Feature Noising

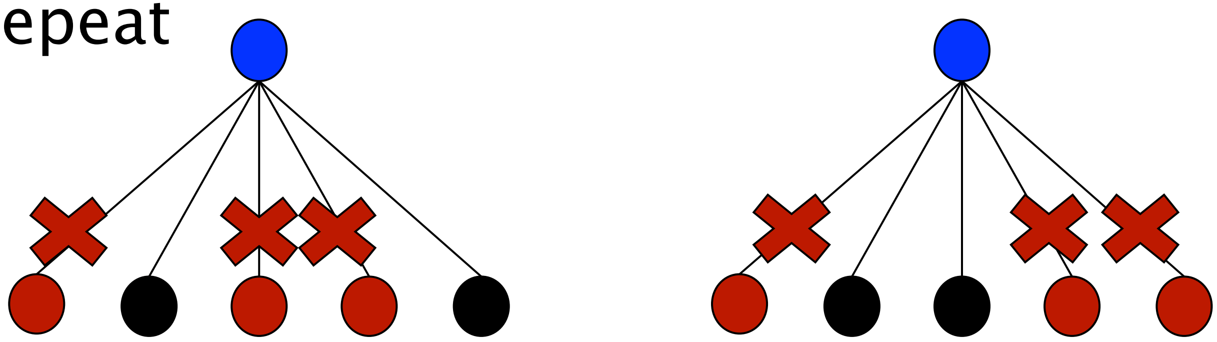
Sida Wang, joint work with
Part 1: Stefan Wager, Percy Liang
Part 2: Mengqiu Wang, Chris Manning, Percy Liang, Stefan Wager

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:

$$\text{Regularizer(Unlabeled data)}$$

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 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:

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

$$= \text{Loss(data)+Regularizer(data)}$$

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: $$\tilde{x}_j = \begin{cases} 2x_j & \text{with p=0.5} \\ 0 & \text{otherwise} \end{cases} \qquad \mathbb{E}[\tilde{x}] = x$$

- Dropout objective:

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

Loss(data)+Regularizer(data)

Dropout for Log-linear Models

- We can rewrite the dropout log-likelihood

$$\begin{aligned}
\mathbb{E}[\log p(y|\tilde{x};\theta)] &= \mathbb{E}[\tilde{x}^T\theta_y] && -\mathbb{E}[A(\tilde{x}^T\theta)] \\
\log p(y|x;\theta) &= x^T\theta_y && -A(x^T\theta) \\
\underbrace{\mathbb{E}[\log p(y|\tilde{x};\theta)]}_{\text{-Loss(Dropout(data))}} &= \underbrace{\log p(y|x;\theta)}_{\text{-Loss(data)}} && -(\underbrace{\mathbb{E}[A(\tilde{x}^T\theta)] - A(x^T\theta)}_{\text{Regularizer(data)}})
\end{aligned}$$

- 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 $s = \tilde{s} \stackrel{\text{def}}{=} \theta^T \tilde{x}$, $s_0 = \mathbb{E}[\tilde{s}]$

Take expectations to get the quadratic approximation:

$$\begin{aligned} R^{\mathrm{q}}(\theta, x) &= \frac{1}{2}\mathbb{E}[(\tilde{s} - \mathbf{s})^T \nabla^2 A(\mathbf{s})(\tilde{s} - \mathbf{s})] \\ &= \frac{1}{2}\mathrm{tr}(\nabla^2 A(\mathbf{s})\mathrm{Cov}(\tilde{s})) \end{aligned}$$

Example: logistic regression

- The quadratic approximation

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

Example: logistic regression

- The quadratic approximation

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

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

$$p = p(y|x;\theta) = (1 + \exp(-yx^T\theta))^{-1}$$

Example: logistic regression

- The quadratic approximation

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

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

$$p = p(y|x;\theta) = (1 + \exp(-yx^T\theta))^{-1}$$

- $\mathrm{Var}[\tilde{x}^T\theta] = \sum_j \theta_j^2 x_j^2$ 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

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

$$R_*(\theta, \mathcal{D}, \mathcal{D}_{\text{unlabeled}}) \stackrel{\text{def}}{=} \frac{n}{n + \alpha m}\Bigg(\sum_{i=1}^{n} R(\theta, x_i) + \alpha \sum_{i=1}^{m} R(\theta, u_i)\Bigg).$$

for some tunable α.

Semi-supervised intuition

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

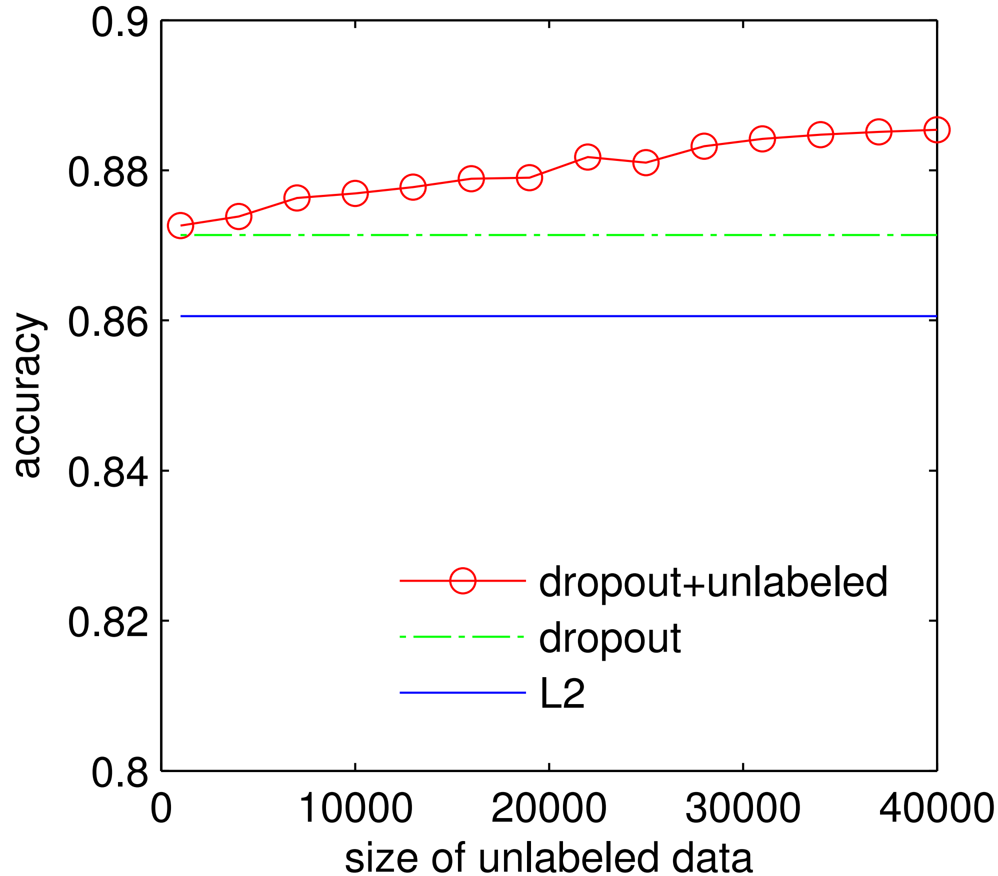
- Like other semi-supervised methods:
  - transductive SVMs [Joachims, 1999]
  - entropy regularization [Grandvalet and Bengio, 2005]
  - EM: guess a label [Nigam et al., 2000]
  - want to make confident predictions on the unlabeled data
- Get a better estimate of the Fisher information

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

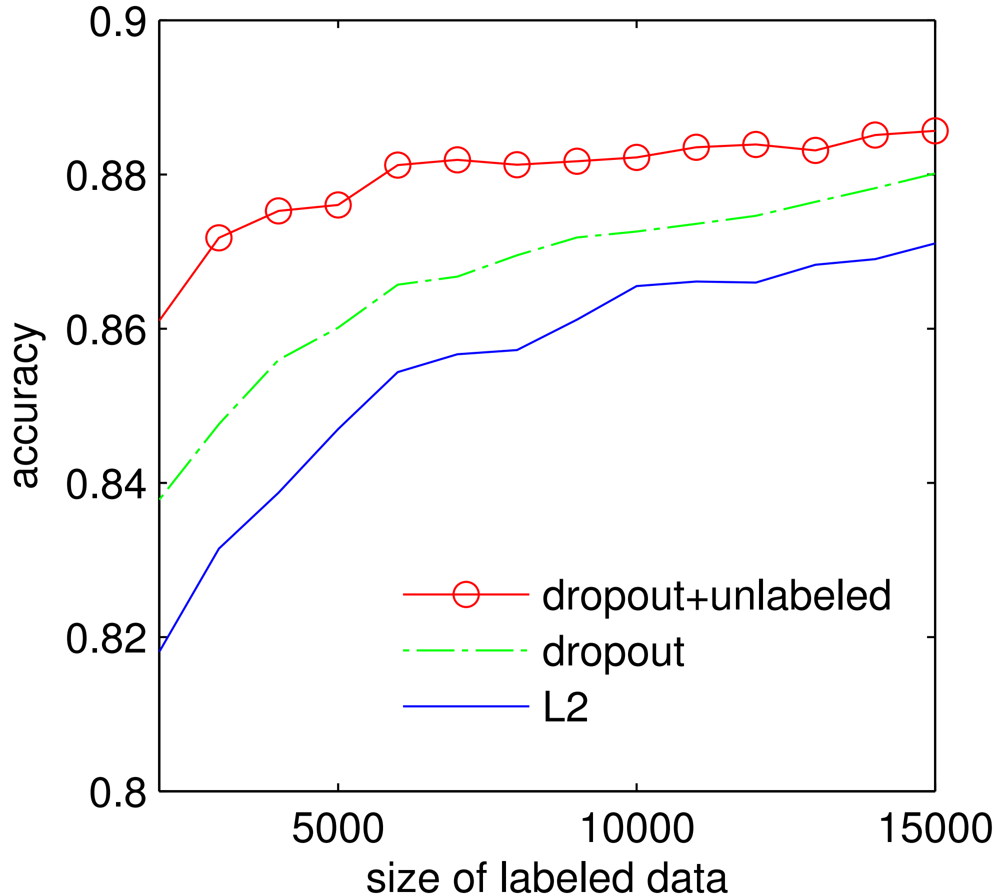
Experiments: semi-supervised

- Add more unlabeled data (10k labeled) improves performance



Experiments: semi-supervised

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



Quantitative results on IMDB

Method \ Settings	Supervised	Semi-sup.
MNB - unigrams with SFE [Su et al., 2011]	83.62	84.13
Vectors for sentiment analysis [Maas et al., 2011]	88.33	88.89
This work: dropout + unigrams	87.78	89.52
This work: dropout + bigrams	91.31	**91.98**

Experiments: other datasets

Dataset \ Settings	L_2	Drop	+Unlbl
Subjectivity [Peng and Lee, 2004]	88.96	90.85	91.48
Rotten Tomatoes [Peng and Lee, 2005]	73.49	75.18	76.56
20-newsgroups	82.19	83.37	84.71
CoNLL-2003	80.12	80.90	81.66

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 $\mathbf{s} = (s_1, \ldots, s_{|\mathcal{Y}|})$ $s_{\mathbf{y}} = f(\mathbf{y}, x) \cdot \theta$

- The likelihood is:

$$p(\mathbf{y} \mid x; \theta) = \exp\{s_{\mathbf{y}} - A(\mathbf{s})\}$$

$$A(\mathbf{s}) = \log \sum_{\mathbf{y}} \exp\{s_{\mathbf{y}}\}$$

- $|\mathcal{Y}|$ 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 $s = \tilde{s} \stackrel{\text{def}}{=} \theta^T \tilde{x}$, $s_0 = \mathbb{E}[\tilde{s}]$

Take expectations to get the quadratic approximation:

$$\begin{aligned} R^{\mathrm{q}}(\theta, x) &= \frac{1}{2}\mathbb{E}[(\tilde{s} - \mathbf{s})^T \nabla^2 A(\mathbf{s})(\tilde{s} - \mathbf{s})] \\ &= \frac{1}{2}\mathrm{tr}(\nabla^2 A(\mathbf{s})\mathrm{Cov}(\tilde{s})) \end{aligned}$$

The structured prediction 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 $s = \tilde{\mathbf{s}} \stackrel{\text{def}}{=} \theta \cdot \tilde{f}(\mathbf{y}, x)$, $s_0 = \mathbb{E}[\tilde{s}]$

Take expectations to get the quadratic approximation:

$$\begin{aligned} R^{\mathrm{q}}(\theta, x) &= \frac{1}{2}\mathbb{E}[(\tilde{s} - \mathbf{s})^T \nabla^2 A(\mathbf{s})(\tilde{s} - \mathbf{s})] \\ &= \frac{1}{2}\mathrm{tr}(\nabla^2 A(\mathbf{s})\mathrm{Cov}(\tilde{s})) \end{aligned}$$

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:

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

$$A(\mathbf{s}) = \log \sum_{y \in \mathcal{Y}} \exp \left\{ \sum_{t=1}^{T} s_{y_{t-1}, y_t, t} \right\}$$

Local Noising

- Global noising:

$$s = \tilde{\mathbf{s}} \stackrel{\text{def}}{=} \theta \cdot \tilde{f}(\mathbf{y}, x)$$

- Local noising:

$$s = \tilde{\mathbf{s}} \stackrel{\text{def}}{=} \theta \cdot \sum_{t=1}^{T} \tilde{g}(y_{t-1}, y_t, x)$$

- Can try to justify in restrospect

The regularizer

- The regularizer is:

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

- For marginals:

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

- And derivatives:

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

Efficient computation

- For every a,b,t we need

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

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

- We provide a dynamic program to compute in $O(KT^2)$, like normal forward backwards, except need to do this for every feature

Feature group trick (Mengqiu)

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

- 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
  - Stanford[ORG] is[O] near[O] Palo[LOC] Alto[LOC]

Dataset \ Settings	None	L_2	Drop
CoNLL 2003 Dev	89.40	90.73	91.86
CoNLL 2003 Test	84.67	85.82	87.42

CRF sequence tagging

- Dropout helps more on precision than recall

Tag	Precision	Recall	$F_{\beta=1}$
LOC	87.96%	86.13%	87.03
MISC	77.53%	79.30%	78.41
ORG	81.30%	80.49%	80.89
PER	90.30%	93.33%	91.79
Overall	85.57%	86.08%	85.82

(e) CoNLL test set with L_2 regularization

Tag	Precision	Recall	$F_{\beta=1}$
LOC	86.26%	87.74%	86.99
MISC	81.52%	77.34%	79.37
ORG	88.29%	81.89%	84.97
PER	92.15%	92.68%	92.41
Overall	88.40%	86.45%	87.42

(f) CoNLL test set with dropout regularization

SANCL POS Tagging

- Test set difference statistically significant for newsgroups and reviews

$F_{\beta=1}$	None	L_2	Drop
	`newsgroups`		
Dev	91.34	91.34	91.47
Test	91.44	91.44	91.81
	`reviews`		
Dev	91.97	91.95	92.10
Test	90.70	90.67	91.07
	`answers`		
Dev	90.78	90.79	90.70
Test	91.00	90.99	91.09

Summary

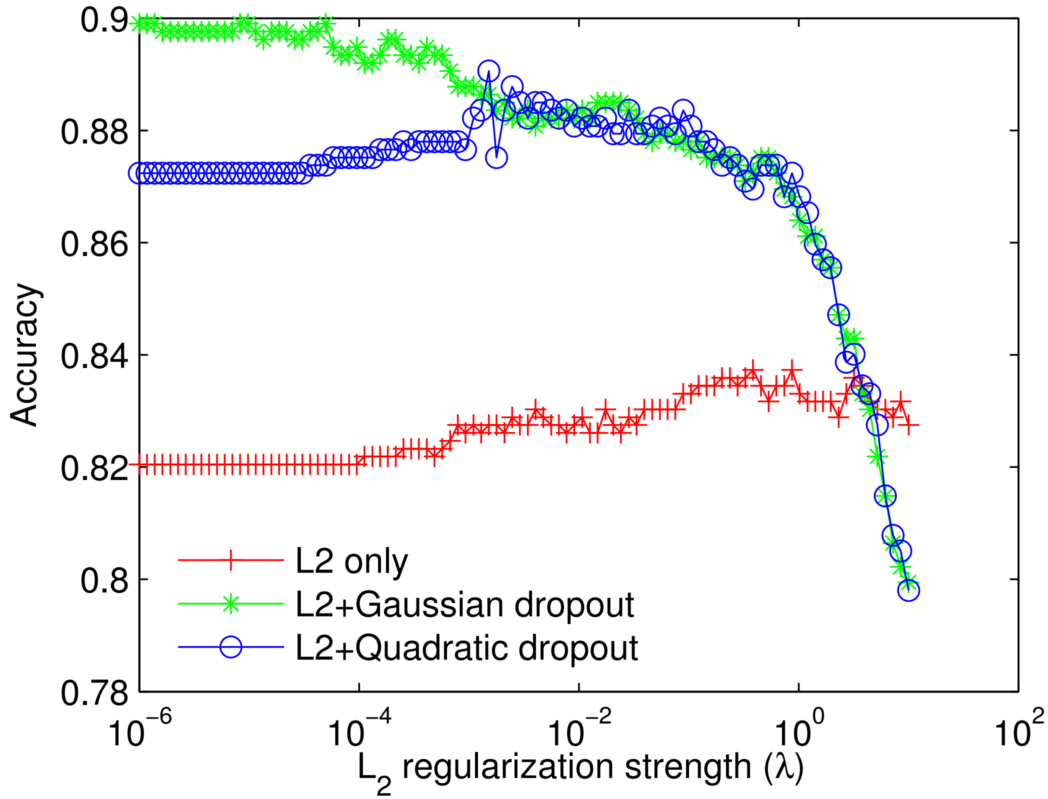
- Part 0: Some backgrounds
- Part 1: Dropout as adaptive regularization
  - with applications to semi-supervised learning
  - joint work with Stefan Wager and Percy
- Part 2: Applications to structured prediction using CRFs
  - when the log-partition function cannot be easily computed
  - joint work 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_2

- Can be much better than all settings of L_2
- Part of the gain comes from normalization



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 $\tilde{x}_j = 2 z_j x_j$, $z_j = \text{Bernoulli}(0.5)$

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

- The total regularizer is

$$R^q(\theta) = \frac{1}{2} \sum_j \theta_j^2 \sum_i x_j^{(i)2}$$

- This is just L2 applied after data normalization

Quantitative results on IMDB

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