to combine the votes? e.g.,
        - Majority: Pick the label with maximum votes

All-vs-all Classification

- Every pair of labels is linearly separable here
  - When a pair of labels is considered, all others are ignored
- Problems
  - $O(K^2)$ weight vectors to train and store
  - Size of training set for a pair of labels could be very small, leading to overfitting of the binary classifiers
  - Prediction is often ad-hoc and might be unstable. E.g., What if two classes get the same number of votes?

Training a Single Classifier

- Rewrite input features and weight vector
  - Define a feature vector for label i being associated to input $\mathbf{x}$
  - Stack all weight vectors into an nK-dimensional vector

$$\phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK \times 1} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK \times 1}$$

This is called the Kesler construction

Let Us Examine One-vs-all Again

- For an example with label i, we want $\mathbf{w}_i^\top \mathbf{x} > \mathbf{w}_j^\top \mathbf{x}$ for all j
- This is equivalent to

$$\mathbf{w}^\top \phi(\mathbf{x}, i) > \mathbf{w}^\top \phi(\mathbf{x}, j)$$

or

$$\mathbf{w}^\top [\phi(\mathbf{x}, i) - \phi(\mathbf{x}, j)] > 0$$

Notes

- The number of weights is still same as one-vs-all, much less than all-vs-all K(K-1)/2
- Still account for all pairwise label preferences
- Come with theoretical guarantees for generalization
- Important idea that is applicable when we move to arbitrary structures

Linear Models for Classification

- "Linear" decision rule

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \phi(\mathbf{x}, y)$$

  where $\phi : \mathcal{V} \times \mathcal{Y} \to \mathbb{R}^d$

- Parameters: $\mathbf{w} \in \mathbb{R}^d$
- What does this remind you of?

MLE for Multinomial Logistic Regression

- When we discussed log-linear language models, we transformed the score into a probability distribution. Here, that would be

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \phi(\mathbf{x}, y))}{\sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}$$

- MLE can be rewritten as a maximization problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}} - \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}}$$

- Recall from language models:
  - Be wise and regularize!
  - Solve with batch or stochastic gradient methods
  - w_i has an interpretation

Log Loss for $(\mathbf{x}, y)$

- Another view is to minimize the negated log-likelihood, which is known as "log loss":

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

Compare Loss

- For an example with label i, we want for all j

$$\mathbf{w}^\top \phi(\mathbf{x}, i) > \mathbf{w}^\top \phi(\mathbf{x}, j)$$

- Average log-loss

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

- Hinge loss

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\max_{y'}(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

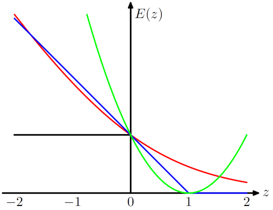
  - The function can be not differentiable
  - But it's still sub-differentiable. Solution: (stochastic) sub-gradient descent!

Hinge Loss for $(\mathbf{x}, y)$

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\max_{y'}(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

In binary case:

$$\Rightarrow \min_{\mathbf{w}} \sum_{\mathbf{x},y} \max\{0, -y\mathbf{w}^\top \mathbf{x}\}$$



Any thoughts about negative sampling?

Minimizing Hinge Loss: Perceptron

$$\min_{\mathbf{w}} \sum_{m=1}^{M} \max_{y'}(\mathbf{w}^\top \phi(\mathbf{x}_m, y')) - \mathbf{w}^\top \phi(\mathbf{x}_m, y_m)$$

- Stochastic subgradient descent on the above is called the perceptron algorithm
  - For $t = 1, \ldots, T$
    - Pick i_t randomly from $\{1, \ldots, n\}$
    - $\hat{y}_{i_t} = \arg\max_{y'} \mathbf{w}^\top \phi(\mathbf{x}, y')$
    - $\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\mathbf{w}^\top \phi(\mathbf{x}_{i_t}, \hat{y}_{i_t}) - \mathbf{w}^\top \phi(\mathbf{x}_{i_t}, y_{i_t}) \right)$

Error Costs

- Suppose that not all mistakes are equally bad
- E.g., false positives vs. false negatives in spam detection
- Let $\text{cost}(y', y)$ quantify the "badness" of substituting y' for correct label y
- Intuition: estimate the scoring function so that $\text{score}(y) - \text{score}(y') \propto \text{cost}(y', y)$

$$\left(\max_{y'}(\mathbf{w}^\top \phi(\mathbf{x}, y')) + \text{cost}(y, y')\right) - \mathbf{w}^\top \phi(\mathbf{x}, y)$$

General Remarks

- Text classification: many problems, all solved with supervised learners
  - Lexicon features can provide problem-specific guidance
- Naive Bayes, log-linear, and linear SVM are all linear methods that tend to work reasonably well, with good features and smoothing/regularization
- Rumor: random forests are widely used in industry when performance matters more than interpretability
- Lots of papers about neural networks, though with hyper-parameter tuning applied fairly to linear models, the advantage is not clear (Yogatama et al. (2015))
- Lots of work on feature design

Further Reading

- Michael Collins. The naive Bayes model, maximum-likelihood estimation, and the EM algorithm (Collins (2011))
- Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research (Crammer and Singer (2001))
- Daniel Jurafsky and James H. Martin. Logistic regression (Jurafsky and Martin (2017)) `https://web.stanford.edu/~jurafsky/slp3/7.pdf`
- Daniel Jurafsky and James H. Martin.Naive Bayes and sentiment classification (Jurafsky and Martin (2017)) `https://web.stanford.edu/~jurafsky/slp3/6.pdf`

References I

Collins, M. (2011). The naive bayes model, maximum-likelihood estimation, and the EM algorithm. Technical report, Columbia University.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Jurafsky, D. and Martin, J. H. (2017). *Speech and Language Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848.

Yogatama, D., Dyer, C., Ling, W., and Blunsom, P. (2017). Generative and discriminative text classification with recurrent neural networks. *CoRR*, abs/1703.01898.

Yogatama, D., Kong, L., and Smith, N. A. (2015). Bayesian optimization of text representations. In *EMNLP*, pages 2100–2105.