Active Accuracy Estimation on Large Datasets

Namit Katariya, Arun Iyer, Sunita Sarawagi

IIT Bombay

International Conference on Data Mining, 2012

Outline

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
- So Measured accuracy on labeled set $\neq$ True accuracy on data
- Hence need a method that can converge to the true accuracy
  1. An algorithm that returns a good estimate $\hat{\mu}$ of true accuracy μ 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
- So Measured accuracy on labeled set $\neq$ True accuracy on data
- Hence need a method that can converge to the true accuracy
  1. An algorithm that returns a good estimate $\hat{\mu}$ of true accuracy μ 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
- So Measured accuracy on labeled set $\neq$ True accuracy on data
- Hence need a method that can converge to the true accuracy
  1. An algorithm that returns a good estimate $\hat{\mu}$ of true accuracy μ 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
- So Measured accuracy on labeled set $\neq$ True accuracy on data
- Hence need a method that can converge to the true accuracy
  1. An algorithm that returns a good estimate $\hat{\mu}$ of true accuracy μ 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
- So Measured accuracy on labeled set $\neq$ True accuracy on data
- Hence need a method that can converge to the true accuracy
  1. An algorithm that returns a good estimate $\hat{\mu}$ of true accuracy μ of the classifier
  2. *Scalable algorithm* : should work on large datasets where sequential scan not possible & data accessible only via an index

Goals

Accuracy estimation : Estimate accuracy of a classifier on a large unlabeled dataset based on a small, possibly unrepresentative, labeled set and a human labeler

- **Results** : Between 15% and 62% relative reduction in error compared to existing approaches *(See Slogistic)*

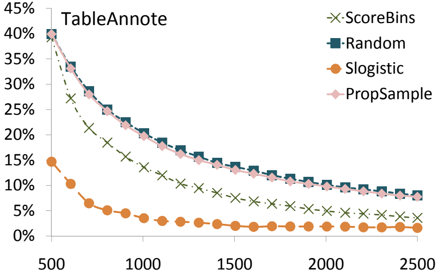

Figure: Absolute error (on the Y axis) of different estimation algorithms against no. of labeled instances (on the X axis)

Goals

Scalable algorithm : Perform accuracy estimation on unlabeled data so large that it makes even a single sequential scan impractical in an interactive setting

- **Results** : Able to match within 0.5% of the estimates of methods based on full scan while sampling just 2.5k instances from indexed unlabeled data

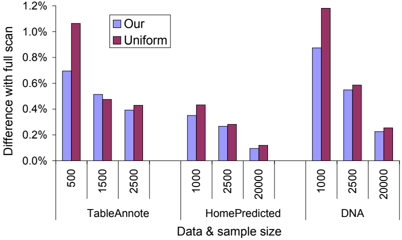

Figure: Comparing methods of sampling from indexed data for estimating bucket weights

Outline

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, both assume classifier $C(\mathbf{x})$ to be probabilistic and use its $\Pr(y|\mathbf{x})$ scores for stratification & selection
- 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, both assume classifier $C(\mathbf{x})$ to be probabilistic and use its $\Pr(y|\mathbf{x})$ scores for stratification & selection
- 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, both assume classifier $C(\mathbf{x})$ to be probabilistic and use its $\Pr(y|\mathbf{x})$ scores for stratification & selection
- 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, both assume classifier $C(\mathbf{x})$ to be probabilistic and use its $\Pr(y|\mathbf{x})$ scores for stratification & selection
- 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, both assume classifier $C(\mathbf{x})$ to be probabilistic and use its $\Pr(y|\mathbf{x})$ scores for stratification & selection
- 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. Problem Motivation
2. Related Work
3. Proposed Solution
4. Results
5. Summary

Overall Idea

Algorithm 1 Loop for active accuracy estimation

```
B = #buckets, f = feature vector, w_{1...r} = hyperplanes
μ̂_b, p_b = accuracy & weight estimates for bucket b
repeat
    Learn stratification function h(f|w_{1...r})
    Stratify L via h(.) & compute {μ̂_b : 1 ≤ b ≤ B}
    Stratify D via h(.) & compute {p_b : 1 ≤ b ≤ B}
    Display accuracy estimates: μ̂_S = Σ_b p_b μ̂_b
    Get stratified sample set L' from D
    For each x_i ∈ L', get label y_i, and add (x_i, y_i) to L
until accuracy μ̂_S not converged and labeler not bored.
Return μ̂_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 (kindly refer the paper for details)
  - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 (kindly refer the paper for details)
    - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
    - *Optimizing the smoothed objective* : EM-like algorithm
    - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 (kindly refer the paper for details)
  - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 (kindly refer the paper for details)
  - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 (kindly refer the paper for details)
  - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 (kindly refer the paper for details)
  - *Smoothing the objective* : Upper-bound with minimum of two convex objectives
  - *Optimizing the smoothed objective* : EM-like algorithm
  - *Ensuring distinct r hyperplanes* : Re-weight instances – ideas from 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 accesed for
  (i) computing p_b (ii) generating sample L' to label & add to L
- *Assumption*: Unlabeled data D indexed so as to partition data into disjoint parts $D_1, \ldots, D_U$. For each partition we can
  - Get its size N_u in terms of number of instances, &
  - Generate a uniform random sample of instances within the partition
- 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 optimization problem (kindly refer to the paper for details)

Scaling up – Instance selection on large amounts of data

- Unlabeled data accesesed for
  (i) computing p_b (ii) generating sample L' to label & add to L
- *Assumption*: Unlabeled data D indexed so as to partition data into disjoint parts $D_1, \ldots, D_U$. For each partition we can
  - Get its size N_u in terms of number of instances, &
  - Generate a uniform random sample of instances within the partition
- 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 optimization problem (kindly refer to the paper for details)

Scaling up – Instance selection on large amounts of data

- Unlabeled data accesesed for
  (i) computing p_b (ii) generating sample L' to label & add to L
- *Assumption*: Unlabeled data D indexed so as to partition data into disjoint parts $D_1, \ldots, D_U$. For each partition we can
  - Get its size N_u in terms of number of instances, &
  - Generate a uniform random sample of instances within the partition
- 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 optimization problem (kindly refer to the paper for details)

Scaling up – Instance selection on large amounts of data

- Unlabeled data accesesed for
  (i) computing p_b (ii) generating sample L' to label & add to L
- *Assumption*: Unlabeled data D indexed so as to partition data into disjoint parts $D_1, \ldots, D_U$. For each partition we can
  - Get its size N_u in terms of number of instances, &
  - Generate a uniform random sample of instances within the partition
- 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 optimization problem (kindly refer to the paper for details)

Scaling up – Instance selection on large amounts of data

- Unlabeled data accesesed for
  (i) computing p_b (ii) generating sample L' to label & add to L
- *Assumption*: Unlabeled data D indexed so as to partition data into disjoint parts $D_1, \ldots, D_U$. For each partition we can
  - Get its size N_u in terms of number of instances, &
  - Generate a uniform random sample of instances within the partition
- 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 optimization problem (kindly refer to the paper for details)

Outline

Results

Summary of datasets used

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

Table: Summary of Datasets

Results

Comparison of estimation strategies on the HomeGround dataset

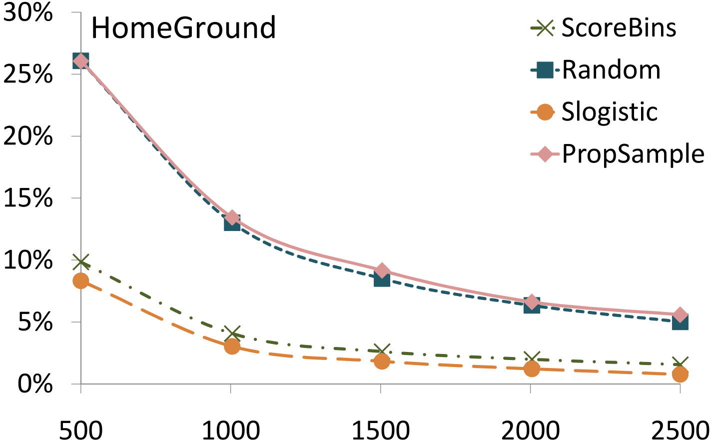

Figure: HomeGround data: Absolute error (Y axis) of different estimation algorithms against increasing number of labeled instances (X axis)

Results

Comparison of estimation strategies on remaining datasets

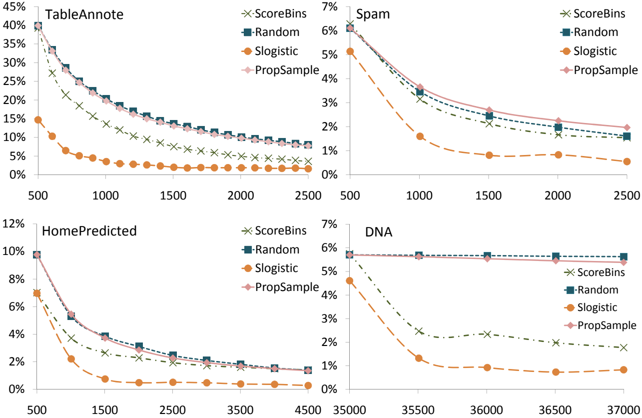

Figure: Absolute error (on the Y axis) of different estimation algorithms against increasing number of labeled instances (on the X axis)

Results

Comparison of different stratification methods on the HomeGround dataset

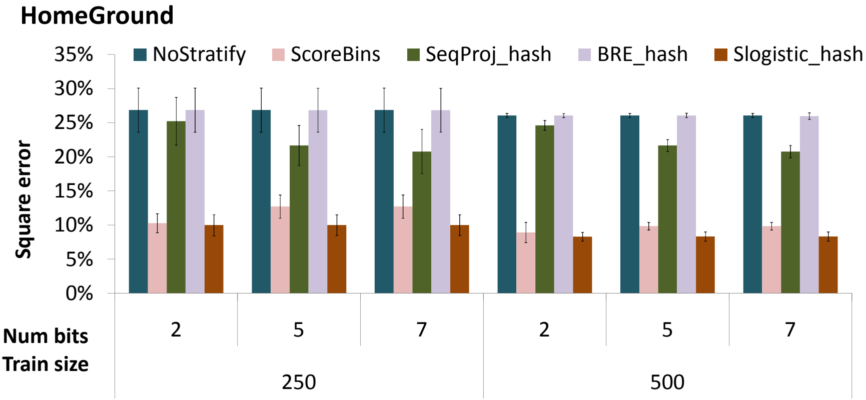

Figure: HomeGround data: Error of different stratification methods against increasing training sizes & for different number of bits

Results

Comparison of different stratification methods on remaining datasets

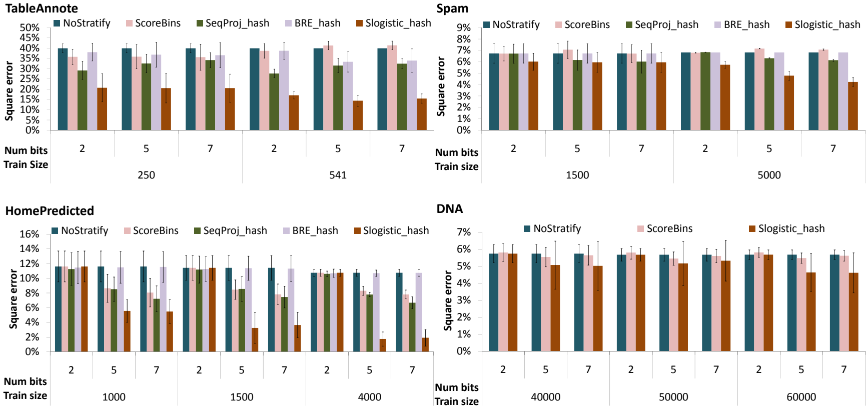

Figure: Error of different stratification methods against increasing training sizes and for different number of bits

Results

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

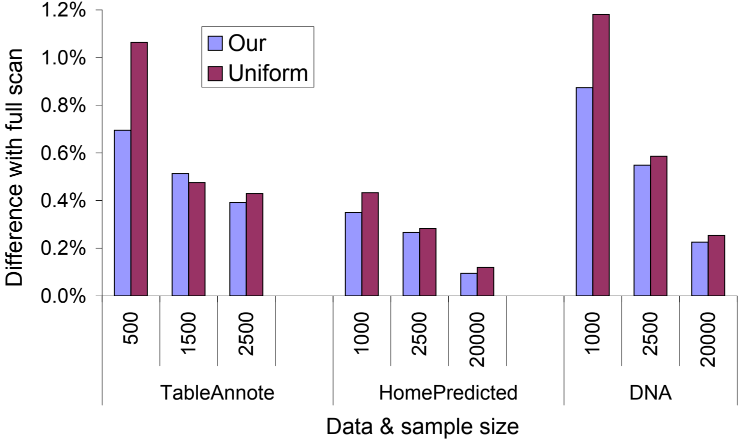

Figure: Comparing methods of sampling from indexed data for estimating bucket weights

Outline

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

Paul N. Bennett and Vitor R. Carvalho. Online stratified sampling: evaluating classifiers at web-scale. In *CIKM*, 2010.

Gregory Druck and Andrew McCallum. Toward interactive training and evaluation. In *CIKM*, 2011.

Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In *ICML*, 2010.

Assigning Bucket Weights

- Sample from a proposal distribution : $\hat{A}_{S_q} = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \frac{1/N}{q(\mathbf{x})} \hat{\mu}_{h(\mathbf{x})}$
- **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 $\hat{A}_{S_q}$ 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 $q_1, \ldots, q_U$ so that expected L1 distance between $f(\mathbf{x})$ and $q(\mathbf{x})$ is minimized

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

- 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