Feature Noising for Log-linear Structured Prediction

Sida I. Wang*, Mengqiu Wang*, Stefan Wager†,
Percy Liang, Christopher D. Manning
Department of Computer Science, †Department of Statistics
Stanford University, Stanford, CA 94305, USA
{sidaw, mengqiu, pliang, manning}@cs.stanford.edu
swager@stanford.edu

Abstract

NLP models have many and sparse features, and regularization is key for balancing model overfitting versus underfitting. A recently repopularized form of regularization is to generate fake training data by repeatedly adding noise to real data. We reinterpret this noising as an explicit regularizer, and approximate it with a second-order formula that can be used during training without actually generating fake data. We show how to apply this method to structured prediction using multinomial logistic regression and linear-chain CRFs. We tackle the key challenge of developing a dynamic program to compute the gradient of the regularizer efficiently. The regularizer is a sum over inputs, so we can estimate it more accurately via a semi-supervised or transductive extension. Applied to text classification and NER, our method provides a $>1\%$ absolute performance gain over use of standard L_2 regularization.

1 Introduction

NLP models often have millions of mainly sparsely attested features. As a result, balancing overfitting versus underfitting through good weight regularization remains a key issue for achieving optimal performance. Traditionally, L_2 or L_1 regularization is employed, but these simple types of regularization penalize all features in a uniform way without taking into account the properties of the actual model.

An alternative approach to regularization is to generate fake training data by adding random noise to the input features of the original training data. Intuitively, this can be thought of as simulating missing features, whether due to typos or use of a previously unseen synonym. The effectiveness of this technique is well-known in machine learning (Abu-Mostafa, 1990; Burges and Schölkopf, 1997; Simard et al., 2000; Rifai et al., 2011a; van der Maaten et al., 2013), but working directly with many corrupted copies of a dataset can be computationally prohibitive. Fortunately, feature noising ideas often lead to tractable deterministic objectives that can be optimized directly. Sometimes, training with corrupted features reduces to a special form of regularization (Matsuoka, 1992; Bishop, 1995; Rifai et al., 2011b; Wager et al., 2013). For example, Bishop (1995) showed that training with features that have been corrupted with additive Gaussian noise is equivalent to a form of L_2 regularization in the low noise limit. In other cases it is possible to develop a new objective function by marginalizing over the artificial noise (Wang and Manning, 2013; van der Maaten et al., 2013).

The central contribution of this paper is to show how to efficiently simulate training with artificially noised features in the context of log-linear structured prediction, without actually having to generate noised data. We focus on dropout noise (Hinton et al., 2012), a recently popularized form of artificial feature noise where a random subset of features is omitted independently for each training example. Dropout and its variants have been shown to outperform L_2 regularization on various tasks (Hinton et al., 2012; Wang and Manning, 2013; Wan et al., 2013). Dropout is is similar in spirit to feature bagging in the deliberate removal of features, but performs the removal in a preset way rather than randomly (Bryll et al., 2003; Sutton et al., 2005; Smith et al., 2005).

*Both authors contributed equally to the paper

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1170–1179,
Seattle, Washington, USA, 18-21 October 2013.

Our approach is based on a second-order approximation to feature noising developed among others by Bishop (1995) and Wager et al. (2013), which allows us to convert dropout noise into a form of adaptive regularization. This method is suitable for structured prediction in log-linear models where second derivatives are computable. In particular, it can be used for multiclass classification with maximum entropy models (a.k.a., softmax or multinomial logistic regression) and for the sequence models that are ubiquitous in NLP, via linear chain Conditional Random Fields (CRFs).

For linear chain CRFs, we additionally show how we can use a noising scheme that takes advantage of the clique structure so that the resulting noising regularizer can be computed in terms of the pairwise marginals. A simple forward-backward-type dynamic program can then be used to compute the gradient tractably. For ease of implementation and scalability to semi-supervised learning, we also outline an even faster approximation to the regularizer. The general approach also works in other clique structures in addition to the linear chain when the clique marginals can be computed efficiently.

Finally, we extend feature noising for structured prediction to a transductive or semi-supervised setting. The regularizer induced by feature noising is label-independent for log-linear models, and so we can use unlabeled data to learn a better regularizer. NLP sequence labeling tasks are especially well suited to a semi-supervised approach, as input features are numerous but sparse, and labeled data is expensive to obtain but unlabeled data is abundant (Li and McCallum, 2005; Jiao et al., 2006).

Wager et al. (2013) showed that semi-supervised dropout training for logistic regression captures a similar intuition to techniques such as entropy regularization (Grandvalet and Bengio, 2005) and transductive SVMs (Joachims, 1999), which encourage confident predictions on the unlabeled data. Semi-supervised dropout has the advantage of only using the predicted label probabilities on the unlabeled data to modulate an L_2 regularizer, rather than requiring more heavy-handed modeling of the unlabeled data as in entropy regularization or expectation regularization (Mann and McCallum, 2007).

In experimental results, we show that simulated feature noising gives more than a 1% absolute boost in performance over L_2 regularization, on both text classification and an NER sequence labeling task.

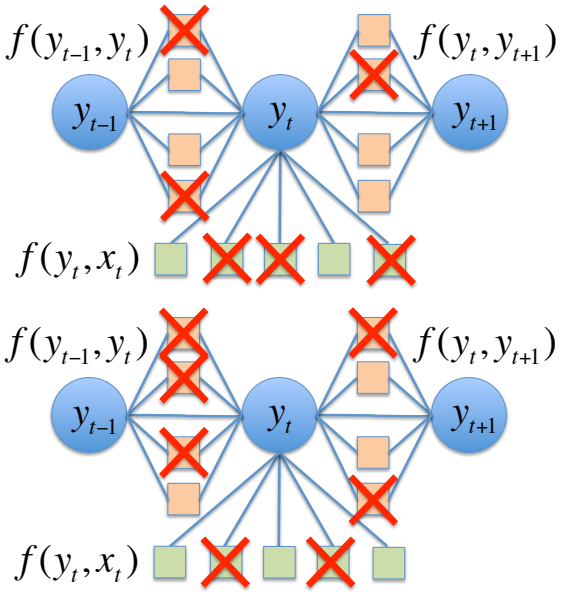

Figure 1: An illustration of dropout feature noising in linear-chain CRFs with only transition features and node features. The green squares are node features $f(y_t, x_t)$, and the orange squares are edge features $f(y_{t-1}, y_t)$. Conceptually, given a training example, we sample some features to ignore (generate fake data) and make a parameter update. Our goal is to train with a roughly equivalent objective, without actually sampling.

2 Feature Noising Log-linear Models

Consider the standard structured prediction problem of mapping some input $x \in \mathcal{X}$ (e.g., a sentence) to an output $\mathbf{y} \in \mathcal{Y}$ (e.g., a tag sequence). Let $f(\mathbf{y}, x) \in \mathbb{R}^d$ be the feature vector, $\theta \in \mathbb{R}^d$ be the weight vector, and $\mathbf{s} = (s_1, \ldots, s_{|\mathcal{Y}|})$ be a vector of scores for each output, with $s_{\mathbf{y}} = f(\mathbf{y}, x) \cdot \theta$. Now define a log-linear model:

$$p(\mathbf{y} \mid x; \theta) = \exp\{s_{\mathbf{y}} - A(\mathbf{s})\}, \qquad (1)$$

where $A(\mathbf{s}) = \log \sum_{\mathbf{y}} \exp\{s_{\mathbf{y}}\}$ is the log-partition function. Given an example $(x, \mathbf{y})$, parameter estimation corresponds to choosing θ to maximize $p(\mathbf{y} \mid x; \theta)$.

The key idea behind feature noising is to artificially corrupt the feature vector $f(\mathbf{y}, x)$ randomly

into some $\tilde{f}(\mathbf{y}, x)$ and then maximize the average log-likelihood of $\mathbf{y}$ given these corrupted features—the motivation is to choose predictors θ that are robust to noise (missing words for example). Let $\tilde{\mathbf{s}}$, $\tilde{p}(\mathbf{y} \mid x; \theta)$ be the *randomly* perturbed versions corresponding to $\tilde{f}(\mathbf{y}, x)$. We will also assume the feature noising preserves the mean: $\mathbb{E}[\tilde{f}(\mathbf{y}, x)] = f(\mathbf{y}, x)$, so that $\mathbb{E}[\tilde{\mathbf{s}}] = \mathbf{s}$. This can always be done by scaling the noised features as described in the list of noising schemes.

It is useful to view feature noising as a form of regularization. Since feature noising preserves the mean, the feature noising objective can be written as the sum of the original log-likelihood plus the difference in log-normalization constants:

$$\mathbb{E}[\log \tilde{p}(\mathbf{y} \mid x; \theta)] = \mathbb{E}[\tilde{\mathbf{s}}_{\mathbf{y}} - A(\tilde{\mathbf{s}})] \quad (2)$$

$$= \log p(\mathbf{y} \mid x; \theta) - R(\theta, x), \quad (3)$$

$$R(\theta, x) \stackrel{\text{def}}{=} \mathbb{E}[A(\tilde{\mathbf{s}})] - A(\mathbf{s}). \quad (4)$$

Since $A(\cdot)$ is convex, $R(\theta, x)$ is always positive by Jensen's inequality and can therefore be interpreted as a regularizer. Note that $R(\theta, x)$ is in general non-convex.

Computing the regularizer (4) requires summing over all possible noised feature vectors, which can imply exponential effort in the number of features. This is intractable even for flat classification. Following Bishop (1995) and Wager et al. (2013), we approximate $R(\theta, x)$ using a second-order Taylor expansion, which will allow us to work with only means and covariances of the noised features. We take a quadratic approximation of the log-partition function $A(\cdot)$ of the noised score vector $\tilde{\mathbf{s}}$ around the the unnoised score vector $\mathbf{s}$:

$$A(\tilde{\mathbf{s}}) \approxeq A(\mathbf{s}) + \nabla A(\mathbf{s})^\top (\tilde{\mathbf{s}} - \mathbf{s}) \quad (5)$$

$$+ \frac{1}{2}(\tilde{\mathbf{s}} - \mathbf{s})^\top \nabla^2 A(\mathbf{s})(\tilde{\mathbf{s}} - \mathbf{s}).$$

Plugging (5) into (4), we obtain a new regularizer $R^{\mathrm{q}}(\theta, x)$, which we will use as an approximation to $R(\theta, x)$:

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2}\mathbb{E}[(\tilde{\mathbf{s}} - \mathbf{s})^\top \nabla^2 A(\mathbf{s})(\tilde{\mathbf{s}} - \mathbf{s})] \quad (6)$$

$$= \frac{1}{2} \operatorname{tr}(\nabla^2 A(\mathbf{s}) \operatorname{Cov}(\tilde{\mathbf{s}})). \quad (7)$$

This expression still has two sources of potential intractability, a sum over an exponential number of noised score vectors $\tilde{\mathbf{s}}$ and a sum over the $|\mathcal{Y}|$ components of $\tilde{\mathbf{s}}$.

Multiclass classification If we assume that the components of $\tilde{\mathbf{s}}$ are independent, then $\operatorname{Cov}(\tilde{\mathbf{s}}) \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ is diagonal, and we have

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2} \sum_{\mathbf{y} \in \mathcal{Y}} \mu_{\mathbf{y}}(1 - \mu_{\mathbf{y}}) \operatorname{Var}[\tilde{s}_{\mathbf{y}}], \quad (8)$$

where the mean $\mu_{\mathbf{y}} \stackrel{\text{def}}{=} p_\theta(\mathbf{y} \mid x)$ is the model probability, the variance $\mu_{\mathbf{y}}(1 - \mu_{\mathbf{y}})$ measures model uncertainty, and

$$\operatorname{Var}[\tilde{s}_{\mathbf{y}}] = \theta^\top \operatorname{Cov}[\tilde{f}(\mathbf{y}, x)]\theta \quad (9)$$

measures the uncertainty caused by feature noising.[1] The regularizer $R^{\mathrm{q}}(\theta, x)$ involves the product of two variance terms, the first is non-convex in θ and the second is quadratic in θ. Note that to reduce the regularization, we will favor models that (i) predict confidently and (ii) have stable scores in the presence of feature noise.

For multiclass classification, we can explicitly sum over each $\mathbf{y} \in \mathcal{Y}$ to compute the regularizer, but this will be intractable for structured prediction.

To specialize to multiclass classification for the moment, let us assume that we have a separate weight vector for each output $\mathbf{y}$ applied to the same feature vector $g(x)$; that is, the score $s_{\mathbf{y}} = \theta_{\mathbf{y}} \cdot g(x)$. Further, assume that the components of the noised feature vector $\tilde{g}(x)$ are independent. Then we can simplify (9) to the following:

$$\operatorname{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \operatorname{Var}[g_j(x)]\theta_{yj}^2. \quad (10)$$

Noising schemes We now give some examples of possible noise schemes for generating $\tilde{f}(\mathbf{y}, x)$ given the original features $f(\mathbf{y}, x)$. This distribution affects the regularization through the variance term $\operatorname{Var}[\tilde{s}_{\mathbf{y}}]$.

- *Additive Gaussian*: $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_{d \times d})$.

[1] Here, we are using the fact that first and second derivatives of the log-partition function are the mean and variance.

In this case, the contribution to the regularizer from noising is $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \sigma^2 \theta_{yj}^2$.

- *Dropout*:

  $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) \odot z$, where $\odot$ takes the elementwise product of two vectors. Here, z is a vector with independent components which has $z_i = 0$ with probability δ, $z_i = \frac{1}{1-\delta}$ with probability $1 - \delta$. In this case, $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \frac{g_j(x)^2 \delta}{1-\delta} \theta_{yj}^2$.

- *Multiplicative Gaussian*:

  $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) \odot (1 + \varepsilon)$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_{d \times d})$. Here, $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j g_j(x)^2 \sigma^2 \theta_{yj}^2$. Note that under our second-order approximation $R^{\mathrm{q}}(\theta, x)$, the multiplicative Gaussian and dropout schemes are equivalent, but they differ under the original regularizer $R(\theta, x)$.

2.1 Semi-supervised learning

A key observation (Wager et al., 2013) is that the noising regularizer R (8), while involving a sum over examples, is independent of the output $\mathbf{y}$. This suggests estimating R using unlabeled data. Specifically, if we have n labeled examples $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ and m unlabeled examples $\mathcal{D}_{\text{unlabeled}} = \{u_1, u_2, \ldots, u_n\}$, then we can define a regularizer that is a linear combination the regularizer estimated on both datasets, with α tuning the tradeoff between the two:

$$R_*(\theta, \mathcal{D}, \mathcal{D}_{\text{unlabeled}}) \tag{11}$$
$$\stackrel{\text{def}}{=} \frac{n}{n + \alpha m} \Big(\sum_{i=1}^{n} R(\theta, x_i) + \alpha \sum_{i=1}^{m} R(\theta, u_i) \Big).$$

3 Feature Noising in Linear-Chain CRFs

So far, we have developed a regularizer that works for all log-linear models, but—in its current form—is only practical for multiclass classification. We now exploit the decomposable structure in CRFs to define a new noising scheme which does not require us to explicitly sum over all possible outputs $\mathbf{y} \in \mathcal{Y}$. The key idea will be to noise each local feature vector (which implicitly affects many $\mathbf{y}$) rather than noise each $\mathbf{y}$ independently.

Assume that the output $\mathbf{y} = (y_1, \ldots, y_T)$ is a sequence of T tags. In linear chain CRFs, the feature vector f decomposes into a sum of local feature vectors g_t:

$$f(\mathbf{y}, x) = \sum_{t=1}^{T} g_t(y_{t-1}, y_t, x), \tag{12}$$

where $g_t(a, b, x)$ is defined on a pair of consecutive tags a, b for positions $t - 1$ and t.

Rather than working with a score $s_{\mathbf{y}}$ for each $\mathbf{y} \in \mathcal{Y}$, we define a collection of *local scores* $\mathbf{s} = \{s_{a,b,t}\}$, for each tag pair (a, b) and position $t = 1, \ldots, T$. We consider noising schemes which independently set $\tilde{g}_t(a, b, x)$ for each a, b, t. Let $\tilde{\mathbf{s}} = \{\tilde{s}_{a,b,t}\}$ be the corresponding collection of noised scores.

We can write the log-partition function of these local scores as follows:

$$A(\mathbf{s}) = \log \sum_{y \in \mathcal{Y}} \exp \left\{ \sum_{t=1}^{T} s_{y_{t-1}, y_t, t} \right\}. \tag{13}$$

The first derivative yields the edge marginals under the model, $\mu_{a,b,t} = p_\theta(y_{t-1} = a, y_t = b \mid x)$, and the diagonal elements of the Hessian $\nabla^2 A(\mathbf{s})$ yield the marginal variances.

Now, following (7) and (8), we obtain the following regularizer:

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2} \sum_{a,b,t} \mu_{a,b,t}(1 - \mu_{a,b,t}) \, \mathrm{Var}[\tilde{s}_{a,b,t}], \tag{14}$$

where $\mu_{a,b,t}(1 - \mu_{a,b,t})$ measures model uncertainty about edge marginals, and $\mathrm{Var}[\tilde{s}_{a,b,t}]$ is simply the uncertainty due to noising. Again, minimizing the regularizer means making confident predictions and having stable scores under feature noise.

Computing partial derivatives So far, we have defined the regularizer $R^{\mathrm{q}}(\theta, x)$ based on feature noising. In order to minimize $R^{\mathrm{q}}(\theta, x)$, we need to take its derivative.

First, note that $\log \mu_{a,b,t}$ is the difference of a restricted log-partition function and the log-partition function. So again by properties of its first derivative, we have:

$$\nabla \log \mu_{a,b,t} = \mathbb{E}_{p_\theta(\mathbf{y} \mid x, y_{t-1}=a, y_t=b)}[f(\mathbf{y}, x)] - \mathbb{E}_{p_\theta(\mathbf{y} \mid x)}[f(\mathbf{y}, x)]. \tag{15}$$

Using the fact that $\nabla \mu_{a,b,t} = \mu_{a,b,t} \nabla \log \mu_{a,b,t}$ and the fact that $\mathrm{Var}[\tilde{s}_{a,b,t}]$ is a quadratic function in θ, we can simply apply the product rule to derive the final gradient $\nabla R^{\mathrm{q}}(\theta, x)$.

3.1 A Dynamic Program for the Conditional Expectation

A naive computation of the gradient $\nabla R^{\mathrm{q}}(\theta, x)$ requires a full forward-backward pass to compute $\mathbb{E}_{p_\theta(\mathbf{y} | y_{t-1}=a, y_t=b, x)}[f(\mathbf{y}, x)]$ for each tag pair (a, b) and position t, resulting in a $\mathcal{O}(K^4 T^2)$ time algorithm.

In this section, we reduce the running time to $\mathcal{O}(K^2 T)$ using a more intricate dynamic program. By the Markov property of the CRF, $\mathbf{y}_{1:t-2}$ only depends on (y_{t-1}, y_t) through y_{t-1} and $\mathbf{y}_{t+1:T}$ only depends on (y_{t-1}, y_t) through y_t.

First, it will be convenient to define the partial sum of the local feature vector from positions i to j as follows:

$$G_{i:j} = \sum_{t=i}^{j} g_t(y_{t-1}, y_t, x). \tag{16}$$

Consider the task of computing the feature expectation $\mathbb{E}_{p_\theta(\mathbf{y} | y_{t-1}=a, y_t=b)}[f(\mathbf{y}, x)]$ for a fixed (a, b, t). We can expand this quantity into

$$\sum_{\mathbf{y}: y_{t-1}=a, y_t=b} p_\theta(\mathbf{y}_{-(t-1:t)} \mid y_{t-1} = a, y_t = b) G_{1:T}.$$

Conditioning on y_{t-1}, y_t decomposes the sum into three pieces:

$$\sum_{\mathbf{y}: y_{t-1}=a, y_t=b} [g_t(y_{t-1} = a, y_t = b, x) + F_t^a + B_t^b],$$

where

$$F_t^a = \sum_{\mathbf{y}_{1:t-2}} p_\theta(\mathbf{y}_{1:t-2} \mid y_{t-1} = a) G_{1:t-1}, \tag{17}$$

$$B_t^b = \sum_{\mathbf{y}_{t+1:T}} p_\theta(\mathbf{y}_{t+1:T} \mid y_t = b) G_{t+1:T}, \tag{18}$$

are the expected feature vectors summed over the prefix and suffix of the tag sequence, respectively. Note that F_t^a and B_t^b are analogous to the forward and backward messages of standard CRF inference, with the exception that they are vectors rather than scalars.

We can compute these messages recursively in the standard way. The forward recurrence is

$$F_t^a = \sum_b p_\theta(y_{t-2} = b \mid y_{t-1} = a) \left[g_t(y_{t-2} = b, y_{t-1} = a, x) + F_{t-1}^b \right],$$

and a similar recurrence holds for the backward messages B_t^b.

Running the resulting dynamic program takes $\mathcal{O}(K^2 T q)$ time and requires $\mathcal{O}(KTq)$ storage, where K is the number of tags, T is the sequence length and q is the number of active features. Note that this is the same order of dependence as normal CRF training, but there is an additional dependence on the number of active features q, which makes training slower.

4 Fast Gradient Computations

In this section, we provide two ways to further improve the efficiency of the gradient calculation based on ignoring long-range interactions and based on exploiting feature sparsity.

4.1 Exploiting Feature Sparsity and Co-occurrence

In each forward-backward pass over a training example, we need to compute the conditional expectations for all features active in that example. Naively applying the dynamic program in Section 3 is $\mathcal{O}(K^2 T)$ for each active feature. The total complexity has to factor in the number of active features, q. Although q only scales linearly with sentence length, in practice this number could get large pretty quickly. For example, in the NER tagging experiments (*cf.* Section 5), the average number of active features per token is about 20, which means $q \simeq 20T$; this term quickly dominates the computational costs. Fortunately, in sequence tagging and other NLP tasks, the majority of features are sparse and they often co-occur. That is, some of the active features would fire and only fire at the same locations in a given sequence. This happens when a particular token triggers multiple rare features.

We observe that all indicator features that only fired once at position t have the same conditional expectations (and model expectations). As a result, we can collapse such a group of features into a single

feature as a preprocessing step to avoid computing identical expectations for each of the features. Doing so on the same NER tagging experiments cuts down q/T from 20 to less than 5, and gives us a 4 times speed up at no loss of accuracy. The exact same trick is applicable to the general CRF gradient computation as well and gives similar speedup.

4.2 Short-range interactions

It is also possible to speed up the method by resorting to approximate gradients. In our case, the dynamic program from Section 3 together with the trick described above ran in a manageable amount of time. The techniques developed here, however, could prove to be useful on larger tasks.

Let us rewrite the quantity we want to compute slightly differently (again, for all a, b, t):

$$\sum_{i=1}^{T} \mathbb{E}_{p_\theta(\mathbf{y}|x, y_{t-1}=a, y_t=b)}[g_i(y_{i-1}, y_i, x)]. \quad (19)$$

The intuition is that conditioned on y_{t-1}, y_t, the terms $g_i(y_{i-1}, y_i, x)$ where i is far from t will be close to $\mathbb{E}_{p_\theta(\mathbf{y}|x)}[g_i(y_{i-1}, y_i, x)]$.

This motivates replacing the former with the latter whenever $|i-k| \geq r$ where r is some window size. This approximation results in an expression which only has to consider the sum of the local feature vectors from $i-r$ to $i+r$, which is captured by $G_{i-r:i+r}$:

$$\begin{aligned} &\mathbb{E}_{p_\theta(\mathbf{y}|y_{t-1}=a, y_t=b, x)}[f(\mathbf{y}, x)] - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[f(\mathbf{y}, x)] \\ &\quad \approx \mathbb{E}_{p_\theta(\mathbf{y}|y_{t-1}=a, y_t=b, x)}[G_{t-r:t+r}] \\ &\qquad - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[G_{t-r:t+r}]. \end{aligned} \quad (20)$$

We can further approximate this last expression by letting $r = 0$, obtaining:

$$g_t(a, b, x) - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[g_t(y_{t-1}, y_t, x)]. \quad (21)$$

The second expectation can be computed from the edge marginals.

The accuracy of this approximation hinges on the lack of long range dependencies. Equation (21) shows the case of $r = 0$; this takes almost no additional effort to compute. However, for some of our experiments, we observed a 20% difference with the real derivative. For $r > 0$, the computational savings are more limited, but the bounded-window method is easier to implement.

Dataset	q	d	K	N_{train}	N_{test}
CoNLL	20	437906	5	204567	46666
SANCL	5	679959	12	761738	82405
20news	81	62061	20	15935	3993
RCV1_4	76	29992	4	9625/2	9625/2
R21578	47	18933	65	5946	2347
TDT2	130	36771	30	9394/2	9394/2

Table 1: Description of datasets. q: average number of non-zero features per example, d: total number of features, K: number of classes to predict, N_{train}: number of training examples, N_{test}: number of test examples.

5 Experiments

We show experimental results on the CoNLL-2003 Named Entity Recognition (NER) task, the SANCL Part-of-speech (POS) tagging task, and several document classification tasks.[2] The datasets used are described in Table 1. We used standard splits whenever available; otherwise we split the data at random into a test set and a train set of equal sizes (RCV1_4, TDT2). CoNLL has a development set of size 51578, which we used to tune regularization parameters. The SANCL test set is divided into 3 genres, namely `answers`, `newsgroups`, and `reviews`, each of which has a corresponding development set.[3]

5.1 Multiclass Classification

We begin by testing our regularizer in the simple case of classification where $\mathcal{Y} = \{1, 2, \ldots, K\}$ for K classes. We examine the performance of the noising regularizer in both the fully supervised setting as well as the transductive learning setting.

In the transductive learning setting, the learner is allowed to inspect the test features at train time (without the labels). We used the method described in Section 2.1 for transductive dropout.

[2]The document classification data are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets` and `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

[3]The SANCL dataset has two additional genres—emails and weblogs—that we did not use, as we did not have access to development sets for these genres.

Dataset	K	None	L_2	Drop	+Test
CoNLL	5	78.03	80.12	80.90	81.66
20news	20	81.44	82.19	83.37	84.71
RCV1$_4$	4	95.76	95.90	96.03	96.11
R21578	65	92.24	92.24	92.24	92.58
TDT2	30	97.74	97.91	98.00	98.12

Table 2: Classification performance and transductive learning results on some standard datasets. None: use no regularization, Drop: quadratic approximation to the dropout noise (8), +Test: also use the test set to estimate the noising regularizer (11).

Dataset	K	L_2	Drop	+Unlabeled
CoNLL	5	91.46	91.81	92.02
20news	20	76.55	79.07	80.47
RCV1$_4$	4	94.76	94.79	95.16
R21578	65	90.67	91.24	90.30
TDT2	30	97.34	97.54	97.89

Table 3: Semisupervised learning results on some standard datasets. A third (33%) of the full dataset was used for training, a third for testing, and the rest as unlabeled.

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]

5.1.2 The Second-Order Approximation

The results reported above all rely on the approximate dropout regularizer (8) 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. There are 1427 examples with 22178 features, split evenly and randomly into a training set and a test set.

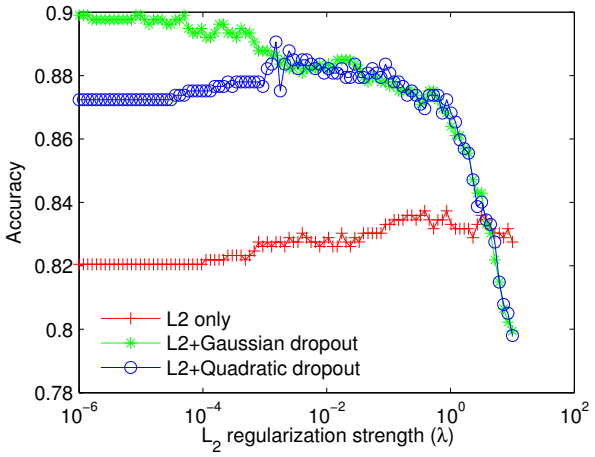

Figure 2: Effect of λ in $\lambda\|\theta\|_2^2$ on the testset performance. Plotted is the test set accuracy with logistic regression as a function of λ for the L_2 regularizer, Gaussian dropout (Wang and Manning, 2013) + additional L_2, and quadratic dropout (8) + L_2 described in this paper. The default noising regularizer is quite good, and additional L_2 does not help. Notice that no choice of λ in L_2 can help us combat overfitting as effectively as (8) without underfitting.

Over a broad range of λ values, we find that dropout plus L_2 regularization performs far better than using just L_2 regularization for any value of λ. We see that Gaussian dropout appears to perform slightly better than the quadratic approximation discussed in this paper. However, our quadratic approximation extends easily to the multiclass case and to structured prediction in general, while Gaussian dropout does not. Thus, it appears that our approximation presents a reasonable trade-off between

[4]The CoNNL results look somewhat surprising, as the semi-supervised results are better than the transductive ones. The reason for this is that the original CoNLL test set came from a different distributions than the training set, and this made the task more difficult. Meanwhile, in our semi-supervised experiment, the test and train sets are drawn from the same distribution and so our semi-supervised task is actually easier than the original one.

computational efficiency and prediction accuracy.

5.2 CRF Experiments

We evaluate the quadratic dropout regularizer in linear-chain CRFs on two sequence tagging tasks: the CoNLL 2003 NER shared task (Tjong Kim Sang and De Meulder, 2003) and the SANCL 2012 POS tagging task (Petrov and McDonald, 2012) .

The standard CoNLL-2003 English shared task benchmark dataset (Tjong Kim Sang and De Meulder, 2003) is a collection of documents from Reuters newswire articles, annotated with four entity types: *Person*, *Location*, *Organization*, and *Miscellaneous*. We predicted the label sequence $\mathcal{Y} = \{\text{LOC, MISC, ORG, PER, O}\}^T$ without considering the BIO tags.

For training the CRF model, we used a comprehensive set of features from Finkel et al. (2005) that gives state-of-the-art results on this task. A total number of 437906 features were generated on the CoNLL-2003 training dataset. The most important features are:

- The word, word shape, and letter n-grams (up to 6gram) at current position
- The prediction, word, and word shape of the previous and next position
- Previous word shape in conjunction with current word shape
- Disjunctive word set of the previous and next 4 positions
- Capitalization pattern in a 3 word window
- Previous two words in conjunction with the word shape of the previous word
- The current word matched against a list of name titles (e.g., Mr., Mrs.)

The $F_{\beta=1}$ results are summarized in Table 4. We obtain a 1.6% and 1.1% absolute gain on the test and dev set, respectively. Detailed results are broken down by precision and recall for each tag and are shown in Table 6. These improvements are significant at the 0.1% level according to the paired bootstrap resampling method of 2000 iterations (Efron and Tibshirani, 1993).

For the SANCL (Petrov and McDonald, 2012) POS tagging task, we used the same CRF framework with a much simpler set of features

- word unigrams: w_{-1}, w_0, w_1
- word bigram: (w_{-1}, w_0) and (w_0, w_1)

$F_{\beta=1}$	None	L_2	Drop
Dev	89.40	90.73	91.86
Test	84.67	85.82	87.42

Table 4: CoNLL summary of results. None: no regularization, Drop: quadratic dropout regularization (14) described in this paper.

$F_{\beta=1}$	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

Table 5: SANCL POS tagging $F_{\beta=1}$ scores for the 3 official evaluation sets.

We obtained a small but consistent improvement using the quadratic dropout regularizer in (14) over the L_2-regularized CRFs baseline.

Although the difference on SANCL is small, the performance differences on the test sets of `reviews` and `newsgroups` are statistically significant at the 0.1% level. This is also interesting because here is a situation where the features are extremely sparse, L_2 regularization gave no improvement, and where regularization overall matters less.

6 Conclusion

We have presented a new regularizer for learning log-linear models such as multiclass logistic regression and conditional random fields. This regularizer is based on a second-order approximation of feature noising schemes, and attempts to favor models that predict confidently and are robust to noise in the data. In order to apply our method to CRFs, we tackle the key challenge of dealing with feature correlations that arise in the structured prediction setting in several ways. In addition, we show that the regularizer can be applied naturally in the semi-supervised setting. Finally, we applied our method to a range of different datasets and demonstrate consistent gains over standard L_2 regularization. Inves-

	Precision	Recall	$F_{\beta=1}$
LOC	91.47%	91.12%	91.29
MISC	88.77%	81.07%	84.75
ORG	85.22%	84.08%	84.65
PER	92.12%	93.97%	93.04
Overall	89.84%	88.97%	89.40

(a) CoNLL dev. set with no regularization

	Precision	Recall	$F_{\beta=1}$
LOC	92.05%	92.84%	92.44
MISC	90.51%	83.52%	86.87
ORG	88.35%	85.23%	86.76
PER	93.12%	94.19%	93.65
Overall	91.36%	90.11%	90.73

(b) CoNLL dev. set with L_2 regularization

	Precision	Recall	$F_{\beta=1}$
LOC	93.59%	92.69%	93.14
MISC	93.99%	81.47%	87.28
ORG	92.48%	84.61%	88.37
PER	94.81%	95.11%	94.96
Overall	93.85%	89.96%	91.86

(c) CoNLL dev. set with dropout regularization

Tag	Precision	Recall	$F_{\beta=1}$
LOC	87.33%	84.47%	85.87
MISC	78.93%	77.12%	78.02
ORG	78.70%	79.49%	79.09
PER	88.82%	93.11%	90.92
Overall	84.28%	85.06%	84.67

(d) CoNLL test set with no regularization

	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

	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

Table 6: CoNLL NER results broken down by tags and by precision, recall, and $F_{\beta=1}$. Top: development set, bottom: test set performance.

tigating how to better optimize this non-convex regularizer online and convincingly scale it to the semi-supervised setting seem to be promising future directions.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, or the US government. S. Wager is supported by a BC and EJ Eaves SGF Fellowship.

References

Yaser S. Abu-Mostafa. 1990. Learning from hints in neural networks. *Journal of Complexity*, 6(2):192–198.

Chris M. Bishop. 1995. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7(1):108–116.

Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. 2003. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302.

Chris J.C. Burges and Bernhard Schölkopf. 1997. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, pages 375–381.

Brad Efron and Robert Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics*, pages 363–370.

Yves Grandvalet and Yoshua Bengio. 2005. Entropy regularization. In *Semi-Supervised Learning*, United Kingdom. Springer.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 209–216.

Thorsten Joachims. 1999. Transductive inference for

text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209.

Wei Li and Andrew McCallum. 2005. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the 20th national conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 813–818.

Gideon S. Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the International Conference on Machine Learning*.

Kiyotoshi Matsuoka. 1992. Noise injection into inputs in back-propagation learning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(3):436–440.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. 2011a. The manifold tangent classifier. *Advances in Neural Information Processing Systems*, 24:2294–2302.

Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. 2011b. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*.

Patrice Y. Simard, Yann A. Le Cun, John S. Denker, and Bernard Victorri. 2000. Transformation invariance in pattern recognition: Tangent distance and propagation. *International Journal of Imaging Systems and Technology*, 11(3):181–197.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 18–25. Association for Computational Linguistics.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2005. Feature bagging: Preventing weight undertraining in structured discriminative learning. *Center for Intelligent Information Retrieval, U. of Massachusetts*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147.

Laurens van der Maaten, Minmin Chen, Stephen Tyree, and Kilian Q. Weinberger. 2013. Learning with marginalized corrupted features. In *Proceedings of the International Conference on Machine Learning*.

Stefan Wager, Sida Wang, and Percy Liang. 2013. Dropout training as adaptive regularization. *arXiv preprint:1307.1493*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine learning*.

Sida Wang and Christopher D. Manning. 2013. Fast dropout training. In *Proceedings of the International Conference on Machine Learning*.