More Reductions: Collective Classification & Learning to Rank

CMSC 422
Slides adapted from Prof. CARPUAT
Topics

• Fundamental ML concept: reductions
  – re-use simple and efficient classification algorithms to perform more complex tasks

• We’ve seen 2 examples of reductions already
  – Weighted binary classification
  – Multiclass classification

• Today: 2 new examples
  – Collective classification
  – Learning to rank
A taste of more complex problems: Collective Classification

• Prediction problems where we want to make **multiple correlated predictions** simultaneously

• Examples:
  – object detection in an image
  – finding part of speech of words in a sentence
Collective vs. multiclass classification

• We can encode dependencies/correlations
  – Between input features only in multiclass classification
  – Between output predictions in collective classification
Structured output spaces: examples

Part-of-speech tagging.

the/D cat/N bit/V the/D dog/N

To score a candidate tagging \( y \) of a sentence \( x \), add up:

- Score for each (word, tag)
- Score for each trigram (tag1, tag2, tag3)
- Other such component scores

Inaccurate to treat each tag as a separate prediction problem.

To tag a given sentence \( x \): find the tagging \( y \) with maximum score. Can be done efficiently by dynamic programming.
Structured output spaces: examples

Parts-based object recognition.
TASK: COLLECTIVE CLASSIFICATION

Given:

1. An input space $\mathcal{X}$ and number of classes $K$

2. An unknown distribution $\mathcal{D}$ over $\mathcal{G}(\mathcal{X} \times [K])$

Compute: A function $f : \mathcal{G}(\mathcal{X}) \to \mathcal{G}([K])$ minimizing:

$$\mathbb{E}_{(V,E) \sim \mathcal{D}} \left[ \sum_{v \in V} \left[ \hat{y}_v \neq y_v \right] \right],$$

where $y_v$ is the label associated with vertex $v$ in $G$ and $\hat{y}_v$ is the label predicted by $f(G)$. 
How to solve collective classification?

- One possible solution: stacking
  - reduces to binary/multiclass classification
  - Learn a stack of K classifiers
    - Classifier at level k uses output of lower level classifiers (k-1, k-2, etc.)

- Many other solutions exist
  - but are out of scope for 422
  - e.g. structured prediction, graphical models, hidden markov models
Algorithm 20 $\textbf{StackTrain}(\mathcal{D}^{cc}, K, \textbf{MulticlassTrain})$

1: $\mathbf{D}_{mc}^{mc} \leftarrow [\ ]$ // our generated multiclas data
2: $\hat{Y}_{k,n,v} \leftarrow 0, \forall k \in [K], n \in [N], v \in G_n$ // initialize predictions for all levels
3: for $k = 1$ to $K$ do
4: for $n = 1$ to $N$ do
5: for all $v \in G_n$ do
6: $(x, y) \leftarrow \text{features and label for node } v$
7: $x \leftarrow x \oplus \hat{Y}_{l,n,u}, \forall u \in \mathcal{N}(v), \forall l \in [k-1]$ // add on features for
8: // neighboring nodes from lower levels in the stack
9: $\mathbf{D}_{mc}^{mc} \leftarrow \mathbf{D}_{mc}^{mc} \oplus (y, x)$ // add to multiclas data
10: end for
11: end for
12: $f_k \leftarrow \textbf{MulticlassTrain}(\mathbf{D}_{mc}^{mc})$ // train $k$th level classifier
13: for $n = 1$ to $N$ do
14: $\hat{Y}_{k,n,v} \leftarrow \textbf{StackTest}(f_1, \ldots, f_k, G_n)$ // predict using $k$th level classifier
15: end for
16: end for
17: return $f_1, \ldots, f_K$ // return all classifiers
Algorithm 21 \textbf{StackTest}($f_1, \ldots, f_K, G$)

1: $\hat{Y}_{k,v} \leftarrow 0, \forall k \in [K], v \in G$ \hspace{1cm} // initialize predictions for all levels
2: \textbf{for} $k = 1$ \textbf{to} $K$ \textbf{do}
3: \hspace{1cm} \textbf{for all} $v \in G$ \textbf{do}
4: \hspace{2cm} $x \leftarrow$ features for node $v$
5: \hspace{2cm} $x \leftarrow x \oplus \hat{Y}_{l,u}, \forall u \in \mathcal{N}(v), \forall l \in [k - 1]$ \hspace{1cm} // add on features for
6: \hspace{2cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} // neighboring nodes from lower levels in the stack
7: \hspace{2cm} $\hat{Y}_{k,v} \leftarrow f_k(x)$ \hspace{1cm} // predict according to $k$th level
8: \hspace{1cm} \textbf{end for}
9: \textbf{end for}
10: \textbf{return} $\{\hat{Y}_{K,v} : v \in G\}$ \hspace{1cm} // return predictions for every node from the last layer
Stacking issues

• Very sensitive to overfitting
  – If $f_1$ makes perfect predictions on the training set, the other classifiers don’t have much to learn from

• Solutions
  – Cross-validation on the training data
  – *Regularize* base learner heavily
Ranking

• Canonical example: web search

• Given all the documents on the web

• For a user query, retrieve relevant documents, ranked from most relevant to least relevant