Active learning in sequence labeling

Tomáš Šabata
11. 5. 2017

Czech Technical University in Prague
Faculty of Information technology
Department of Theoretical Computer Science
## Table of contents

1. Introduction

2. Active learning

3. Graphical models

4. Active learning in sequence labeling

5. Semi-supervised active learning in sequence labeling

6. Experiment

7. Summary
Introduction
Sequence modeling and labeling problem definition

- Sequence of states
- Sequence of observations

Obrázek 1: Sequence representation
1. Sequence modeling
   - Given a sequence of states/labels and sequence of observations, find a model that the most likely generates the sequences.

2. Sequence labeling
   - Given a sequence of observations, determine an appropriate label/state for each observation.
   - Reducing errors by considering relations.
Applications

- Handwriting recognition
- Facial expression dynamic modeling
- DNA analysis
- Part-of-speech tagging
- Speech recognition
- Video analysis
Active learning
What is active learning?

- The quality of labels makes a huge difference. 
  Garbage in, garbage out.
- Obtaining "golden" annotation data can be really expensive.
What is active learning?

Obrázek 2: Active learning cycle
Obrázek 3: Active learning scenarios
Data:
L - set of labeled examples
U - set of unlabeled examples
\( \theta \) - utility function

while stopping criterion is not met do
  1. learn model \( M \) from \( L \);
  2. for all \( x_i \in U \) : \( u_{x_i} \leftarrow \theta_M(x_i) \);
  3. select example \( x^* \in U \) with the highest utility function \( u_i \);
  4. query annotator for label of example \( x^* \);
  5. move \( < y, x^* > \) to \( L \);
end

return \( L \)

Algorithm 1: General pool-based AL framework
Query strategies frameworks

1. Uncertainty Sampling
2. Query-By-Committee
3. Expected Model Change
4. Expected Error Reduction
5. Variance Reduction
6. Density-Weighted Methods
Frameworks: Uncertainty Sampling

- Simplest, most commonly used
- Intuitive for probabilistic learning model
- Binary problems: choose instance with posterior probability near to 0.5
- Multiclass problems:
  - Least confident - $x_{LC}^* = \arg \max_x 1 - P_\theta(\hat{y}|x)$
  - Margin sampling - $x_{M}^* = \arg\min_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$
  - Entropy - $x_{H}^* = \arg\max_x - \sum_i P_\theta(y_i|x) \log P_\theta(y_i|x)$
Frameworks: Uncertainty Sampling

Obrázek 4: Uncertainty sampling for three-class classification problem

- Application dependent
- Entropy - minimizing log-loss
- LC + Margin - minimizing classification error
Frameworks: Query-By-Committee

- We maintain a committee $\mathcal{C} = \{\theta_1, ..., \theta_C\}$ of models trained on $\mathcal{L}$
- The most informative query is considered to be the instance about which they most disagree.
- We need to ensure variability of models in the beginning
- Measure of disagreement:
  - Vote entropy - $x_{VE}^* = \arg\max_x \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$
  - Kullback-Leibler divergence - $x_{KL}^* = \arg\max_x \frac{1}{C} \sum_{c=1}^{C} D_{KL}(P_{\theta(c)} || P_{\mathcal{C}})$
Frameworks: Expected Model Change

- Capable for models using gradient based training.
- Query instance which would cause the largest model change.
- Use a gradient of the objective function $\nabla \ell_\theta(\mathcal{L})$

$$x_{EMC}^* = \arg\max_x \sum_i P_\theta(y_i|x) \parallel \nabla \ell_\theta(\mathcal{L} \cup <y_i,x>) \parallel$$

- Note: $\parallel \nabla \ell_\theta(\mathcal{L}) \parallel$ should be close to zero therefore we can use an approximation $\parallel \nabla \ell_\theta(\mathcal{L} \cup <y_i,x>) \parallel \approx \parallel \nabla \ell_\theta(<y_i,x>) \parallel$
Frameworks: Expected Error Reduction

- Estimate the expected future error of a model trained on $\mathcal{L} \cup <x, y>$

- Methods:
  - Minimizing the expected 0/1-loss
    \[
    x^* = \arg\min_x \sum_i P_\theta(y_i|x) \left( \sum_{u=1}^U 1 - P_{\theta+<x,y_i>} (\hat{y}|x^{(u)}) \right)
    \]
  - Minimizing the expected log-loss
    \[
    x^* = \arg\min_x \sum_i P_\theta(y_i|x) \left( - \sum_{u=1}^U \sum_j P_{\theta+<x,y_i>} (y_j|x^{(u)}) \log P_{\theta+<x,y_i>} (y_j|x^{(u)}) \right)
    \]
  - In most cases the most computationally expensive query framework
    - Logistic regression - $\mathcal{O}(ULG)$
    - CRF - $\mathcal{O}(TM^{T+2}ULG)$
Frameworks: Variance Reduction

- Use the bias-variance decomposition
- \[ E_T[(\hat{y} - y)^2 | x] = \]
  \[ E[(y - E[y|x])^2] + (E_L[\hat{y}] - E[y|x])^2 + E_L[(\hat{y} - E_L[\hat{y}])^2] \]
- Model dependent framework
Frameworks: Density-Weighted Methods

- Informative instances should not only be those which are uncertain, but also those which are "representative" of the underlying distribution.
- Uses one of other query strategies as base query strategy (e.g. uncertainty sampling).
- \( x^* = \argmax_x \phi_A(x) \times \left( \frac{1}{U} \sum_{u=1}^{U} \text{sim}(x, x^{(u)}) \right)^\beta \)
- The method is more robust to outliers in dataset.
Active learning problem variants

- **Active Learning for Structured Outputs**
  - Instance is not represented by a single feature vector, but rather a structure.
  - e.g.: Sequences, trees, grammars.

- **Active Feature Acquisition**
  - Selection of salient unused features
  - e.g.: Medical tests, sensitive information

- **Active Class Selection**
  - Learner is allowed to query a known class label, and obtaining each instance incurs a cost.

- **Active Clustering**
  - Generate (or subsample) instances in such a way that they self-organize into groupings
  - Try to get less overlap or noise than with random sampling
Active learning problem variants

- **Active Learning for Structured Outputs**
  - Instance is not represented by a single feature vector, but rather a structure.
  - e.g.: Sequences, trees, grammars.

- **Active Feature Acquisition**
  - Selection of salient unused features
  - e.g.: Medical tests, sensitive information

- **Active Class Selection**
  - Learner is allowed to query a known class label, and obtaining each instance incurs a cost.

- **Active Clustering**
  - Generate (or subsample) instances in such a way that they self-organize into groupings
  - Try to get less overlap or noise than with random sampling
Graphical models
Models: Markov model

Obrázek 5: Markov model/chain
Models: Hidden Markov model

Obrázek 6: Hidden Markov model
Models: Hidden Markov model

\[ \lambda = (A, B, \pi) \]

- Set of hidden states \( Y = \{y_1, y_2, ..., y_N\} \), set of observable values \( X = \{x_1, x_2, ..., x_M\} \)
- Sequence of states \( Q = q_1 q_2 q_3 ... q_T \) sequence of outputs \( O = o_1 o_2 o_3 ... o_T \)
- Transition probability matrix \( A = \{a_{ij}\} \)
  \[ a_{ij} = P(q_t = y_j | q_{t-1} = y_i), \ 1 \leq i, j \leq N \]
- Emission probability distribution \( B = \{b_{i,j}\} \)
  \[ b_{i,j} = P(o_t = x_j | q_t = y_i), \ 1 \leq i \leq N, \ 1 \leq i \leq M \]
- Initial probability distribution \( \pi = \{\pi_i\} \)
  \[ \pi_i = P(q_1 = y_i), \ 1 \leq i \leq N \]
Models: Conditional random field (linear chain)

- Discriminative model $P(Y|X)$, we do not explicitly model $P(X)$.
- Perform better than HMMs when the true data distribution has higher-order dependencies than the model.

$$P(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \exp \left( \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right)$$

$$Z(X) = \sum_{Y} \prod_{t=1}^{T} \exp \left( \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right)$$

Obrázek 7: Linear chain conditional random field
1. HMM
   $P(Q, O) \propto \prod_{t=1}^{T} P(q_t | q_{t-1}) P(o_t | q_t)$

2. CRF
   $P(Q|O) \propto \frac{1}{Z_O} \prod_{t=1}^{T} \exp \left( \sum_j \lambda_j f_j(q_t, q_{t-1}) + \sum_k \mu_k g_k(q_t, o_t) \right)$
Active learning in sequence labeling
Uncertainty sampling

• Least confident
  • $x_{LC}^* = \arg\max_x 1 - P_\theta(\hat{y}|x)$
  • Viterbi path

• Margin sampling
  • $x_M^* = \arg\min_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$
  • N-best algorithm

• Entropy
  • Token entropy - $x_{TE}^* = \arg\max_x -\frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} P_\theta(y_t = m) \log P_\theta(y_t = m)$
  • Total token entropy
  • Sequence entropy $x_{SE}^* = \arg\max_x - \sum_{\hat{y}} P_\theta(\hat{y}|x) \log P_\theta(\hat{y}|x)$
  • N-best sequence entropy $x_{SE}^* = \arg\max_x - \sum_{\hat{y} \in \mathcal{N}} P_\theta(\hat{y}|x) \log P_\theta(\hat{y}|x)$
Query by Committee

- Query-by-bagging (each model has unique modified set $L^{(c)}$)
- Vote entropy - disagreement over Viterbi’s paths
  \[ x_{VE}^* = \arg\max_x \left( -\frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} \frac{V(y_t,m)}{C} \log \frac{V(y_t,m)}{C} \right) \]
- Kullback Leibler
  \[ x_{KL}^* = \arg\max_x \left( \frac{1}{T} \sum_{t=1}^{T} \frac{1}{C} \sum_{c=1}^{C} D_{KL}(\theta^{(c)} \| C) \right) \]
  \[ D(\theta^{(c)} \| C) = \sum_{m=1}^{M} P_{\theta^{(c)}}(y_t = m) \log \frac{P_{\theta^{(c)}}(y_t = m)}{P_C(y_t = m)} \]
- Non-normalized variants
- Sequence vote entropy
  \[ x_{SVE}^* = \arg\max_x \left( -\sum_{\hat{y} \in \mathcal{N}^c} P(\hat{y}|x,C) \log P(\hat{y}|x,C) \right) \]
- Sequence Kullback-Leibler
  \[ x_{SKL}^* = \arg\max_x \left( \frac{1}{C} \sum_{c=1}^{C} \sum_{\hat{y} \in \mathcal{N}^c} \log \frac{P_{\theta^{(c)}}(\hat{y}|x)}{P_C(\hat{y}|x)} \right) \]
Expected Gradient Length

- Exception over the N-best labelings.
- $x^*_{EMC} = \arg\max_x \sum_{\hat{y} \in N^C} P_\theta(\hat{y} | x) \| \nabla \ell_\theta(L \cup < y_i, x >) \|$
• $O(TM^{T+2}ULG)$ too expensive :(
Density-Weighted Methods

- Information density
  - solves problem that US and QBC are prone to querying outliers
- We need a distance measure for sequences.
  - Kullback-Leibler
  - Euclidean distance
  - Cosine distance
- Drawback: number of required similarity calculations grows quadratically with the number of instances in $U$.
- Solution: Precompute them.
Semi-supervised active learning in sequence labeling
Fully supervised vs semi-supervised

- **FuSAL:**
  - Sequence is handled as a whole unit.
  - Sequence-wise vs. token-wise utility functions

- **SeSAL**
  - Some subsequences can be easily labelled automatically.
  - Decrease labelling effort.
  - Usage of self-training principle.
Algorithm 2: General AL framework

Data:
B - number of examples to be selected
L - set of labeled examples
U - set of unlabeled examples
θ - utility function

while stopping criterion is not met do
  1. learn model M from L;
  2. for all $x_i \in U : u_{x_i} \leftarrow \theta_M(x_i)$;
  3. select B examples $x_i \in U$ with highest utility function $u_i$;
  4. annotate sequences using M;
  5. query for labels of non-confidential tokens;
  6. move newly annotated examples to L;
end
return L
Experiment
Problem definition

- "Handwritten" letters recognition.
- Each letter is randomly written in one of 6 fonts
- Downscaled to 4x4 pixels
- Letters are organized in real sentences.
Model and training

- Linear chain conditional random fields.
- 3 labeled sentences in train dataset in the beginning.
- Labeled sentences are added to the dataset iteratively (50 times).
  - Random choice
  - FuSAL (Least Confident)
  - SeSAL (Least Confident + marginal probability)
Results

Does it even worth it?

![Graph showing comparison of methods: Random, FuSAL, SeSAL across sentence queries]

- Method: Random
- Method: FuSAL
- Method: SeSAL

Accuracy (acc) vs. sentence_queries
Let’s look from another point of view.
Results
Summary
Summary

- It does not work in all cases.
- It can be implementation overhead.
- In the most of cases the active learning helps.
- It can be applied to different structures like sequences or trees.
- The combination with semi-supervised learning can lead to rapid save of costs.
- Future work: Different query costs, automatic threshold finding, CT-HMM.
Thank you for your attention.
Questions?