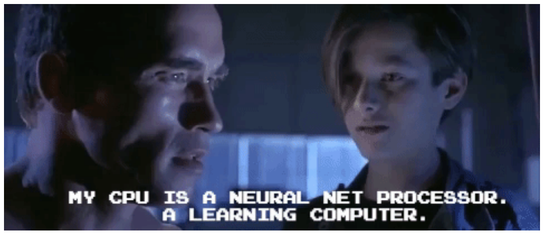
CS371N: Natural Language Processing
Lecture 5: Fairness, Neural Nets

Greg Durrett
(he/him)

TEXAS
The University of Texas at Austin



Announcements

- A1 due Thursday
- A2 released Thursday
- Fairness response (in class today) due in 1 week

Fairness

Fairness in Classification

- Classifiers can be used to make real-world decisions:
  - Who gets an interview?
  - Who should we lend money to?
  - Is this online activity suspicious?
  - Is a convicted person likely to re-offend?
- Humans making these decisions are typically subject to anti-discrimination laws; how do we ensure classifiers are *fair* in the same way?
- Many other factors to consider when deploying classifiers in the real world (e.g., impact of a false positive vs. a false negative) but we'll focus on fairness here

Fairness Response (SUBMIT ON CANVAS)

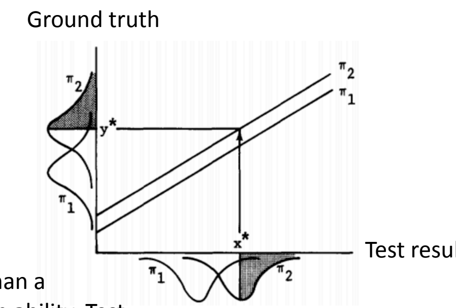
Consider having each data instance *x* associated with a **protected attribute A** when making a prediction. Example: sentiment analysis where we know the **ethnicity of the director** of the movie being reviewed. We can consider prediction as P(y | **x**, A)

- What do **you** think it would mean for a classification model to be discriminatory in this context? Try to be as precise as you can!
- Do you think our **unigram bag-of-words** model might be discriminatory according to your criterion above? Why or why not?
- Suppose we add A as an additional "word" to each example, so our bag-of-words can use it as part of the input. Do you think the unigram model might be discriminatory according to your criterion? Why or why not?
- Suppose we enforce that the model must predict at least *k*% positives across every value of A; that is, if you filter to only the data around a particular ethnicity, the model must predict at least *k*% positives on that data slice. Is this fair? Why/why not?

Fairness in Classification

Idea 1: Classifiers need to be evaluated beyond just accuracy

- T. Anne Cleary (1966-1968): a test is biased if prediction on a subgroup makes *consistent* nonzero prediction errors compared to the aggregate
- Individuals of X group could still score lower on average. But the *errors* should not be consistently impacting X
- Member of π_1 has a test result higher than a member of π_2 for the same ground truth ability. Test penalizes π_2



Hutchinson and Mitchell (2018)

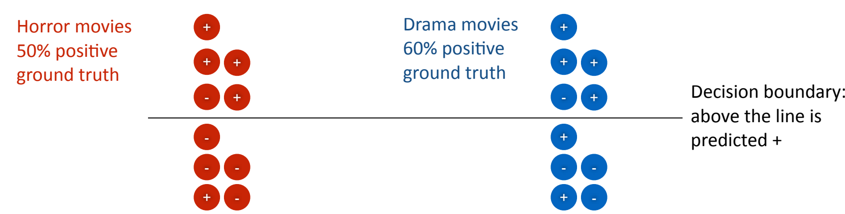
Fairness in Classification

Idea 1: Classifiers need to be evaluated beyond just accuracy

- Thorndike (1971), Petersen and Novik (1976): fairness in classification: ratio of predicted positives to ground truth positives must be approximately the same for each group ("**equalized odds**")
  - Group 1: 50% positive movie reviews. Group 2: 60% positive movie reviews

Petersen and Novik (1976)
Hutchinson and Mitchell (2018)

Fairness in Classification



- Is this classifier fair?
- Equalized odds says no, ratio of predicted positives to ground truth positives differs.
- How can we fix this?

Petersen and Novik (1976)
Hutchinson and Mitchell (2018)

Discrimination

Idea 2: It is easy to build classifiers that discriminate even *without meaning to*

- A feature might correlate with minority group X and penalize that group:
  - Bag-of-words features can identify non-English words, dialects of English like AAVE, or code-switching (using two languages). (Why might this be bad for sentiment?)
  - ZIP code as a feature is correlated with race
- Reuters: “Amazon scraps secret AI recruiting tool that showed bias against women”
  - “Women’s X” organization, women’s colleges were negative-weight features
  - Accuracy will not catch these problems, very complex to evaluate depending on what humans did in the **actual** recruiting process

Credit: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

Takeaways

- What marginalized groups in the population should I be mindful of? (Review sentiment: movies with female directors, foreign films, …)
- Can I check one of these fairness criteria?
- Do aspects of my system or features it uses introduce potential correlations with protected classes or minority groups?

Neural Networks

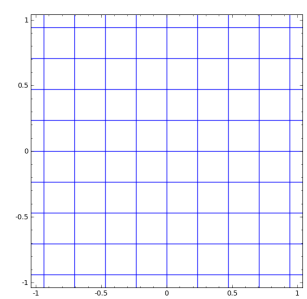
Neural Networks

$$\mathbf{z} = g(V f(\mathbf{x}) + \mathbf{b})$$

Nonlinear transformation (g) — Warp space (V) — Shift ($\mathbf{b}$)

$$y_{\text{pred}} = \text{argmax}_y \mathbf{w}_y^\top \mathbf{z}$$

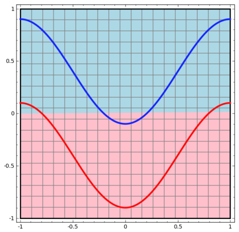
- Ignore shift / +**b** term for the rest of the course



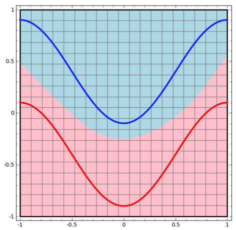
Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Neural Networks

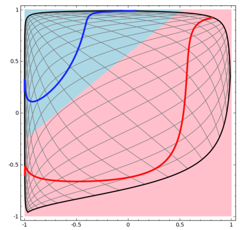
Linear classifier



Neural network



Linear classification in the transformed space!



Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

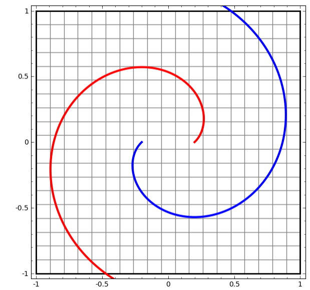
Deep Neural Networks

$$\mathbf{z}_1 = g(V_1 f(\mathbf{x}))$$

$$\mathbf{z}_2 = g(V_2 \mathbf{z}_1)$$

$$\ldots$$

$$y_{\text{pred}} = \text{argmax}_y \mathbf{w}_y^\top \mathbf{z}_n$$



Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Feedforward Networks

Vectorization and Softmax

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top f(\mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{y'}^\top f(\mathbf{x}))}$$

- Single scalar probability

- Three classes, "different weights"

$$\begin{array}{lcccc} \mathbf{w}_1^\top f(\mathbf{x}) & & -1.1 & & 0.036 \\ \mathbf{w}_2^\top f(\mathbf{x}) & = & 2.1 & \xrightarrow{\text{softmax}} & 0.89 \\ \mathbf{w}_3^\top f(\mathbf{x}) & & -0.4 & & 0.07 \end{array}$$

class probs

- Softmax operation = "exponentiate and normalize"
- We write this as: $\text{softmax}(W f(\mathbf{x}))$

Logistic Regression as a Neural Net

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top f(\mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{y'}^\top f(\mathbf{x}))}$$

- Single scalar probability

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W f(\mathbf{x}))$$

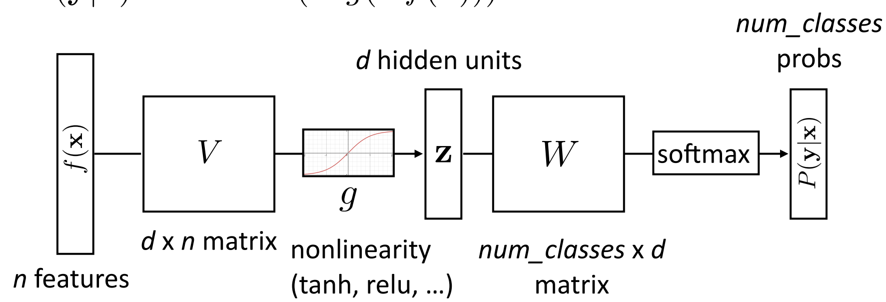
- Weight vector per class;
  W is [num classes x num feats]

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$

- Now one hidden layer

Neural Networks for Classification

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$



Backpropagation (in picture form)

Training Objective

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$

- Consider the log likelihood of a single training example:

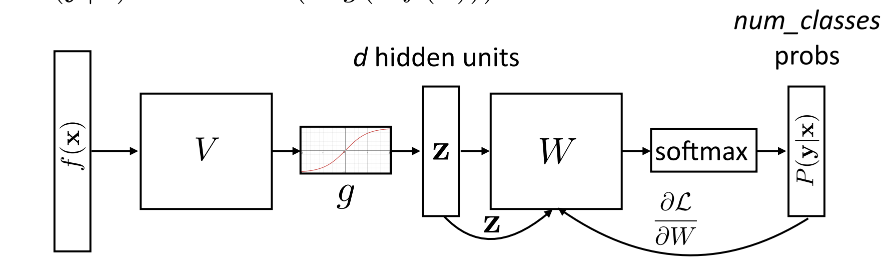
$$\mathcal{L}(\mathbf{x}, i^*) = \log P(y = i^*|\mathbf{x})$$

where *i** is the index of the gold label for an example

- Backpropagation is an algorithm for computing gradients of *W* and *V* (and in general any network parameters)

Backpropagation: Picture

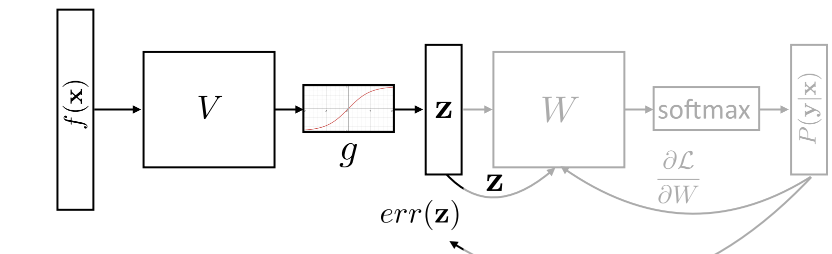
$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(W g(V f(\mathbf{x})))$$



- Gradient w.r.t. *W*: looks like logistic regression, can be computed treating **z** as the features

Backpropagation: Picture

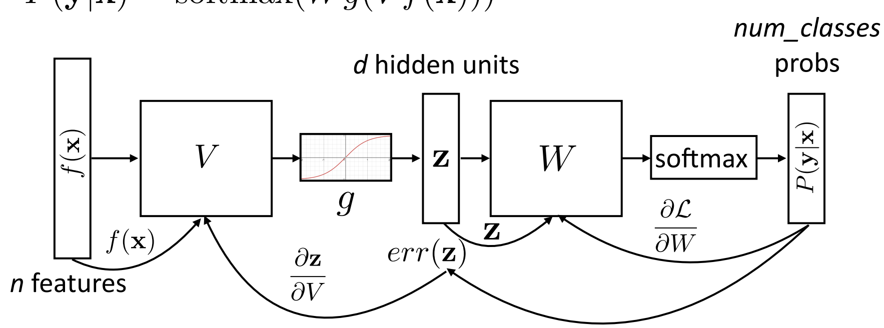
$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(W g(V f(\mathbf{x})))$$



- Can forget everything after **z**, treat it as the output and keep backpropping

Backpropagation: Picture

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(W g(V f(\mathbf{x})))$$



- Combine backward gradients with forward-pass products