Lecture 8. Evaluation. Other Predictors. Clustering

Ricard Gavaldà

MIRI Seminar on Data Streams, Spring 2015

Contents

Predictor Evaluation

Predictor Evaluation

Target $T : X \to Y$, predictor $P : X \to Y$
Error under loss function $\ell : Y \times Y \to \mathbb{R}$ and some distribution:

$$E_x[\ell(P(x), T(x))]$$

Approximated on a finite labeled sample
$S = ((x_1, y_1), \ldots, (x_n, y_n))$ by

$$\frac{1}{n} \sum_{i=1}^{n} \ell(P(x_i), y_i)$$

Common loss functions

- $\ell(a, b) = (a - b)^2$ (regression)
- $\ell(a, b) = 0$ if $a = b$, 1 otherwise (classification)

Evaluation - Batch setting

- Split training set + test set
- Leave-one-out
- k-fold cross-validation
- ...

None translates obviously to stream setting; esp. with drift

Evaluation in Streams: test-then-train

Interleaved test then train

- train on next N stream items
- evaluate on next M stream items
- repeat

Problem: choice of N and M?

Evaluation in Streams: prequential

Prequential

for each stream item,
    predict
    when (and if) its label is known,
        use prediction and label to evaluate
        then use (item,label) to train

Problem 1: Tends to be pessimistic - early errors when model undertrained count as errors forever

Example. Suppose Pr[error at time t is $1/\sqrt{t}$]
At time T, current error is $\simeq 1/\sqrt{T}$
But E[observed prequential error at time T] $\simeq 2/\sqrt{T}$, twice

Evaluation in Streams: prequential (2)

Prequential

for each stream item,
predict
when (and if) its label is known,
    use prediction and label to evaluate
    then use (item,label) to train

Problem 2: If there is drift, estimation may be arbitrarily off

Solution to problems 1 and 2: Use fading/decaying or sliding windows

Evaluation in Streams: issues

1. Count only “edge over chance agreement”: Kappa statistic

$$\kappa = \frac{\text{Pr[agreement]} - \text{Pr[chance agreement]}}{1 - \text{Pr[chance agreement]}}$$

where

$$\text{Pr[chance agreement]} = \sum_{c} \text{Pr}[c]^2$$

Evaluation in Streams: issues

2. Temporal dependencies

- Question: will it rain tomorrow?
- Pretty good answer: 'yes' if it rained today, 'no' otherwise

Observations are not independently drawn!

Evaluation in Streams: issues

Temporal dependencies: use to your advantage!

- Baseline classifier: $pred_t = y_{t-1}$
- Temporally augmented classifiers [Zliobaite,Bifet et al 15]

$$pred_t = \text{classifier}(x_t, y_{t-1}, \ldots, y_{t-k})$$

Other predictors

Other predictors

- Naive Bayes: Easy to streamize
- Linear regression: Next slide
- Model trees: Decision trees with a model in each leaf
  - Naïve Bayes and Linear regression common choices
- Bagging, Boosting
- Dynamic ensemble methods

Linear regression

Linear regression model:

$$f(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = \mathbf{x} \cdot \mathbf{w}$$

Least squares fitting:

Given $\{(\mathbf{x}_j, y_j)\}_{j=1}^{t}$, minimize sum of squares

$$\sum_{j=1}^{t} (y_j - f(\mathbf{x}_j))^2 = (\mathbf{y} - \mathbf{X} \cdot \mathbf{w})^T \cdot (\mathbf{y} - \mathbf{X} \cdot \mathbf{w})$$

Solution:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Linear regression

- We mentioned a sketch with low memory for this in Linear Algebra lecture
- But, in practice, good old Perceptron has all the advantages:
  - Cost $O(d)$ per item
  - Memory $O(d)$
  - Adapts to change (at rate λ)

Weight update rule: Given $(\mathbf{x}, y)$

$$\mathbf{w_i} = \mathbf{w_i} + \lambda \cdot (y - f_{\mathbf{w}}(\mathbf{x}))\,\mathbf{x_i}$$

= minimizes MSE via Stochastic Gradient Descent

Bagging

How to simulate sampling with replacement in streams?

```
create k empty base classifiers
for each example x
    for i = 1 to k
    give r copies of x to ith classifier with prob. P(r)

predict(x) = majority vote of k classifiers
```

It can be shown that this works for P(r) = Poisson(1)
[Oza-Russell 01]

Boosting

Stream setting: How to do this without storing sample S?

"increase weight in S of instances x where
$sign(w_1 C_1(x) + \cdots + w_t C_t(x))$ is wrong"

- Several proposals exist
- None outperforms bagging so far
- Not well understood theoretically

Dynamic Ensemble Methods

- Many variants
- Keep a pool of classifiers
- Rules for creating new classifiers
- Rules for deleting classifiers
- Rule for predicting from the pool

Exercise 1.

Suggest a sensible implementation for the above that can deal with evolving streams.

Clustering

Clustering

Three main strategies:

- Point assignment
- Agglomerative: bottom-up hierarchical
- Divisive: top-down hierarchical

Clustering: point assignment

Fix k, desired number of clusters:

- k-means / k-median: minimize avg distance to closest cluster
- k-center: minimize max distance to closest cluster (= cluster radius)

Specific sketches mentioned in Lectures 4 and 5

Clustering: point assignment

Several streaming proposals for *k*-means

VFKM: Very Fast *k*-means (Domingos-Hulten 01)

```
repeat
    1. assign points to closest centroid;
    2. move centroids to average of their clusters;
until 3. stable
```

1. S *new* points each round

2. *aproximate* average by Hoeffding bound on S

3. If it does not stabilize, we saw too few points: restart with larger S

Clustering: point assignment

StreamKM++ [Ackermann+12]

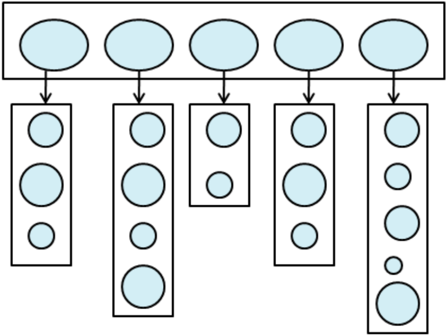
- "Coreset" of a set S w.r.t. a problem: subset of S such that solving the problem on the coreset approximately solves the problem on S

- Recursively builds a tree whose leaves form a coreset for k-means-like algorithm

Divisive clustering: BIRCH [Zhang+96]

- Fast bottom-up clustering
- Works well with “spherical” cluster structure

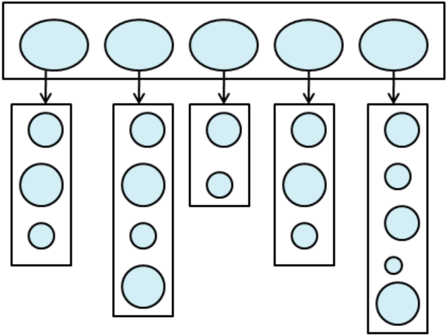
- Tree of clusters, similar to B-tree
- Parameters: branch factor + max radius of clusters
- Stores center + radius + sumofsquares at each node



Divisive clustering: BIRCH [Zhang+96]

- Fast bottom-up clustering
- Works well with “spherical” cluster structure

- Push a new point to closest leaf
- If it fits in that leaf (within radius), done
- Otherwise, create new node at same level
- If capacity exceeded, split parent & recurse



Divisive clustering: CLUSTREAM [Aggarwal+03]

- Unlike BIRCH, can deal with time change

- Each point comes with a time stamp
- Each tree node keeps earliest and latest timestamp
- Nodes that are too old can be dropped

- Snapshot: set of nodes of similar timestamps
- Comparing snapshots = Cluster evolution

Divisive clustering: ODAC [Rodrigues+08]

ODAC: Online Divisive Agglomerative Clustering

- Top-down hierarchical clustering
- Initially for time series clustering, but idea can be generalized to other concepts
- Different tree levels use points from different time windows

Divisive clustering: ODAC [Rodrigues+08]

```
create initial node (root leaf);
for each stream point
    push down point to appropriate leaf;
    update leaf statistics;
    if (leaf is too heterogeneous)
        make it inner node;
        create children = more homogeneous clusters
```

ClusTree [Kranen+11]

- Adaptive index for microclusters (log-time insertion)
- Also timestamps: time-adapting
- Buffer and hitchhikers: adapt to stream speed
- Adapt to available memory
- Implemented in the MOA system