

# Active Accuracy Estimation on Large Datasets

Namit Katariya, Arun Iyer, Sunita Sarawagi

IIT Bombay

December 7, 2012

International Conference on Data Mining, 2012

# Outline

- 1 Motivation
- 2 Problem Statement
- 3 Related Work
- 4 Proposed Solution
- 5 Results
- 6 Summary

# Motivation

- Many applications rely on output of imperfect classifiers deployed on large datasets

**Examples:** Web page classification, classifying columns of a table to their semantic types

- Common characteristics**

- Large and diverse dataset  $D$
- Labeled data  $L$  unrepresentative of the entire dataset

- Measured accuracy on labeled set  $\neq$  True accuracy on data

- Need a method that can converge to the true accuracy

- 1 An algorithm that returns a good estimate ( $\hat{\mu}$ ) of true accuracy ( $\mu$ ) of the classifier
- 2 *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

# Motivation

- Many applications rely on output of imperfect classifiers deployed on large datasets

**Examples:** Web page classification, classifying columns of a table to their semantic types

- **Common characteristics**

- Large and diverse dataset  $D$
- Labeled data  $L$  unrepresentative of the entire dataset

- Measured accuracy on labeled set  $\neq$  True accuracy on data

- Need a method that can converge to the true accuracy

- ① An algorithm that returns a good estimate ( $\hat{\mu}$ ) of true accuracy ( $\mu$ ) of the classifier
- ② *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

# Motivation

- Many applications rely on output of imperfect classifiers deployed on large datasets

**Examples:** Web page classification, classifying columns of a table to their semantic types

- **Common characteristics**

- Large and diverse dataset  $D$
  - Labeled data  $L$  unrepresentative of the entire dataset
- Measured accuracy on labeled set  $\neq$  True accuracy on data
- Need a method that can converge to the true accuracy
  - ① An algorithm that returns a good estimate ( $\hat{\mu}$ ) of true accuracy ( $\mu$ ) of the classifier
  - ② *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

# Motivation

- Many applications rely on output of imperfect classifiers deployed on large datasets

**Examples:** Web page classification, classifying columns of a table to their semantic types

- **Common characteristics**

- Large and diverse dataset  $D$
- Labeled data  $L$  unrepresentative of the entire dataset

- Measured accuracy on labeled set  $\neq$  True accuracy on data

- Need a method that can converge to the true accuracy

- ① An algorithm that returns a good estimate ( $\hat{\mu}$ ) of true accuracy ( $\mu$ ) of the classifier
- ② *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

# Motivation

- Many applications rely on output of imperfect classifiers deployed on large datasets

**Examples:** Web page classification, classifying columns of a table to their semantic types

- **Common characteristics**

- Large and diverse dataset  $D$
  - Labeled data  $L$  unrepresentative of the entire dataset
- Measured accuracy on labeled set  $\neq$  True accuracy on data
- Need a method that can converge to the true accuracy
  - ① An algorithm that returns a good estimate ( $\hat{\mu}$ ) of true accuracy ( $\mu$ ) of the classifier
  - ② *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

# Outline

- 1 Motivation
- 2 Problem Statement**
- 3 Related Work
- 4 Proposed Solution
- 5 Results
- 6 Summary



# Problem Statement

- ➊ **Accuracy estimation** : Estimate accuracy of a classifier on a large unlabeled dataset based on a **small, possibly unrepresentative, labeled set** and a human labeler
- ➋ **Scalable algorithm** : Perform accuracy estimation on unlabeled data so large that it makes **even a single sequential scan impractical** in an interactive setting

# Outline

- 1 Motivation
- 2 Problem Statement
- 3 Related Work**
- 4 Proposed Solution
- 5 Results
- 6 Summary

# Related Work

- Most existing work on *learning* rather than *evaluating* classifiers
- Existing works on selecting instances for evaluating classifiers:
  - (Sawade et al., 2010) present a new proposal distribution for sampling instances
  - (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011) use stratified sampling

However, all assume classifier  $C(\mathbf{x})$  to be probabilistic and use its  $\Pr(y|\mathbf{x})$  scores for selection and/or stratification.

- Unlike (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011), *instead of fixing the stratification, we learn a new one every time more data gets labeled*
- None of the existing methods consider cases where the dataset  $D$  is too large to even afford a single sequential scan

# Related Work

- Most existing work on *learning* rather than *evaluating* classifiers
- Existing works on selecting instances for evaluating classifiers:
  - (Sawade et al., 2010) present a new proposal distribution for sampling instances
  - (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011) use stratified sampling

However, all assume classifier  $C(\mathbf{x})$  to be probabilistic and use its  $\Pr(y|\mathbf{x})$  scores for selection and/or stratification.

- Unlike (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011), *instead of fixing the stratification, we learn a new one every time more data gets labeled*
- None of the existing methods consider cases where the dataset  $D$  is too large to even afford a single sequential scan

# Related Work

- Most existing work on *learning* rather than *evaluating* classifiers
- Existing works on selecting instances for evaluating classifiers:
  - (Sawade et al., 2010) present a new proposal distribution for sampling instances
  - (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011) use stratified sampling

However, all assume classifier  $C(\mathbf{x})$  to be probabilistic and use its  $\Pr(y|\mathbf{x})$  scores for selection and/or stratification.

- Unlike (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011), *instead of fixing the stratification, we learn a new one every time more data gets labeled*
- None of the existing methods consider cases where the dataset  $D$  is too large to even afford a single sequential scan

# Related Work

- Most existing work on *learning* rather than *evaluating* classifiers
- Existing works on selecting instances for evaluating classifiers:
  - (Sawade et al., 2010) present a new proposal distribution for sampling instances
  - (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011) use stratified sampling

However, all assume classifier  $C(\mathbf{x})$  to be probabilistic and use its  $\Pr(y|\mathbf{x})$  scores for selection and/or stratification.

- Unlike (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011), *instead of fixing the stratification, we learn a new one every time more data gets labeled*
- None of the existing methods consider cases where the dataset  $D$  is too large to even afford a single sequential scan

# Related Work

- Most existing work on *learning* rather than *evaluating* classifiers
- Existing works on selecting instances for evaluating classifiers:
  - (Sawade et al., 2010) present a new proposal distribution for sampling instances
  - (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011) use stratified sampling

However, all assume classifier  $C(\mathbf{x})$  to be probabilistic and use its  $\Pr(y|\mathbf{x})$  scores for selection and/or stratification.

- Unlike (Bennett and Carvalho, 2010) and (Druck and McCallum, 2011), *instead of fixing the stratification, we learn a new one every time more data gets labeled*
- None of the existing methods consider cases where the dataset  $D$  is too large to even afford a single sequential scan

# Outline

- 1 Motivation
- 2 Problem Statement
- 3 Related Work
- 4 Proposed Solution**
- 5 Results
- 6 Summary



# Overall Idea

---

**Algorithm 1** Loop for active accuracy estimation

---

- 1:  $B = \# \text{buckets}$ ,  $r = \# \text{bits}$ ,  $\mathbf{f} = \text{feat. vector}$ ,  $\mathbf{w}_{1\dots r} = \text{hyperplanes}$
  - 2:  $\hat{\mu}_b, p_b = \text{accuracy \& weight estimates for bucket } b$
  - 3: **repeat**
  - 4:   Learn stratification function  $h(\mathbf{f}|\mathbf{w}_{1\dots r})$
  - 5:   Stratify  $L$  via  $h(\cdot)$  & compute  $\{\hat{\mu}_b : 1 \leq b \leq B\}$
  - 6:   Stratify  $D$  via  $h(\cdot)$  & compute  $\{p_b : 1 \leq b \leq B\}$
  - 7:   Display accuracy estimates:  $\hat{\mu}_S = \sum_b p_b \hat{\mu}_b$
  - 8:   Get stratified sample set  $L'$  from  $D$
  - 9:   For each  $\mathbf{x}_i \in L'$ , get label  $y_i$ , and add  $(\mathbf{x}_i, y_i)$  to  $L$
  - 10: **until** accuracy  $\hat{\mu}_S$  not converged and labeler not bored.
  - 11: **Return**  $\hat{\mu}_S$
-

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. **Smooth based on labeled data in neighbouring buckets**
- Method agnostic to the type of classifier under consideration

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. **Smooth based on labeled data in neighbouring buckets**
- Method agnostic to the type of classifier under consideration

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. **Smooth based on labeled data in neighbouring buckets**
- Method agnostic to the type of classifier under consideration

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. Smooth based on labeled data in neighbouring buckets
- Method agnostic to the type of classifier under consideration

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. **Smooth based on labeled data in neighbouring buckets**
- Method agnostic to the type of classifier under consideration

# Learning a stratification strategy

- Stratify input space so that instances in each stratum have similar accuracy values
  - *Supervised clustering methods* : Learn a distance function  
**Issue** : Do not scale well
  - **Proposal** : Use **Hash codes** based on projections on hyperplanes learned over the feature space
- Learning hyperplanes (details in the paper)
  - *Smoothing the objective* : Convex upper bound
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct hyperplanes* : Re-weight instances (boosting)
- $\hat{\mu}_b = \frac{1}{n_b} \sum_{i \in L_b} a_i$  prone to over-fitting. **Smooth based on labeled data in neighbouring buckets**
- Method agnostic to the type of classifier under consideration

# Scaling up – Instance selection on large amounts of data

- Unlabeled data accessed for
  - ① computing  $p_b$  = the weight corresponding to each bucket  $b$
  - ② generating sample  $L'$  from  $D$  to label and add to  $L$
- Solutions for both, assigning bucket weights and selecting instances, based on sampling from a proposal distribution
- In each case, proposal distribution found by setting up an appropriate convex optimization problem
- Optimal proposal distribution can be calculated using some standard assumptions on the index (details in the paper)



# Scaling up – Instance selection on large amounts of data

- Unlabeled data accessed for
  - ➊ computing  $p_b$  = the weight corresponding to each bucket  $b$
  - ➋ generating sample  $L'$  from  $D$  to label and add to  $L$
- Solutions for both, assigning bucket weights and selecting instances, based on sampling from a proposal distribution
- In each case, proposal distribution found by setting up an appropriate convex optimization problem
- Optimal proposal distribution can be calculated using some standard assumptions on the index (details in the paper)

# Scaling up – Instance selection on large amounts of data

- Unlabeled data accessed for
  - ① computing  $p_b$  = the weight corresponding to each bucket  $b$
  - ② generating sample  $L'$  from  $D$  to label and add to  $L$
- Solutions for both, assigning bucket weights and selecting instances, based on sampling from a proposal distribution
- In each case, proposal distribution found by setting up an appropriate convex optimization problem
- Optimal proposal distribution can be calculated using some standard assumptions on the index (details in the paper)

# Scaling up – Instance selection on large amounts of data

- Unlabeled data accessed for
  - ① computing  $p_b$  = the weight corresponding to each bucket  $b$
  - ② generating sample  $L'$  from  $D$  to label and add to  $L$
- Solutions for both, assigning bucket weights and selecting instances, based on sampling from a proposal distribution
- In each case, proposal distribution found by setting up an appropriate convex optimization problem
- Optimal proposal distribution can be calculated using some standard assumptions on the index (details in the paper)

# Scaling up – Instance selection on large amounts of data

- Unlabeled data accessed for
  - ① computing  $p_b$  = the weight corresponding to each bucket  $b$
  - ② generating sample  $L'$  from  $D$  to label and add to  $L$
- Solutions for both, assigning bucket weights and selecting instances, based on sampling from a proposal distribution
- In each case, proposal distribution found by setting up an appropriate convex optimization problem
- Optimal proposal distribution can be calculated using some standard assumptions on the index (details in the paper)

# Outline

- 1 Motivation
- 2 Problem Statement
- 3 Related Work
- 4 Proposed Solution
- 5 Results**
- 6 Summary

# Results

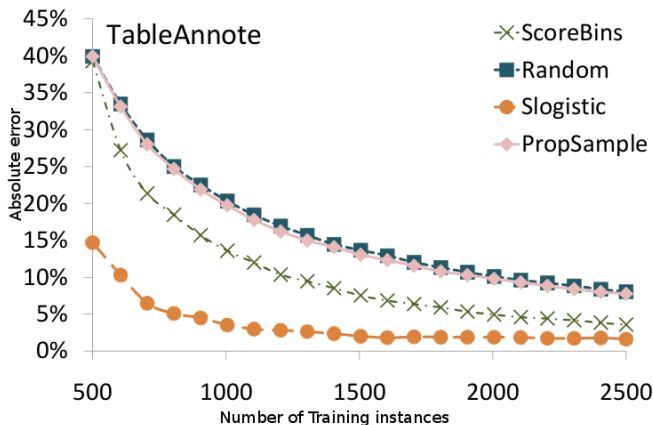
## Summary of datasets used

- **TableAnnote** : Annotate columns of Web tables to type nodes of an ontology
- **Spam** : Classifying web-pages as spam or not
- **DNA** : Binary DNA classification task
- **HomeGround, HomePredicted** : Dataset of (entity, web-page) instances and decide if web-page was a homepage for the entity

Dataset	# Features	Size		Accuracy (%)	
		Seed( $L$ )	Unlabeled( $D$ )	Seed( $L$ )	True( $D$ )
TableAnnote	42	541	11,954,983	56.4	16.5
Spam	1000	5000	350,000	86.4	93.2
DNA	800	100,000	50,000,000	72.2	77.9
HomeGround	66	514	1060	50.4	32.8
HomePredicted	66	8658	13,951,053	83.2	93.9

# Results

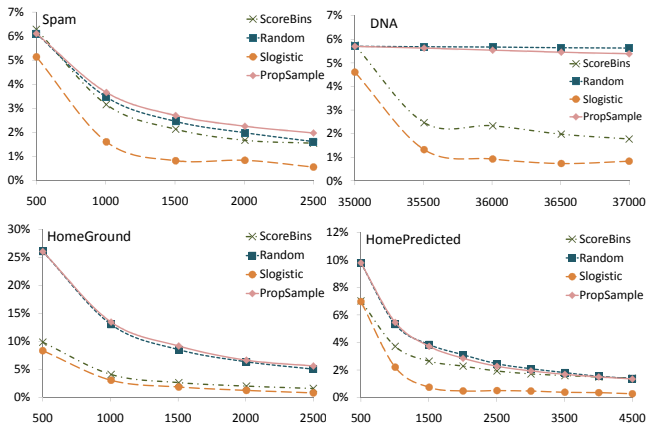
Comparison of estimation strategies on the TableAnnote dataset



Random	Random Sampling
PropSample	Sampling from proposal distribution
ScoreBins	Stratified sampling with scores

# Results

## Comparison of estimation strategies on remaining datasets



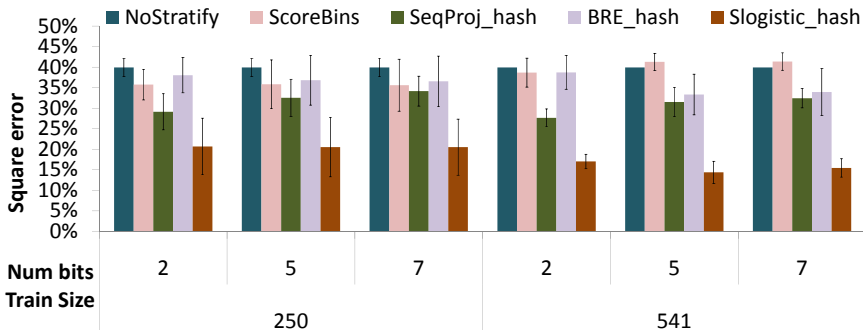
**Figure:** Absolute error (on the Y axis) of different estimation algorithms against increasing number of labeled instances (on the X axis)



# Results

Comparison of different stratification methods on the TableAnnote dataset

## TableAnnote



NoStratify

Simple averaging (no stratification)

ScoreBins

Stratify using classifier scores

SeqProj\_hash

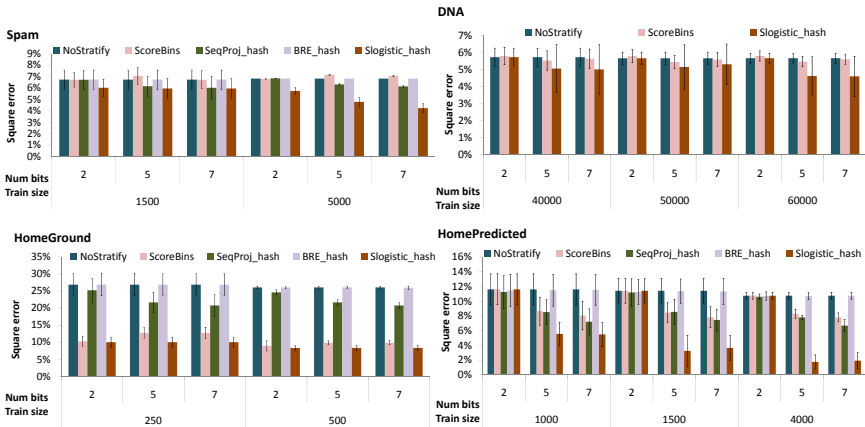
Learn hyperplanes using method in (Wang et al., 2010)

BRE\_hash

Learn hyperplanes using method in (Kulis and Darrell, 2009)

# Results

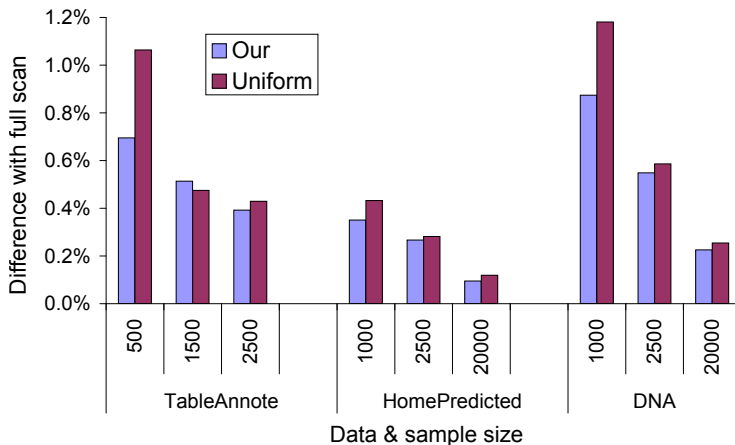
## Comparison of different stratification methods on remaining datasets



**Figure:** Error of different stratification methods against increasing training sizes and for different number of bits

# Results

Comparison of methods of sampling from indexed data for estimating bucket weights



**Figure:** Comparing methods of sampling from indexed data for estimating bucket weights

# Outline

- 1 Motivation
- 2 Problem Statement
- 3 Related Work
- 4 Proposed Solution
- 5 Results
- 6 Summary**

# Summary

- 1 Addressed the challenge of *calibrating a classifier's accuracy on large unlabeled datasets* given small amounts of labeled data and a human labeler
- 2 Proposed a stratified sampling-based method for accuracy estimation that provides better estimates than simple averaging & better selection of instances for labeling than random sampling
- 3 Between 15% and 62% relative reduction in error achieved compared to existing approaches
- 4 Algorithm made *scalable* by proposing optimal sampling strategies for accessing indexed unlabeled data directly
- 5 Close to optimal performance while reading three orders of magnitude fewer instances on large datasets

# Summary

- 1 Addressed the challenge of *calibrating a classifier's accuracy on large unlabeled datasets* given small amounts of labeled data and a human labeler
- 2 Proposed a stratified sampling-based method for accuracy estimation that provides better estimates than simple averaging & better selection of instances for labeling than random sampling
- 3 Between 15% and 62% relative reduction in error achieved compared to existing approaches
- 4 Algorithm made *scalable* by proposing optimal sampling strategies for accessing indexed unlabeled data directly
- 5 Close to optimal performance while reading three orders of magnitude fewer instances on large datasets

# Summary

- 1 Addressed the challenge of *calibrating a classifier's accuracy on large unlabeled datasets* given small amounts of labeled data and a human labeler
- 2 Proposed a stratified sampling-based method for accuracy estimation that provides better estimates than simple averaging & better selection of instances for labeling than random sampling
- 3 Between 15% and 62% relative reduction in error achieved compared to existing approaches
- 4 Algorithm made *scalable* by proposing optimal sampling strategies for accessing indexed unlabeled data directly
- 5 Close to optimal performance while reading three orders of magnitude fewer instances on large datasets

# Summary

- ① Addressed the challenge of *calibrating a classifier's accuracy on large unlabeled datasets* given small amounts of labeled data and a human labeler
- ② Proposed a stratified sampling-based method for accuracy estimation that provides better estimates than simple averaging & better selection of instances for labeling than random sampling
- ③ Between 15% and 62% relative reduction in error achieved compared to existing approaches
- ④ Algorithm made *scalable* by proposing optimal sampling strategies for accessing indexed unlabeled data directly
- ⑤ Close to optimal performance while reading three orders of magnitude fewer instances on large datasets



# Summary

- ① Addressed the challenge of *calibrating a classifier's accuracy on large unlabeled datasets* given small amounts of labeled data and a human labeler
- ② Proposed a stratified sampling-based method for accuracy estimation that provides better estimates than simple averaging & better selection of instances for labeling than random sampling
- ③ Between 15% and 62% relative reduction in error achieved compared to existing approaches
- ④ Algorithm made *scalable* by proposing optimal sampling strategies for accessing indexed unlabeled data directly
- ⑤ Close to optimal performance while reading three orders of magnitude fewer instances on large datasets

# Thank You

# References I

- Bennett, P. N. and Carvalho, V. R. (2010). Online stratified sampling: evaluating classifiers at web-scale. In *CIKM*.
- Druck, G. and McCallum, A. (2011). Toward interactive training and evaluation. In *CIKM*.
- Kulis, B. and Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In *NIPS*.
- Sawade, C., Landwehr, N., Bickel, S., and Scheffer, T. (2010). Active risk estimation. In *ICML*.
- Wang, J., Kumar, S., and Chang, S. (2010). Sequential projection learning for hashing with compact codes. In *ICML*.

# Assigning Bucket Weights

- Sample from a proposal distribution :  $\hat{\mu}_{S_q} = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \frac{1/N}{q(\mathbf{x})} \hat{\mu}_{h(\mathbf{x})}$
- **Result** : When  $q(\mathbf{x})$  is restricted so that all instances within a partition  $u$  are sampled with the same probability  $q_u$ , the expected squared error between  $\hat{\mu}_{S_q}$  and  $\hat{\mu}_S$  is minimized when

$$q_u \propto \sqrt{\sum_b \hat{\mu}_b^2 p(b|u)}$$

- $p(b|u)$  = fraction of instances in  $D_u$  with  $h(\mathbf{x}) = b$
- Initially, use labeled data to estimate  $p(b|u)$
- As more instances are sampled from any  $D_u$ , refine estimates of  $p(b|u)$

# Instance Selection

- Perform importance sampling where  $\text{imp}(\mathbf{x}) \propto \hat{\sigma}_{h(\mathbf{x})}$  without evaluating  $h(\mathbf{x})$  over entire  $D$
- Generate a larger sample  $S$  via proposal distribution  $q(\mathbf{x})$  restricted to choose same  $q(\mathbf{x}) \forall \mathbf{x}$  in data partition  $D_u$
- Then from  $S$  generate the sample of  $k$  instances by weighting each instance as  $f(\mathbf{x})/q(\mathbf{x})$ . Good only if  $q(\mathbf{x}) \sim f(\mathbf{x})$
- Best  $q(\mathbf{x})$  found by solving for unlabeled bucket weights  $q_1, \dots, q_U$  so that expected L1 distance between  $f(\mathbf{x})$  and  $q(\mathbf{x})$  is minimized

$$\min_{q_1, \dots, q_U} \sum_u \sum_b p_u p(b|u) \left| \frac{\hat{\sigma}_b}{Z_f} - q_u \right| \text{ s.t. } \sum_u N p_u q_u = 1$$

- $Z_f$  approximated as  $\sum_b \hat{\sigma}_b \sum_u p_u p(b|u)$
- Get  $p(b|u)$  as explained in the previous slide