Methods and metrics: Model evaluation

Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding

Overview

- Baselines
- Hyperparameter optimization
- Classifier comparison
- Assessing models without convergence
- The role of random parameter initialization

Baselines

Evaluation numbers can never be understood properly in isolation:

1. Your system gets 0.95 F1! Is your task too easy?
2. Your system gets 0.60 F1. But what do humans get?

Baselines are crucial for strong experiments

- Defining baselines should not be an afterthought, but rather central to how you define your overall hypotheses.
- Baselines are essential to building a persuasive case.
- They can also be used to illuminate specific aspects of the problem and specific virtues of your proposed system.

Random baselines

Random baselines are almost always useful to include. sklearn:

- `DummyClassifier`
  - `stratified`
  - `uniform`
  - `most_frequent`
- `DummyRegressor`
  - `mean`
  - `median`

Task-specific baselines

It is worth considering whether your problem suggests a baseline that will reveal something about the problem or the ways it is modeled.

Two recent examples from NLU:

- NLI: Hypothesis-only baselines.
- The Story Cloze task: Distinguish between a coherent and incoherent ending for a story. Systems that look only at the ending options can do really well (Schwartz et al. 2017).

Hyperparameter optimization

Discussed in our unit on sentiment analysis. Rationales:

- Obtaining the best version of your model.
- Conducting fair comparisons between models.
- Understanding the stability of your architecture.

All hyperparameter tuning must be done only on train and development data.

The ideal hyperparameter optimization setting

1. For each hyperparameter, identify a large set of values for it.
2. Create a list of all the combinations of all the hyperparameter values. This will be the cross-product of all the values for all the features identified at step 1.
3. For each of the settings, cross-validate it on the available training data.
4. Choose the settings that did best in step 3, train on all the training data using those settings, and then evaluate that model on the test set.

An example

1. Parameter h_1 has 5 values.
2. Parameter h_2 has 10 values.
3. Total settings: $5 \cdot 10 = 50$.
4. Add h_3 with 2 values.
5. Total settings: $5 \cdot 10 \cdot 2 = 100$.
6. 5-fold cross-validation to select optimal parameters: 500 runs

Practical considerations

The above is untenable as a set of laws for the scientific community.

If we adopted it, then complex models trained on large datasets would end up disfavored, and only the very wealthy would be able to participate.

Rajkomar et al. (2018):

"the performance of all above neural networks were [sic] tuned automatically using Google Vizier [35] with a total of $> 201,000$ GPU hours"

Reasonable compromises

Pragmatic steps you can take to alleviate this problem, in descending order of attractiveness:

1. Random sampling and guided sampling allow you to explore a large space on a fixed budget of runs.
2. Search based on a few epochs of training. (Could be bolstered with short learning curves for different settings.)
3. Search based on subsets of the data. (However, some parameters will be very dependent on dataset size, so this can be risky.)
4. Via heuristic search, determine which hyperparameters matter less, and set them by hand. (Justify this in the paper!)
5. Find optimal hyperparameters via a single split and use them for all the subsequent splits. Justified if the splits are similar.
6. Adopt others' choices. The skeptic will complain that these findings don't translate to your new data sets, but it could be the only option.

Tools for hyperparameter search

- `from sklearn.model_selection import GridSearchCV, RandomizedSearchCV, HalvingGridSearchCV`
- scikit-optimize offers a variety of methods for guided search through the grid of hyperparameters.

Classifier comparison

Suppose you've assessed two classifier models. Their performance is probably different to some degree. What can be done to establish whether these models are different in any meaningful sense?

- Practical differences
- Confidence intervals
- Wilcoxon signed-rank test (covered in the sentiment unit)
- McNemar's test (covered in the sentiment unit)

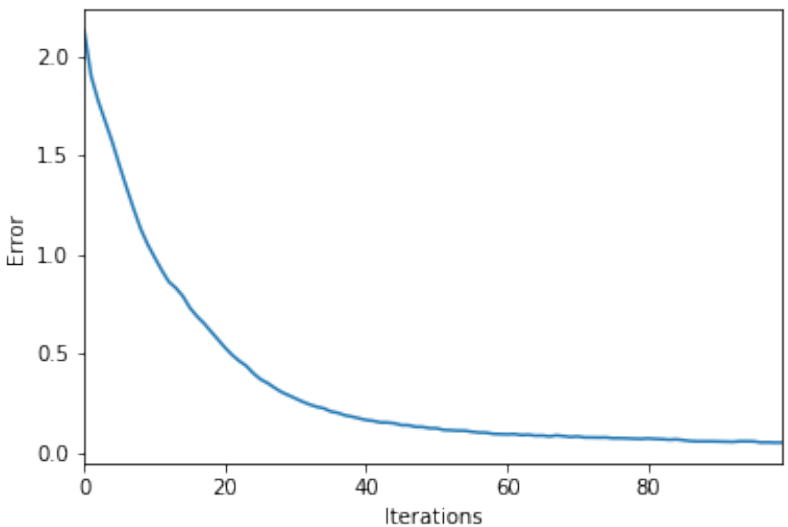
Assessing models without convergence

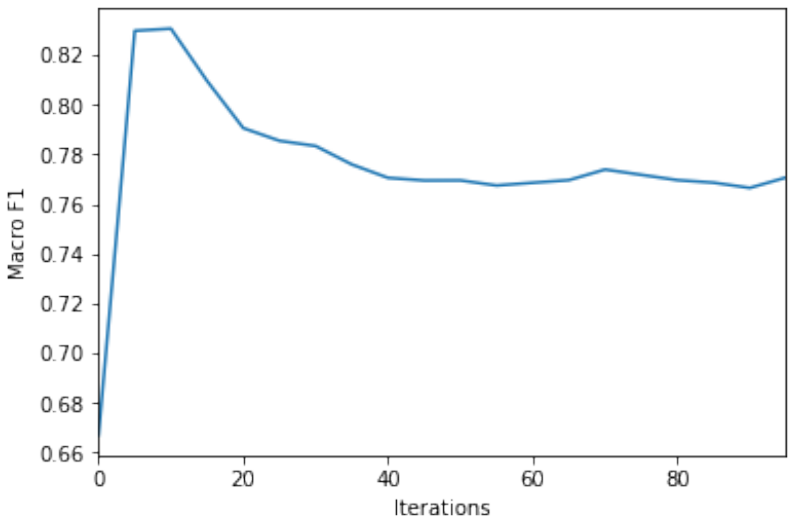
- When working with linear models, convergence issues rarely arise.
- With neural networks, convergence takes center stage:
  - The models rarely converge.
  - For they converge at different rates between runs.
  - Their performance on the test data is often heavily dependent on these differences.
- Sometimes a model with a low final error turns out to be great, and sometimes it turns out to be worse than one that finished with a higher error. Who knows?!

Incremental dev-set testing

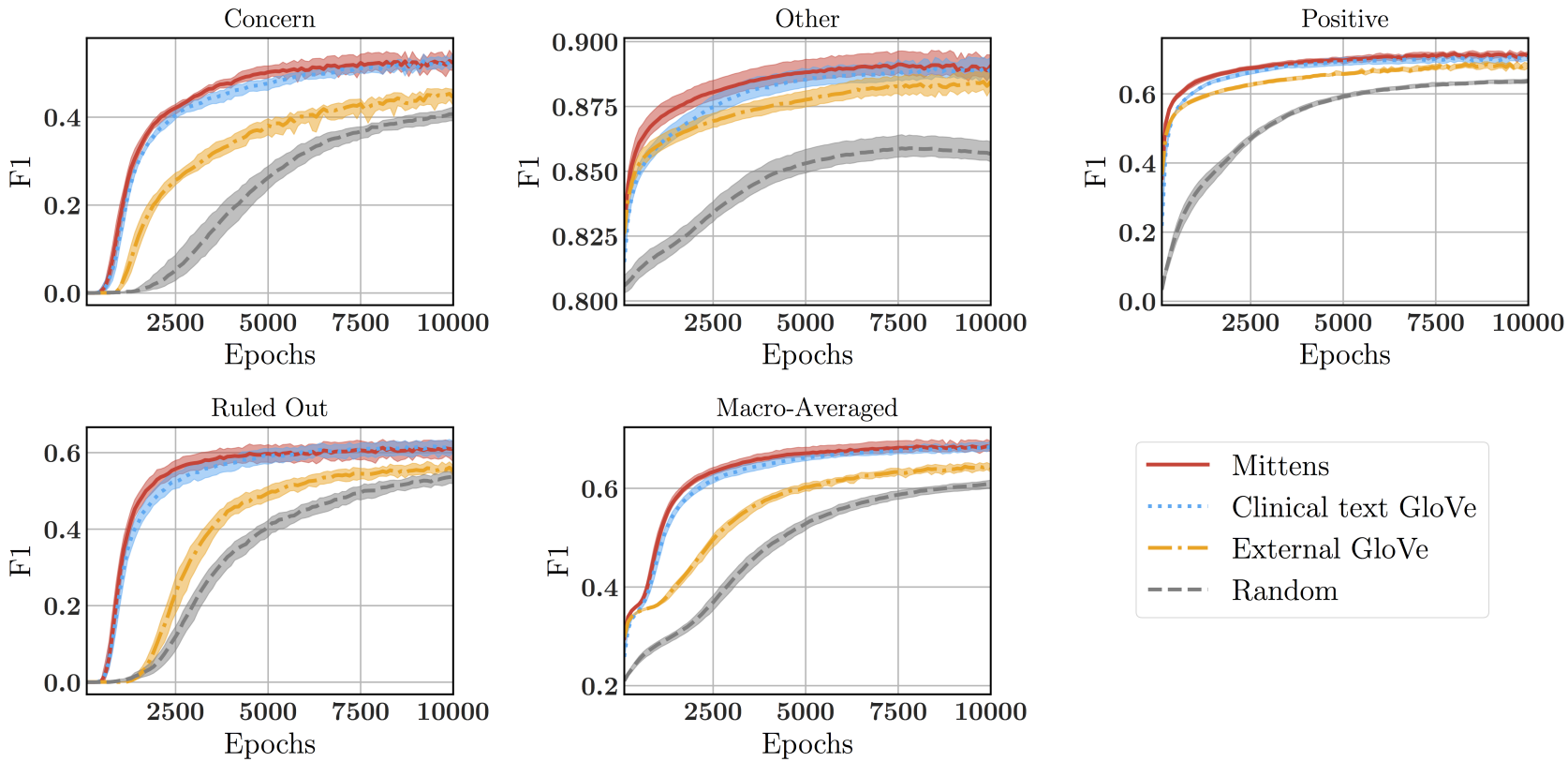
1. To address this uncertainty: regularly collect information about dev set performance as part of training.
2. For example, at every 100th iteration, one could make predictions on the dev set and store that vector of predictions.
3. All the PyTorch models for this course have an `early_stopping` with various controllable parameters.

A bit of motivation for early stopping





Learning curves with confidence intervals



Dingwall and Potts 2018

The role of random parameter initialization

1. Most deep learning models have their parameters initialized randomly
2. Clearly meaningful for non-convex optimization problems Simpler models can be impacted as well.
3. Reimers and Gurevych (2017):
   - Different initializations for neural sequence models can lead to statistically significant differences.
   - A number of recent systems are indistinguishable in terms of raw performance once this source of variation is taken into account.
4. Related: catastrophic failure as a result of unlucky initialization.
5. In `evaluation_methods.ipynb`: A feedforward network on the XOR problem succeeds 8 of 10 times.

References I

Nicholas Dingwall and Christopher Potts. 2018. Mittens: An extension of GloVe for learning domain-specialized representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 212–217, Stroudsburg, PA. Association for Computational Linguistics.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Peter J Liu, Xiaobing Liu, Mimi Sun, Patrik Sundberg, Hector Yee, et al. 2018. Scalable and accurate deep learning for electronic health records. *arXiv preprint arXiv:1801.07860*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. Story cloze task: UW NLP system. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 52–55, Valencia, Spain. Association for Computational Linguistics.