Outline

• Part 0: Some backgrounds
• Part 1: Dropout as adaptive regularization
  • with applications to semi-supervised learning
  • joint work with Stefan Wager and Percy
• Part 2: Applications to structured prediction using CRFs
  • when the log-partition function cannot be easily computed
  • joint work with Mengqiu, Chris, Percy and Stefan Wager
The basics of dropout training

- Introduced by Hinton et al. in “Improving neural networks by preventing co-adaptation of feature detectors”
- For each example, randomly select features
  - zero them
  - compute the gradient, make an update
  - repeat
Empirically successful

- Dropout is important in some recent successes
  - won the ImageNet challenge [Krizhevsky et al., 2012]
  - won the Merck challenge [Dahl et al., 2012]

- Improved performance on standard datasets
  - images: MNIST, CIFAR, ImageNet, etc.
  - document classification: Reuters, IMDB, Rotten Tomatoes, etc.
  - speech: TIMIT, GlobalPhone, etc.
Lots of related works already

Variants

- DropConnect [Wan et al., 2013]
- Maxout networks [Goodfellow et al., 2013]

Analytical integration

- Fast Dropout [Wang and Manning, 2013]
- Marginalized Corrupted Features [van der Maaten et al., 2013]

Many other works report empirical gains
Outline

• Part 0: Some backgrounds

• Part 1: Dropout as adaptive regularization
  • with applications to semi-supervised learning

• Part 2: Applications to structured prediction using CRFs
  • when the log-partition function cannot be easily computed
Theoretical understanding?

- **Dropout as adaptive regularization**
  - feature noising -> interpretable penalty term
    
    \[
    \text{Loss( Dropout(data) )} = \text{Loss(data)} + \text{Regularizer(data)}
    \]

- **Semi-supervised learning**
  - feature dependent, label independent regularizer:
    
    \[
    \text{Regularizer(Unlabeled data)}
    \]
• Log likelihood (e.g., softmax classification):

\[
\log p(y|x; \theta) = x^T \theta_y - A(x^T \theta)
\]

\[
\theta = [\theta_1, \theta_2, \ldots, \theta_K]
\]
Dropout for Log-linear Models

- Log likelihood (e.g., softmax classification):
  \[
  \log p(y|x; \theta) = x^T \theta_y - A(x^T \theta)
  \]
  \[
  \theta = [\theta_1, \theta_2, \ldots, \theta_K]
  \]

- Dropout: \( \tilde{x}_j = \begin{cases} 
  2x_j & \text{with } p=0.5 \\
  0 & \text{otherwise}
\end{cases} \)

\[ \mathbb{E}[\tilde{x}] = x \]

- Dropout objective:
  \[
  \mathbb{E}[\log p(y|\tilde{x}; \theta)] = \mathbb{E}[\tilde{x}^T \theta_y] - \mathbb{E}[A(\tilde{x}^T \theta)]
  \]

- Loss:
  \[
  \text{Loss}(\text{data}) + \text{Regularizer}(\text{data})
  \]
  - Loss(Dropout(data))
Dropout for Log-linear Models

- We can rewrite the dropout log-likelihood

\[ \mathbb{E}[\log p(y|\tilde{x}; \theta)] = \mathbb{E}[\tilde{x}^T \theta_y] - \mathbb{E}[A(\tilde{x}^T \theta)] \]
\[ \log p(y|x; \theta) = x^T \theta_y - A(x^T \theta) \]
\[ -\text{Loss(Dropout(data))} = \mathbb{E}[\log p(y|\tilde{x}; \theta)] - \mathbb{E}[A(\tilde{x}^T \theta)] - A(x^T \theta) \]

- Dropout reduces to a regularizer

\[ R(\theta, x) = \mathbb{E}[A(\tilde{x}^T \theta)] - A(x^T \theta) \]
Second-order delta method

Take the Taylor expansion

\[ A(s) \approx A(s_0) + (s - s_0)^T A'(s_0) + (s - s_0)^T \frac{A''(s_0)}{2} (s - s_0) \]
Second-order delta method

Take the Taylor expansion

\[ A(s) \approx A(s_0) + (s - s_0)^T A'(s_0) + (s - s_0)^T \frac{A''(s_0)}{2} (s - s_0) \]

Substitute \( s = \tilde{s} \overset{\text{def}}{=} \theta^T \tilde{x}, \ s_0 = \mathbb{E}[\tilde{s}] \)

Take expectations to get the quadratic approximation:

\[ R^q(\theta, x) = \frac{1}{2} \mathbb{E}[(\tilde{s} - s)^T \nabla^2 A(s)(\tilde{s} - s)] \]
\[ = \frac{1}{2} \text{tr}(\nabla^2 A(s) \text{Cov}(\tilde{s})) \]
Example: logistic regression

- The quadratic approximation

\[ R^q(\theta, x) = \frac{1}{2} A''(x^T \theta) \text{Var}[\tilde{x}^T \theta] \]
Example: logistic regression

- The quadratic approximation

\[ R^q(\theta, x) = \frac{1}{2} A''(x^T \theta) \text{Var}[\tilde{x}^T \theta] \]

- \( A''(x^T \theta) = p(1 - p) \) represents uncertainty:

\[ p = p(y|x; \theta) = (1 + \exp(-yx^T \theta))^{-1} \]
Example: logistic regression

- The quadratic approximation
  \[ R^q(\theta, x) = \frac{1}{2} A''(x^T \theta) \text{Var}[\tilde{x}^T \theta] \]

- \( A''(x^T \theta) = p(1 - p) \) represents uncertainty:
  \[ p = p(y|x; \theta) = (1 + \exp(-yx^T \theta))^{-1} \]

- \( \text{Var}[\tilde{x}^T \theta] = \sum_j \theta_j^2 x_j^2 \) is L$_2$-regularization after normalizing the data
The regularizers

- Dropout on Linear Regression
  \[ R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i x_j^{(i)2} \]

- Dropout on Logistic Regression
  \[ R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i p_i(1 - p_i)x_j^{(i)2} \]

- Multiclass, CRFs [Wang et al., 2013]
Dropout intuition

\[ R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i p_i(1 - p_i)x_j^{(i)2} \]

- Regularizes “rare” features less, like AdaGrad: there is actually a more precise connection [Wager et al., 2013]
- Big weights are okay if they contribute only to confident predictions
- Normalizing by the diagonal Fisher information
Generally applicable

• We motivated by dropout, but this framework is generally applicable to any independent feature noising:
  • Additive Gaussian
  • Multiplicative Gaussian
  • Etc.
Semi-supervised Learning

- These regularizers are label-independent
  - but can be data adaptive in interesting ways
- labeled dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$
- unlabeled data $\mathcal{D}_{\text{unlabeled}} = \{u_1, u_2, \ldots, u_n\}$
- We can better estimate the regularizer

$$R^*_\theta(\theta, \mathcal{D}, \mathcal{D}_{\text{unlabeled}})$$

$$\text{def} = \frac{n}{n + \alpha m} \left( \sum_{i=1}^{n} R(\theta, x_i) + \alpha \sum_{i=1}^{m} R(\theta, u_i) \right).$$

for some tunable $\alpha$. 
Semi-supervised intuition

\[ R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i p_i (1 - p_i) x_j^{(i)2} \]

- Like other semi-supervised methods:
  - transductive SVMs [Joachims, 1999]
  - entropy regularization [Grandvalet and Bengio, 2005]
  - EM: guess a label [Nigam et al., 2000]
  - want to make confident predictions on the unlabeled data
- Get a better estimate of the Fisher information
IMDB dataset [Maas et al., 2011]

- 25k examples of positive reviews
- 25k examples of negative reviews
- Half for training and half for testing
- 50k unlabeled reviews also containing neutral reviews
- 300k sparse unigram features
- ~5 million sparse bigram features
Experiments: semi-supervised

- Add more unlabeled data (10k labeled) improves performance
Experiments: semi-supervised

- Add more labeled data (40k unlabeled) improves performance

![Graph showing accuracy vs. size of labeled data]
## Quantitative results on IMDB

<table>
<thead>
<tr>
<th>Method \ Settings</th>
<th>Supervised</th>
<th>Semi-sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB - unigrams with SFE [Su et al., 2011]</td>
<td>83.62</td>
<td>84.13</td>
</tr>
<tr>
<td>Vectors for sentiment analysis [Maas et al., 2011]</td>
<td>88.33</td>
<td>88.89</td>
</tr>
<tr>
<td>This work: dropout + unigrams</td>
<td>87.78</td>
<td>89.52</td>
</tr>
<tr>
<td>This work: dropout + bigrams</td>
<td>91.31</td>
<td>91.98</td>
</tr>
</tbody>
</table>
## Experiments: other datasets

<table>
<thead>
<tr>
<th>Dataset \ Settings</th>
<th>L₂</th>
<th>Drop</th>
<th>+Unlbl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjectivity [Peng and Lee, 2004]</td>
<td>88.96</td>
<td>90.85</td>
<td>91.48</td>
</tr>
<tr>
<td>Rotten Tomatoes [Peng and Lee, 2005]</td>
<td>73.49</td>
<td>75.18</td>
<td>76.56</td>
</tr>
<tr>
<td>20-newsgroups</td>
<td>82.19</td>
<td>83.37</td>
<td>84.71</td>
</tr>
<tr>
<td>CoNLL-2003</td>
<td>80.12</td>
<td>80.90</td>
<td>81.66</td>
</tr>
</tbody>
</table>
Outline

• Part 0: Some backgrounds
• Part 1: Dropout as adaptive regularization
  • with applications to semi-supervised learning
  • With Stefan and Percy
• Part 2: Applications to structured prediction using CRFs
  • when the log-partition function cannot be easily computed
  • with Mengqiu, Chris, Percy and Stefan
Log-linear structured prediction

- A vector of scores $s = (s_1, \ldots, s_{|Y|})$  
  $s_y = f(y, x) \cdot \theta$

- The likelihood is:
  
  $$p(y \mid x; \theta) = \exp\{s_y - A(s)\}$$

  $$A(s) = \log \sum_y \exp\{s_y\}$$

- $|Y|$ might be really huge!
What about structured prediction?

- Recall that in logistic regression:

\[ R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i p_i (1 - p_i) x_j^{(i)^2} \]

- What if we cannot easily compute the log-partition function \( A \) and its second derivatives?
The original setup

Take the Taylor expansion

\[ A(s) \approx A(s_0) + (s - s_0)^T A'(s_0) + (s - s_0)^T \frac{A''(s_0)}{2} (s - s_0) \]

Substitute \( s = \tilde{s} \overset{\text{def}}{=} \theta^T \tilde{x} \), \( s_0 = \mathbb{E}[\tilde{s}] \)

Take expectations to get the quadratic approximation:

\[ R^q(\theta, x) = \frac{1}{2} \mathbb{E}[(\tilde{s} - s)^T \nabla^2 A(s)(\tilde{s} - s)] \]

\[ = \frac{1}{2} \text{tr}(\nabla^2 A(s) \text{Cov}(\tilde{s})) \]
Take the Taylor expansion

\[ A(s) \approx A(s_0) + (s - s_0)^T A'(s_0) + (s - s_0)^T \frac{A''(s_0)}{2} (s - s_0) \]

Substitute \( s = \tilde{s} \overset{\text{def}}{=} \theta \cdot \tilde{f}(y, x) \), \( s_0 = \mathbb{E}[\tilde{s}] \)

Take expectations to get the quadratic approximation:

\[ R^q(\theta, x) = \frac{1}{2} \mathbb{E}[(\tilde{s} - s)^T \nabla^2 A(s)(\tilde{s} - s)] \]

\[ = \frac{1}{2} \text{tr}(\nabla^2 A(s) \text{Cov}(\tilde{s})) \]
Use the independence structure

- Depends on the underlying graphical model
- We assume we can do exact inference via message passing (e.g. clique tree)

- E.g. Linear-chain CRF:

\[
f(y, x) = \sum_{t=1}^{T} g_t(y_{t-1}, y_t, x)
\]

\[
A(s) = \log \sum_{y \in Y} \exp \left\{ \sum_{t=1}^{T} s_{y_{t-1}, y_t, t} \right\}
\]
Local noising

- Global noising:
  \[
  \tilde{s} \overset{\text{def}}{=} \theta \cdot \tilde{f}(y, x)
  \]

- Local noising:
  \[
  \tilde{s} \overset{\text{def}}{=} \theta \cdot \sum_{t=1}^{T} \tilde{g}(y_{t-1}, y_t, x)
  \]

- Can try to justify in restrospect
The regularizer

- The regularizer is:

\[ R^q(\theta, x) = \frac{1}{2} \sum_{a,b,t} \mu_{a,b,t}(1 - \mu_{a,b,t}) \text{Var}[\tilde{s}_{a,b,t}] \]

- For marginals:

\[ \mu_{a,b,t} = p_\theta(y_{t-1} = a, y_t = \tilde{b} \mid x) \]

- And derivatives:

\[ \nabla \mu_{a,b,t} = \mathbb{E}_{p_\theta(y \mid x, y_{t-1}=a, y_t=b)} [f(y, x)] - \mathbb{E}_{p_\theta(y \mid x)} [f(y, x)] \]
Efficient computation

• For every $a, b, t$ we need

\[ \nabla \mu_{a, b, t} = \mathbb{E}_{p_\theta(y|x, y_{t-1}=a, y_t=b)}[f(y, x)] - \mathbb{E}_{p_\theta(y|x)}[f(y, x)] \]

• Naïve computation is $O(K^4T^2)$
  • Can reduce to $O(K^3T^2)$ with independence

• We provide a dynamic program to compute in $O(K^2T)$
Feature group trick (Mengqiu)

\[ \nabla \mu_{a,b,t} = \mathbb{E}_{p_{\theta}(y|x, y_{t-1} = a, y_t = b)}[f(y, x)] - \mathbb{E}_{p_{\theta}(y|x)}[f(y, x)] \]

- Features that always appeared in the same location all have the same conditional expectations
- Gives a 4x speedup, applicable to general CRFs
CRF sequence tagging

- CoNLL 2003 Named Entity Recognition

<table>
<thead>
<tr>
<th>Dataset \ Settings</th>
<th>None</th>
<th>L₂</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL 2003 Dev</td>
<td>89.40</td>
<td>90.73</td>
<td>91.86</td>
</tr>
<tr>
<td>CoNLL 2003 Test</td>
<td>84.67</td>
<td>85.82</td>
<td>87.42</td>
</tr>
</tbody>
</table>
CRF sequence tagging

- Dropout helps more on precision than recall

<table>
<thead>
<tr>
<th>Tag</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_{\beta=1}$</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>87.96%</td>
<td>86.13%</td>
<td>87.03</td>
<td>86.26%</td>
<td>87.74%</td>
<td>86.99</td>
</tr>
<tr>
<td>MISC</td>
<td>77.53%</td>
<td>79.30%</td>
<td>78.41</td>
<td>81.52%</td>
<td>77.34%</td>
<td>79.37</td>
</tr>
<tr>
<td>ORG</td>
<td>81.30%</td>
<td>80.49%</td>
<td>80.89</td>
<td>88.29%</td>
<td>81.89%</td>
<td>84.97</td>
</tr>
<tr>
<td>PER</td>
<td>90.30%</td>
<td>93.33%</td>
<td>91.79</td>
<td>92.15%</td>
<td>92.68%</td>
<td>92.41</td>
</tr>
<tr>
<td>Overall</td>
<td>85.57%</td>
<td>86.08%</td>
<td>85.82</td>
<td>88.40%</td>
<td>86.45%</td>
<td>87.42</td>
</tr>
</tbody>
</table>

(e) CoNLL test set with $L_2$ regularization  
(f) CoNLL test set with dropout regularization
SANCL POS Tagging

- Test set difference statistically significant for newsgroups and reviews

<table>
<thead>
<tr>
<th>$F_{\beta=1}$</th>
<th>None</th>
<th>$L_2$</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>newsgroups</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>91.34</td>
<td>91.34</td>
<td>91.47</td>
</tr>
<tr>
<td>Test</td>
<td>91.44</td>
<td>91.44</td>
<td>91.81</td>
</tr>
<tr>
<td><strong>reviews</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>91.97</td>
<td>91.95</td>
<td>92.10</td>
</tr>
<tr>
<td>Test</td>
<td>90.70</td>
<td>90.67</td>
<td>91.07</td>
</tr>
<tr>
<td><strong>answers</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>90.78</td>
<td>90.79</td>
<td>90.70</td>
</tr>
<tr>
<td>Test</td>
<td>91.00</td>
<td>90.99</td>
<td>91.09</td>
</tr>
</tbody>
</table>
Summary

- Part 0: Some backgrounds
- Part 1: Dropout as adaptive regularization
  - with applications to semi-supervised learning
  - joint work with Stefan Wager and Percy
- Part 2: Applications to structured prediction using CRFs
  - when the log-partition function cannot be easily computed
  - joint work with Mengqiu, Chris, Percy and Stefan Wager
References

• Our arXiv paper [Wager et al., 2013] has more details, including the relation to AdaGrad

• Our EMNLP paper [Wang et al., 2013] extends this framework to structured prediction

• Our ICML paper [Wang and Manning, 2013] applies a related technique to neural networks and provides some negative examples
Dropout vs. L\textsubscript{2}

- Can be much better than all settings of L\textsubscript{2}
- Part of the gain comes from normalization

<table>
<thead>
<tr>
<th>Dataset</th>
<th>L\textsubscript{2} only</th>
<th>L\textsubscript{2}+Gaussian dropout</th>
<th>L\textsubscript{2}+Quadratic dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL</td>
<td>78.03</td>
<td>80.12</td>
<td>80.90</td>
</tr>
<tr>
<td>20news</td>
<td>81.44</td>
<td>82.19</td>
<td>83.37</td>
</tr>
<tr>
<td>RCV1</td>
<td>95.76</td>
<td>95.90</td>
<td>96.03</td>
</tr>
<tr>
<td>R21578</td>
<td>92.24</td>
<td>92.24</td>
<td>92.24</td>
</tr>
<tr>
<td>TDT2</td>
<td>97.74</td>
<td>97.91</td>
<td>98.00</td>
</tr>
</tbody>
</table>

Table 2: Classification performance and transductive learning results on some standard datasets.

None: use no regularization, Drop: quadratic approximation to the dropout noise (7), +Test: also use the test set to estimate the noising regularizer (10).

5.1.1 Semi-supervised Learning with Feature Noising

In the transductive setting, we used test data (without labels) to learn a better regularizer. As an alternative, we could also use unlabeled data in place of the test data to accomplish a similar goal; this leads to a semi-supervised setting.

To test the semi-supervised idea, we use the same datasets as above. We split each dataset evenly into 3 thirds that we use as a training set, a test set and an unlabeled dataset. Results are given in Table 3.

In most cases, our semi-supervised accuracies are lower than the transductive accuracies given in Table 2; this is normal in our setup, because we used less labeled data to train the semi-supervised classifier than the transductive one.

4.5.2 The Second-Order Approximation

The results reported above all rely on the approximate dropout regularizer (7) that is based on a second-order Taylor expansion. To test the validity of this approximation we compare it to the Gaussian method developed by Wang and Manning (2013) on a two-class classification task.

We use the 20-newsgroups alt.atheism vs soc.religion.christian classification task; results are shown in Figure 2.
Example: linear least squares

- The loss function is  \( f(\theta \cdot x) = 1/2(\theta \cdot x - y)^2 \)
- Let \( X = \theta \cdot \tilde{x} \) where \( \tilde{x}_j = 2z_jx_j, z_j = \text{Bernoulli}(0.5) \)

\[
\mathbb{E}[f(X)] = f(\mathbb{E}[X]) + \frac{f''(\mathbb{E}[X])}{2} \text{Var}[X] 
\]
\[
= 1/2(\theta \cdot x - y)^2 + 1/2 \sum_j x_j^2 \theta_j^2 
\]

- The total regularizer is

\[
R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i x_{ji}^2 
\]

- This is just L2 applied after data normalization
## Quantitative results on IMDB

<table>
<thead>
<tr>
<th>Method \ Settings</th>
<th>Supervised</th>
<th>Semi-sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB - unigrams with SFE [Su et al., 2011]</td>
<td>83.62</td>
<td>84.13</td>
</tr>
<tr>
<td>MNB – bigrams</td>
<td>86.63</td>
<td>86.98</td>
</tr>
<tr>
<td>Vectors for sentiment analysis [Maas et al., 2011]</td>
<td>88.33</td>
<td>88.89</td>
</tr>
<tr>
<td>NBSVM – bigrams [Wang and Manning, 2012]</td>
<td>91.22</td>
<td>-</td>
</tr>
<tr>
<td>This work: dropout + unigrams</td>
<td>87.78</td>
<td>89.52</td>
</tr>
<tr>
<td>This work: dropout + bigrams</td>
<td>91.31</td>
<td>91.98</td>
</tr>
</tbody>
</table>