Recurrent tweets
Project presentation

Mathias Berglund, Petri Kyröläinen, Yu Shen
December 9, 2013
Project background – tweet sentiment classification

Model part 1: Learning word representations

Model part 2: Classification with recurrent neural network
Goal to classify whether tweet has positive, negative or neutral sentiment

Goal to create model for classifying tweets

- Tweets can have multiple sentiments
- Model is to classify tweets into positive, neutral or negative sentiment

“Pretty Little Liars was the shit! I can't wait til tomorrow! I wanna see who all innocent & who got something to do with Allison dying!”

Positive tweet

“@Duffy_Louise Nooooooo this Sunday is the last episode of Downton Abbey. :( There's a Christmas special coming but that's AGES away.”

Negative tweet

“Manchester United will try to return to winning ways when they face Arsenal in the Premier League at Old Trafford on Saturday.”

Neutral tweet
We use data from "SemEval-2013: Sentiment Analysis in Twitter" with annotated tweets

SemEval-2013 a sentiment analysis challenge in summer 2013

SemEval-2013 had multiple challenges
• Multiple challenges in SemEval-2013
• Data included Tweets and text messages
• Tasks included message and word classification

SemEval-2013 workshop co-located with NAACL
• Organized with “The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies” in summer of 2013

Twitter data includes 7 485 annotated tweets

Twitter data set includes 7 463 annotated tweets
• Hand annotated tweets classified into “positive”, “negative”, “objective” or “neutral”
• Split into 6 448 training set and 1 017 “development set” (used as test set)

Vocabulary size 23 123
Our approach: two-stage training where semantics of tokens learned in first part, and second part used for classification

Part 1: Map each token (word) to a continuous-valued vector with semantic meaning

Part 2: Map a stream of words (i.e. a tweet) into sentiment

Motivation: Unsupervised training for first part enables use of unlabeled data
Results: We get an average F1-score of 42, which is still behind state-of-the-art 69 with handcrafted features. Our approach reached an F1-score of 41.75, while state-of-the-art reached an F1-score of 69.02. The training first step with a large data set is still under development. NRC-Canada SVM is used, which is a multitude of hand-crafted features used in conjunction with SVM classification.
Agenda

Project background – tweet sentiment classification

Model part 1: Learning word representations

Model part 2: Classification with recurrent neural network
Mikolov’s Recurrent Neural Network Language Model

Recurrent neural networks in mathematical notation

Computation of input, hidden and output layer activations (forward pass)

\[ x(t) = [w(t)^T \ s(t - 1)^T]^T \]

\[ s_j(t) = f \left( \sum_i x_i(t)u_{ji} \right) \]

\[ y_k(t) = g \left( \sum_j s_j(t)v_{kj} \right) \]

\[ f(z) = \frac{1}{1 + e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \]

Weight updates in Backprogogation Through Time algorithm

Weight updates (backward pass)

\[ V(t+1) = V(t) + s(t)e_o(t)^T \alpha - V(t)\beta \]

\[ U(t+1) = U(t) + \sum_{z=0}^{T} w(t-z)e_h(t-z)^T \alpha - U(t)\beta \]

\[ W(t+1) = W(t) + \sum_{z=0}^{T} s(t-z-1)e_h(t-z)^T \alpha - W(t)\beta \]

Error functions

\[ e_o(t) = d(t) - y(t) \]

\[ e_h(t-\tau-1) = d_h \left( e_h(t-\tau)^T W, t-\tau-1 \right) \]

\[ d_{hj}(x,t) = xs_j(t)(1 - s_j(t)) \]

\( \alpha = \) learning rate \n\( \beta = \) regularization parameter

Time complexity of RNNLM is fairly high

\[ O = E \times T \times [(H + 1) \times H \times \tau + H \times V] \]

- \( E \) = epochs
- \( T \) = tokens (or minipatches) in training set
- \( H \) = hidden neurons
- \( V \) = size of vocabulary
- \( \tau \) = time steps in back propagation through time algorithm

RNN language model with output layer factorized by classes

We calculate the probability of a word GIVEN the class

Conditional probability of word can be factorized

\[ P(w_i|\text{history}) = P(c_i|s(t))P(w_i|c_i, s(t)) \]

The two factors are computed as

\[ c_l(t) = g\left( \sum_j s_j(t)w_{lj} \right) \]
\[ y_c(t) = g\left( \sum_j s_j(t)v_{cj} \right) \]
Classes reduce time complexity of RNNLM considerably

**Standard**

\[ O = E \times T \times [(H + 1) \times H \times \tau + H \times V] \]

**Factorized by class**

\[ O = E \times T \times [(H + 1) \times H \times \tau + H \times (C + V_C)] \]

\(H\) = hidden neurons

\(V\) = size of vocabulary

\(C\) = classes

\(V_C\) = expected number of word types in the class

Preliminary results using RNN language model

<table>
<thead>
<tr>
<th>Hidden layer size</th>
<th>Classes</th>
<th>Log prob of valid data</th>
<th>Validation perplexity</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>500</td>
<td>-55118.9</td>
<td>488.0</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>-53708.3</td>
<td>416.5</td>
<td>13</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>-53731.1</td>
<td>417.6</td>
<td>13</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>-53827.0</td>
<td>422.1</td>
<td>13</td>
</tr>
<tr>
<td><strong>200</strong></td>
<td><strong>50</strong></td>
<td><strong>-53644.3</strong></td>
<td><strong>413.6</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>

Perplexity = average branching factor

Learning reduces perplexity a lot in the word prediction task

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Valid. Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>729.2</td>
</tr>
<tr>
<td>1</td>
<td>631.9</td>
</tr>
<tr>
<td>2</td>
<td>576.5</td>
</tr>
<tr>
<td>3</td>
<td>550.6</td>
</tr>
<tr>
<td>4</td>
<td>540.9</td>
</tr>
<tr>
<td>5</td>
<td>516.5</td>
</tr>
<tr>
<td>6</td>
<td>492.4</td>
</tr>
<tr>
<td>7</td>
<td>473.8</td>
</tr>
<tr>
<td>8</td>
<td>456.3</td>
</tr>
<tr>
<td>9</td>
<td>440.7</td>
</tr>
<tr>
<td>10</td>
<td>429.7</td>
</tr>
<tr>
<td>11</td>
<td>419.7</td>
</tr>
<tr>
<td>12</td>
<td>413.6</td>
</tr>
</tbody>
</table>

RNNLM with hidden layer size of 200 and 50 classes

Agenda

Project background – tweet sentiment classification

Model part 1: Learning word representations

Model part 2: Classification with recurrent neural network
In step 2, we train recurrent neural network in supervised fashion using only the labeled tweets

Visible layer (input)

Tanh layer (latent)

Softmax layer (output)

I  am  so  glad

# of neurons in layer

# of sentiments in data

100

200
We train the network using stochastic gradient descent and Nesterov-type momentum

Training details

1. **BPTT with Nesterov momentum**
   - Network trained with stochastic gradient descent
   - Learning rate set with ADADELTA
   - Nesterov-type momentum used with mom=0.99

2. **Weights pretrained as a language model**
   - Pretraining as a regularization tool
   - Weights initialized by predicting the next word in semantic space

3. **Training for max 100 epochs in minibatches**
   - Training in minibatches of 10
   - Training done for 100 epochs or until early-stop criterion with error on 20% validation set rising
Regularization by language model that predicts the next word, but this time in the semantic space provided by step 1.
Nesterov-type momentum was shown to improve very deep network learning considerably

Traditional momentum is "slow" to react

\[ v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t) \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]

Nesterov-type momentum reacts faster to gradient change

\[ v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t) \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]

Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning."
When evaluating the test set, we insert a tweet, and classify it based on the largest value in the output layer.

Prediction based on largest value in output layer.