# Transfer Learning for Text Categorization

Chuong B. Do and Andrew Y. Ng
Presenter: Lei Tang

Department of Computer Science & Engineering
Arizona State University

6th October 2006

- also Meta Learning, Multi-task Learning, Structural Learning
- For linear classifiers, past work(naive Bayes, TFIDF) focus on learning the weight (actually try to identify better functions mapping from statistics to weight)
- This work focus on learning the MAPPING FUNCTION from related classification problems

# Linear Classifier

- Linear Classifier: $f_k(x) = \sum \theta_{ki} x_i$, and

$$y = \arg\max_k f_k(x)$$

where $x_i$ is usually a frequency corresponding to term $i$.

- Naïve Bayes:

$$f_k^{NB}(x) = \log \hat{p}(y = k) + \sum_{i=1}^{n} x_i \log \hat{p}(w_i | y = k)$$

- Rocchio Alorithm

$$f_k^{Rocchio} = \sum_{i=1}^{n} (\bar{x}_i | y = k \cdot \log idf)(x_i \cdot \log idf)$$

# Mapping Function(1)

- Linear Classifier: $f_k(x) = \sum \theta_{ki} x_i$, and where $x_i$ is usually a frequency corresponding to term $i$. We can rewrite $f_k(x) = \sum \theta_{ki} x_i$ as

$$f_k(x) = \sum g(\mathbf{u}_{ki}) x_i$$

  where $\mathbf{u}_k i$ is a vector of some statistics computed from the training set. (Similar to the concept of *sufficient statistics*)

-

$$\mathbf{u}_{ki} = \begin{bmatrix} u_{ki1} \\ u_{ki2} \\ u_{ki3} \\ u_{ki4} \\ u_{ki5} \end{bmatrix} = \begin{bmatrix} \#w_i \textit{ appear in documents of class } k \\ \#\textit{documents of class } k \textit{ containing } w_i \\ \#\textit{total words in documents of class } k \\ \#\textit{documents of class } k \\ \#\textit{total documents} \end{bmatrix}$$

# Mapping Function(2)

$$f_k^{NB}(x) = \log \hat{p}(y = k) + \sum_{i=1}^{n} x_i \log \hat{p}(w_i | y = k)$$

$$\mathbf{u}_{ki} = \begin{bmatrix} u_{ki1} \\ u_{ki2} \\ u_{ki3} \\ u_{ki4} \\ u_{ki5} \end{bmatrix} = \begin{bmatrix} \#w_i \; appear \; in \; documents \; of \; class \; k \\ \#documents \; of \; class \; k \; containing \; w_i \\ \#total \; words \; in \; documents \; of \; class \; k \\ \#documents \; of \; class \; k \\ \#total \; documents \end{bmatrix}$$

$$g_{NB}(\mathbf{u}_{ki}) = \log \frac{u_{ki1} + \varepsilon}{u_{ki3} + n\varepsilon}$$

Similarly, TFIDF can also be written as linear combination of $g(u_{ki}) \cdot x_i$

# Reformulation of logistic regression

- Why not learn the mapping function $g$ automatically?
- The authors assume the mapping function to be linear (Can be extended later to nonlinear case by kernel trick)

$$g(\mathbf{u}_{ki}) = \beta^T \mathbf{u}_{ki}$$

- Reformulate the logistic regression:

$$p(y = k|\mathbf{x}; \{\theta_{ki}\}) := \frac{\exp(\sum_i \theta_{ki} x_i)}{\sum_{k'} \exp(\sum_i \theta_{k'i} x_i)}$$

$$p(y = k|\mathbf{x}; \beta) = \frac{\exp(\sum_i \beta^T \mathbf{u}_{ki} x_i)}{\sum_{k'} \exp(\sum_i \beta^T \mathbf{u}_{k'i} x_i)}$$

**To maximize the log likelihood**

$$\ell(\beta : \Omega) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}; \beta) - C\|\beta\|^2$$

# Reformulation of logistic regression

- Why not learn the mapping function $g$ automatically?
- The authors assume the mapping function to be linear (Can be extended later to nonlinear case by kernel trick)

$$g(\mathbf{u}_{ki}) = \beta^T \mathbf{u}_{ki}$$

- Reformulate the logistic regression:

$$p(y = k|\mathbf{x}; \{\theta_{ki}\}) := \frac{\exp(\sum_i \theta_{ki} x_i)}{\sum_{k'} \exp(\sum_i \theta_{k'i} x_i)}$$

$$p(y = k|\mathbf{x}; \beta) = \frac{\exp(\sum_i \beta^T \mathbf{u}_{ki} x_i)}{\sum_{k'} \exp(\sum_i \beta^T \mathbf{u}_{k'i} x_i)}$$

**To maximize the log likelihood**

$$\ell(\beta : \Omega) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}; \beta) - C\|\beta\|^2$$

# Reformulation of logistic regression

- Why not learn the mapping function $g$ automatically?
- The authors assume the mapping function to be linear (Can be extended later to nonlinear case by kernel trick)

$$g(\mathbf{u}_{ki}) = \beta^T \mathbf{u}_{ki}$$

- Reformulate the logistic regression:

$$p(y = k|\mathbf{x}; \{\theta_{ki}\}) := \frac{\exp(\sum_i \theta_{ki} x_i)}{\sum_{k'} \exp(\sum_i \theta_{k'i} x_i)}$$

$$p(y = k|\mathbf{x}; \beta) = \frac{\exp(\sum_i \beta^T \mathbf{u}_{ki} x_i)}{\sum_{k'} \exp(\sum_i \beta^T \mathbf{u}_{k'i} x_i)}$$

**To maximize the log likelihood**

$$\ell(\beta : \Omega) = \sum_{i=1}^{m} \log p(y^{(j)}|x^{(j)}; \beta) - C\|\beta\|^2$$

# Reformulation of logistic regression

- Why not learn the mapping function $g$ automatically?
- The authors assume the mapping function to be linear (Can be extended later to nonlinear case by kernel trick)

$$g(\mathbf{u}_{ki}) = \beta^T \mathbf{u}_{ki}$$

- Reformulate the logistic regression:

$$\begin{aligned} p(y = k|\mathbf{x}; \{\theta_{ki}\}) &:= \frac{\exp(\sum_i \theta_{ki} x_i)}{\sum_{k'} \exp(\sum_i \theta_{k'i} x_i)} \\ p(y = k|\mathbf{x}; \beta) &= \frac{\exp(\sum_i \beta^T \mathbf{u}_{ki} x_i)}{\sum_{k'} \exp(\sum_i \beta^T \mathbf{u}_{k'i} x_i)} \end{aligned}$$

**To maximize the log likelihood**

$$\ell(\beta : \Omega) = \sum_{i=1}^{m} \log p(y^{(j)}|x^{(j)}; \beta) - C\|\beta\|^2$$

# Reformulation of logistic regression

- Why not learn the mapping function $g$ automatically?
- The authors assume the mapping function to be linear (Can be extended later to nonlinear case by kernel trick)

$$g(\mathbf{u}_{ki}) = \beta^T \mathbf{u}_{ki}$$

- Reformulate the logistic regression:

$$\begin{aligned} p(y = k|\mathbf{x}; \{\theta_{ki}\}) &:= \frac{\exp(\sum_i \theta_{ki} x_i)}{\sum_{k'} \exp(\sum_i \theta_{k'i} x_i)} \\ p(y = k|\mathbf{x}; \beta) &= \frac{\exp(\sum_i \beta^T \mathbf{u}_{ki} x_i)}{\sum_{k'} \exp(\sum_i \beta^T \mathbf{u}_{k'i} x_i)} \end{aligned}$$

**To maximize the log likelihood**

$$\ell(\beta : \Omega) = \sum_{i=1}^{m} \log p(y^{(j)}|x^{(j)}; \beta) - C\|\beta\|^2$$

# Nonlinear case

$$\beta^* = \sum_{j=1}^{m} \sum_{k} \alpha^*_{jk} \sum_{i} \mathbf{u}^{(j)}_{ki} x^{(j)}_i$$

User kernel trick, we can extend the mapping function $g$ to nonlinear cases(Details skipped here)

# Experiments

1. How to evaluate?
   - Typical cross validation
   - dmoz with 16 top-level categories: Test on one category based on built 450 classification problems from the other 15 categories
   - Four corpora: learned from one corpora and test on the other three corpora
2. Results:
   Better than NBC, logistic regression, 1-vs-all SVM, MC-SVM

# Some concerns

1. Each learning task is 10-class with 2 instances in each category for training, and 1 instance in each category for testing. Not very surprising that it outperforms SVM.
2. Why regularization?

God says: regularization is always helpful! ??

3. Why use logistic regression formulation?

Because logistic regression can model any decision boundaries?

4. Transfer Learning: Learn how to learn!

A new machine learning problem?

# Some concerns

1. Each learning task is 10-class with 2 instances in each category for training, and 1 instance in each category for testing. Not very surprising that it outperforms SVM.
2. Why regularization?

God says: regularization is always helpful! ??

3. Why use logistic regression formulation?

Because logistic regression can model any decision boundaries?

4. Transfer Learning: Learn how to learn!

A new machine learning problem?

# Some concerns

1. Each learning task is 10-class with 2 instances in each category for training, and 1 instance in each category for testing. Not very surprising that it outperforms SVM.
2. Why regularization?

God says: regularization is always helpful! ??

3. Why use logistic regression formulation?

Because logistic regression can model any decision boundaries?

4. Transfer Learning: Learn how to learn!

A new machine learning problem?

# Some concerns

1. Each learning task is 10-class with 2 instances in each category for training, and 1 instance in each category for testing. Not very surprising that it outperforms SVM.
2. Why regularization?

God says: regularization is always helpful! ??

3. Why use logistic regression formulation?

Because logistic regression can model any decision boundaries?

4. Transfer Learning: Learn how to learn!

A new machine learning problem?