Final review

LING572
Advanced Statistical Methods for NLP
March 12, 2020

Topics covered

- Supervised learning: eight algorithms
  - kNN, NB: training and decoding
  - DT: training and decoding (with binary features)
  - MaxEnt: training (GIS) and decoding
  - SVM: decoding, tree kernel
  - CRF: introduction
  - NN and backprop: introduction, training and decoding; RNNs, transformers

Other topics

- From LING570:
  - Introduction to classification tasks
  - Mallet
  - Beam search
- Information theory: entropy, KL divergence, info gain
- Feature selection: e.g., chi-square, feature frequency
- Sequence labeling problems
- Reranking

Assignments

- Hw1: Probability and Info theory
- Hw2: Decision tree
- Hw3: Naïve Bayes
- Hw4: kNN and chi-square
- Hw5: MaxEnt decoder
- Hw6: Beam search
- Hw8: SVM decoder
- Hw9, Hw10: Neural Networks

Main steps for solving a classification task

- Formulate the problem
- Define features
- Prepare training and test data
- Select ML learners
- Implement the learner
- Run the learner
  - Tune hyperparameters on the dev data
  - Error analysis
  - Conclusion

Learning algorithms

Generative vs. discriminative models

- Joint (generative) models estimate P(x,y) by maximizing the likelihood: $P(X,Y|\mu)$
  - Ex: n-gram models, HMM, Naïve Bayes, PCFG
  - Training is trivial: just use relative frequencies.

- Conditional (discriminative) models estimate P(y|x) by maximizing the conditional likelihood: $P(Y|X, \mu)$
  - Ex: MaxEnt, SVM, CRF, etc.
  - Training is harder

Parametric vs. non-parametric models

- Parametric model:
  - The number of parameters do not change w.r.t. the number of training instances
  - Ex: NB, MaxEnt, linear SVM

- Non-parametric model:
  - More examples could potentially mean more complex classifiers.
  - Ex: kNN, non-linear SVM

Feature-based vs. kernel-based

- Feature-based:
  - Representing x as a feature vector
  - Need to define features
  - Ex: DT, NB, MaxEnt, TBL, CRF, …

- Kernel-based:
  - Calculating similarity between two objects
  - Need to define similarity/kernel function
  - Ex: kNN, SVM

DT

- DT:
  - Training: build the tree
  - Testing: traverse the tree

- Uses the greedy approach:
  - DT chooses the split that maximizes info gain, etc.

NB and MaxEnt

- NB:
  - Training: estimate P(c) and P(f | c)
  - Testing: calculate P(y) P(x | y)

- MaxEnt:
  - Training: estimate the weight for each (f, c)
  - Testing: calculate P(y | x)

- Differences:
  - generative vs. discriminative models
  - MaxEnt does not assume features are conditionally independent

kNN and SVM

- Both work with data through "similarity" functions between vectors.

- kNN:
  - Training: Nothing
  - Testing: Find the nearest neighbors

- SVM
  - Training: Estimate the weights of training instances ➔ w and b
  - Testing: Calculating f(x), which uses all the SVs

MaxEnt and SVM

- Both are discriminative models.

- Start with an objective function and find the solution to an optimization problem by using
  - Lagrangian, the dual problem, etc.
  - Iterative approach: e.g., GIS
  - Quadratic programming

  → numerical optimization

HMM, MaxEnt and CRF

- Linear-chain CRF is like HMM + MaxEnt
  - Training is similar to training for MaxEnt
  - Decoding is similar to Viterbi for HMM decoding
  - Features are similar to the ones for MaxEnt

Comparison of three learners

	Naïve Bayes	MaxEnt	SVM
Modeling	Maximize $P(X,Y\|\theta)$	Maximize $P(Y\|X, \theta)$	Maximize the minimal margin
Training	Learn P(c) and P(f|c)	Learn λ_i for feature function	Learn α_i for each (x_i, y_i)
Decoding	Calc P(y) P(x | y)	Calc P(y | x)	Calc f(x)
Things to decide	Features Delta for smoothing	Features Regularization Training algorithm	Kernel function Regularization Training algorithm C for penalty

NNs

- No need to choose features or kernels, choose an architecture instead
- Objective function: e.g., mean squared errors, cross entropy
- Training: learn weights and biases via SGD + backprop
- Testing: one forward pass
- RNNs + Transformers (and transfer learning)

Questions for each method

- Modeling:
    - what is the objective function?
    - How does decomposition work?
    - What kind of assumptions are made?
    - How many model parameters?
    - How many hyperparameters?
    - How to handle multi-class problem?
    - How to handle non-binary features?
    - …

Questions for each method (cont'd)

- Training: estimating parameters

- Decoding: finding the "best" solution

- Weaknesses and strengths:
  - parametric?
  - generative/discriminative?
  - performance?
  - robust? (e.g., handling outliners)
  - prone to overfitting?
  - scalable?
  - efficient in training time? Test time?

Implementation Issues

Implementation Issues

- Taking the log:

$$logP(X_1, ..., X_n) = log \prod_i P(X_i|X_1, ..., X_{i-1})$$
$$= \sum_i logP(X_i|X_1, ..., X_{i-1})$$

- Ignoring constants:

$$P(d_i|c) = P(|d_i|)|d_i|! \prod_{k=1}^{|V|} \frac{P(w_k|c)^{N_{ik}}}{N_{ik}!}$$

- Increasing small numbers before dividing

$$P(c1|x) = \frac{P(x,c1)}{P(x)} = \frac{P(x,c1)}{P(x,c1)+P(x,c2)+...}$$

$logP(x, c_1)$ is -200, $logP(x, c_2)$ is -201.

Implementation Issues (cont'd)

- Reformulating the formulas: e.g., Naïve Bayes

$$P(x, c)$$

$$= P(c) \prod_{w_k \in d_i} P(w_k|c) \prod_{w_k \notin d_i} (1 - P(w_k|c))$$

$$= P(c) \prod_{w_k \in d_i} \frac{P(w_k|c)}{1-P(w_k|c)} \prod_{w_k} (1 - P(w_k|c))$$

- Storing the useful intermediate results

$$\prod_{w_k} (1 - P(w_k|c))$$

An example: calculating model expectation in MaxEnt

$$E_p f_j = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} p(y \mid x_i) f_j(x_i, y)$$

for each instance x

calculate P(y|x) for every y in Y

for each feature t in x

for each y in Y

model_expect [t] [y] += 1/N * P(y|x)

What’s next?

What's next?

- Course evaluations:
  - Overall: open until **03/13!!!**
  - For TA: you should have received an email
  - Please fill out both.
- Hw9: Due 11pm on 3/19

What’s next (beyond ling572)?

- Supervised learning:
  - Covered algorithms: e.g., L-BFGS for MaxEnt, training for SVM, building a complex NN
  - Other algorithms: e.g., Graphical models, Bayes Nets

- Using algorithms:
  - Formulate the problem
  - Select features, kernels, or architecture
  - Choose/compare ML algorithms

What's next? (cont'd)

- Semi-supervised learning: labeled data and unlabeled data
  - Analysis / interpretation
- Using them for real applications: LING573
- Ling575s:
  - Representation Learning
  - Mathematical Foundations
  - Information Extraction
- Machine learning, AI, etc.
- Next spring: new deep learning for NLP course by yours truly