Natural Language Processing (CSE 517): Text Classification (II)

Noah Smith

University of Washington
nasmith@cs.washington.edu

February 1, 2016

Quick Review: Text Classification

Input: a piece of text $\boldsymbol{x} \in \mathcal{V}^\dagger$, usually a document (r.v. $\boldsymbol{X}$)
Output: a label from a finite set $\mathcal{L}$ (r.v. L)

Standard line of attack:

1. Human experts label some data.
2. Feed the data to a supervised machine learning algorithm that constructs an automatic classifier $\text{classify} : \mathcal{V}^\dagger \rightarrow \mathcal{L}$
3. Apply classify to as much data as you want!

We covered naïve Bayes, reviewed multinomial logistic regression, and, briefly, the perceptron.

Multinomial Logistic Regression as "Log Loss"

$$p(L = \ell \mid \boldsymbol{x}) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)}{\sum_{\ell' \in \mathcal{L}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell')}$$

MLE can be rewritten as a minimization problem:

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{n} \underbrace{\log \sum_{\ell' \in \mathcal{L}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell')}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell_i)}_{\text{hope}}$$

Recall from lecture 3:

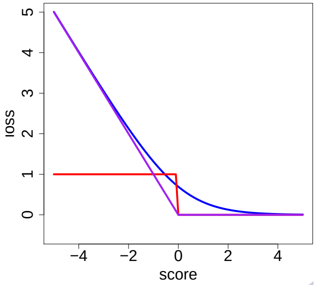
- Be wise and regularize!
- Solve with batch or stochastic gradient methods.
- w_j has an interpretation.

Log Loss and Hinge Loss for $(\boldsymbol{x}, \ell)$

$$\text{log loss:} \quad \left(\log \sum_{\ell' \in \mathcal{L}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell') \right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)$$

$$\text{hinge loss:} \quad \left(\max_{\ell' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell') \right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)$$

In the binary case, where "score" is the linear score of the correct label:



Minimizing Hinge Loss: Perceptron

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \left(\max_{\ell' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell') \right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell_i)$$

Stochastic subgradient descent on the above is called the **perceptron** algorithm.

- For $t \in \{1, \ldots, T\}$:
  - Pick i_t uniformly at random from $\{1, \ldots, n\}$.
  - $\hat{\ell}_{i_t} \leftarrow \mathrm{argmax}_{\ell \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_{i_t}, \ell)$
  - $\mathbf{w} \leftarrow \mathbf{w} - \alpha \left(\boldsymbol{\phi}(\boldsymbol{x}_{i_t}, \hat{\ell}_{i_t}) - \boldsymbol{\phi}(\boldsymbol{x}_{i_t}, \ell_{i_t}) \right)$

Error Costs

Suppose that not all mistakes are equally bad.

E.g., false positives vs. false negatives in spam detection.

Error Costs

Suppose that not all mistakes are equally bad.

E.g., false positives vs. false negatives in spam detection.

Let $\mathrm{cost}(\ell, \ell')$ quantify the "badness" of substituting ℓ' for correct label ℓ.

Error Costs

Suppose that not all mistakes are equally bad.

E.g., false positives vs. false negatives in spam detection.

Let $\mathrm{cost}(\ell, \ell')$ quantify the "badness" of substituting ℓ' for correct label ℓ.

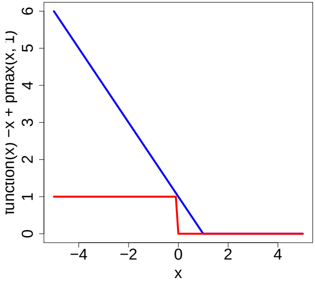
Intuition: estimate the scoring function so that

$$\mathrm{score}(\ell_i) - \mathrm{score}(\hat{\ell}) \propto \mathrm{cost}(\ell_i, \hat{\ell})$$

General Hinge Loss for $(\boldsymbol{x}, \ell)$

$$\left(\max_{\ell' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell') + \mathrm{cost}(\ell, \ell') \right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)$$

In the binary case, with $\mathrm{cost}(-1, 1) = 1$:



In blue is the general hinge loss; in red is the "zero-one" loss (error).

Support Vector Machines

A different motivation for the generalized hinge:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \sum_{\ell \in \mathcal{L}} \alpha_{i,\ell} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell)$$

where most only a small number of $\alpha_{i,\ell}$ are nonzero.

Support Vector Machines

A different motivation for the generalized hinge:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \sum_{\ell \in \mathcal{L}} \alpha_{i,\ell} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell)$$

where most only a small number of $\alpha_{i,\ell}$ are nonzero.

Those $\boldsymbol{\phi}(\boldsymbol{x}_i, \ell)$ are called "support vectors" because they "support" the decision boundary.

$$\hat{\mathbf{w}} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell') = \sum_{(i,\ell) \in \mathcal{S}} \alpha_{i,\ell} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell) \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell')$$

See Crammer and Singer (2001) for the multiclass version.

Support Vector Machines

A different motivation for the generalized hinge:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \sum_{\ell \in \mathcal{L}} \alpha_{i,\ell} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell)$$

where most only a small number of $\alpha_{i,\ell}$ are nonzero.

Those $\boldsymbol{\phi}(\boldsymbol{x}_i, \ell)$ are called "support vectors" because they "support" the decision boundary.

$$\hat{\mathbf{w}} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell') = \sum_{(i,\ell) \in \mathcal{S}} \alpha_{i,\ell} \cdot \boldsymbol{\phi}(\boldsymbol{x}_i, \ell) \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell')$$

See Crammer and Singer (2001) for the multiclass version.

Really good tool: SVM*light*, `http://svmlight.joachims.org`

Support Vector Machines: Remarks

- Regularization is critical; squared ℓ_2 is most common, and often used in (yet another) motivation around the idea of "maximizing margin" around the hyperplane separator.

Support Vector Machines: Remarks

- Regularization is critical; squared ℓ_2 is most common, and often used in (yet another) motivation around the idea of "maximizing margin" around the hyperplane separator.
- Often, instead of linear models that explicitly calculate $\mathbf{w} \cdot \boldsymbol{\phi}$, these methods are "kernelized" and rearrange all calculations to involve inner-products between $\boldsymbol{\phi}$ vectors.
  - Example:

$$K_{\text{linear}}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w}$$
$$K_{\text{polynomial}}(\mathbf{v}, \mathbf{w}) = (\mathbf{v} \cdot \mathbf{w} + 1)^p$$
$$K_{\text{Gaussian}}(\mathbf{v}, \mathbf{w}) = \exp - \frac{\|\mathbf{v} - \mathbf{w}\|_2^2}{2\sigma^2}$$

  - Linear kernels are most common in NLP.

General Remarks

- Text classification: many problems, all solved with supervised learners.
  - Lexicon features can provide problem-specific guidance.

General Remarks

- Text classification: many problems, all solved with supervised learners.
  - Lexicon features can provide problem-specific guidance.
- Naïve Bayes, log-linear, and SVM are all *linear* methods that tend to work reasonably well, with good features and smoothing/regularization.
  - You should have a basic understanding of the tradeoffs in choosing among them.

General Remarks

- Text classification: many problems, all solved with supervised learners.
  - Lexicon features can provide problem-specific guidance.
- Naïve Bayes, log-linear, and SVM are all *linear* methods that tend to work reasonably well, with good features and smoothing/regularization.
  - You should have a basic understanding of the tradeoffs in choosing among them.
- Rumor: random forests are widely used in industry when performance matters more than interpretability.

General Remarks

- Text classification: many problems, all solved with supervised learners.
  - Lexicon features can provide problem-specific guidance.
- Naïve Bayes, log-linear, and SVM are all *linear* methods that tend to work reasonably well, with good features and smoothing/regularization.
  - You should have a basic understanding of the tradeoffs in choosing among them.
- Rumor: random forests are widely used in industry when performance matters more than interpretability.
- Lots of papers about neural networks, but with hyperparameter tuning applied fairly to linear models, the advantage is not clear (Yogatama et al., 2015).

Readings and Reminders

- Jurafsky and Martin (2015); Collins (2011)
- Submit a suggestion for an exam question by Friday at 5pm.

References I

Michael Collins. The naive Bayes model, maximum-likelihood estimation, and the EM algorithm, 2011. URL http://www.cs.columbia.edu/~mcollins/em.pdf.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(5): 265–292, 2001.

Daniel Jurafsky and James H. Martin. Classification: Naive Bayes, logistic regression, sentiment (draft chapter), 2015. URL https://web.stanford.edu/~jurafsky/slp3/7.pdf.

Dani Yogatama, Lingpeng Kong, and Noah A. Smith. Bayesian optimization of text representations. In *Proc. of EMNLP*, 2015. URL http://www.aclweb.org/anthology/D/D15/D15-1251.pdf.