Exponential Priors for Maximum Entropy Models

Joshua Goodman
One Microsoft Way
Redmond, WA 98052
joshuago@microsoft.com

Abstract

Maximum entropy models are a common modeling technique, but prone to overfitting. We show that using an exponential distribution as a prior leads to bounded absolute discounting by a constant. We show that this prior is better motivated by the data than previous techniques such as a Gaussian prior, and often produces lower error rates. Exponential priors also lead to a simpler learning algorithm and to easier to understand behavior. Furthermore, exponential priors help explain the success of some previous smoothing techniques, and suggest simple variations that work better.

1 Introduction

Conditional Maximum Entropy (maxent) models have been widely used for a variety of tasks, including language modeling (Rosenfeld, 1994), part-of-speech tagging, prepositional phrase attachment, and parsing (Ratnaparkhi, 1998), word selection for machine translation (Berger et al., 1996), and finding sentence boundaries (Reynar and Ratnaparkhi, 1997). They are also sometimes called logistic regression models, maximum likelihood exponential models, log-linear models, and are even equivalent to a form of perceptrons/single layer neural networks. In particular, perceptrons that use the standard sigmoid function, and optimize for log-loss are equivalent to maxent. Multi-layer neural networks that optimize log-loss are closely related, and much of the discussion will apply to them implicitly.

Conditional maxent models have traditionally either been unregularized or regularized by using a Gaussian prior on the parameters. We will show that by using an exponential distribution as the prior, several advantages can be gained. We will show that in at least one case, an exponential prior experimentally better matches the actual distribution of the parameters. We will also show that it can lead to improved accuracy, and a simpler learning algorithm. In addition, the exponential prior inspires an improved version of Good Turing discounting with lower perplexity.

Conditional maxent models are of the form

$$P_\Lambda(y|\overline{x}) = \frac{\exp \sum_{i=1}^{F} \lambda_i f_i(\overline{x}, y)}{\sum_{y'} \exp \sum_i \lambda_i f_i(\overline{x}, y')}$$

where $\overline{x}$ is an input vector, y is an output, the f_i are the so-called indicator functions or feature values that are true if a particular property of $\overline{x}, y$ is true, F is the number of such features, Λ represents the parameter set $\lambda_1 ... \lambda_n$, and λ_i is a weight for the indicator f_i. For instance, if trying to do word sense disambiguation for the word "bank", $\overline{x}$ would be the context around an occurrence of the word; y would be a particular sense, e.g. financial or river; $f_i(\overline{x}, y)$ could be 1 if the context includes the word "money" and y is the financial sense; and λ_i would be a large positive number.

Maxent models have several valuable properties (Della Pietra et al. (1997) give a good overview.) The most important is constraint satisfaction. For a given f_i, we can count how many times f_i was observed in the training data with value y, *observed*$[i] = \sum_j f_i(\overline{x_j}, y_j)$. For a model P_Λ with parameters Λ, we can see how many times the model predicts that f_i would be expected to occur: *expected*$[i] = \sum_{j,y} P_\Lambda(y|\overline{x_j}) f_i(\overline{x_j}, y)$. Maxent models have the property that *expected*$[i]$ = *observed*$[i]$ for all i and y. These equalities are called *constraints.* The next important property is that the likelihood of the training data is maximized (thus, the name maximum likelihood exponential model.) Third, the model is as similar as possible to the uniform distribution (minimizes the Kullback-Leibler divergence), given the constraints, which is why these models are called maximum entropy models.

This last property – similarity to the uniform distribution – is a form of regularization. However, it turns out to be an extremely weak one – it is not uncommon for models, especially those that use all or most possible features, to assign near-zero probabilities (or, if λs may be infi-

nite, even actual zero probabilities), and to exhibit other symptoms of severe overfitting. There have been a number of approaches to this problem, which we will discuss in more detail in Section 3. The most relevant approach, however, is the work of Chen and Rosenfeld (2000), who implemented a Gaussian prior for maxent models. They compared this technique to most of the previously implemented techniques, on a language modeling task, and concluded that it was consistently the best. We thus use it as a baseline for our comparisons, and similar considerations motivate our own technique, an exponential prior.

Chen *et al.* place a Gaussian prior with 0 mean and σ_i^2 variance on the model parameters (the λ_is), and then find a model that maximizes the posterior probability of the data and the model. In particular, maxent models without priors use the parameters Λ that maximize

$$\arg\max_{\Lambda} \prod_{j=1}^{n} P_\Lambda(y_j|\overline{x_j})$$

where $\overline{x_j}, y_j$ are training data instances. With a Gaussian prior we find

$$\arg\max_{\Lambda} \prod_{j=1}^{n} P_\Lambda(y_j|\overline{x_j}) \times \prod_{i=1}^{F} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{\lambda_i^2}{2\sigma_i^2}\right)$$

In this case, a trained model does not satisfy the constraints *expected*$[i] =$ *observed*$[i]$, but, as Chen and Rosenfeld show, instead satisfies constraints

$$\textit{expected}[i] = \textit{observed}[i] - \frac{\lambda_i}{\sigma_i^2} \tag{1}$$

That is, instead of a model that matches the observed count, we get a model matching the observed count minus $\frac{\lambda_i}{\sigma_i^2}$: in language modeling terms, this is "discounting."

We do not believe that all models are generated by the same process, and therefore we do not believe that a single prior will work best for all problem types. In particular, as we will describe in our experimental results section, when looking at one particular set of parameters, we noticed that it was not at all Gaussian, but much more similar to a 0 mean Laplacian, $f(\lambda_i) = \frac{1}{2\beta_i} \exp\left(-\frac{|\lambda_i|}{\beta_i}\right)$, or to an exponential distribution $f(\lambda_i) = \alpha_i \exp(-\alpha_i \lambda_i)$, which is non-zero only for non-negative λ_i. In some cases, learned parameter distributions will not match the prior distribution, but in some cases they will, so it seemed worth exploring one of these alternate forms. Later, when we try to optimize our models, the parameter seach will turn out to be much simpler with an exponential prior, so we focus on that distribution.

With an exponential prior, we maximize

$$\arg\max_{\Lambda \geq \overline{0}} \prod_{j=1}^{n} P_\Lambda(y_j|\overline{x_j}) \times \prod_{i=1}^{F} \alpha_i \exp(-\alpha_i \lambda_i) \tag{2}$$

As we will describe, it is significantly simpler to perform this maximization than the Gaussian maximization. Furthermore, as we will describe, models satisfying Equation 2 will have the property that, for each λ_i, either a) $\lambda_i = 0$ and *expected*$[i] \geq$ *observed*$[i] - \alpha_i$ or b) *expected*$[i] =$ *observed*$[i] - \alpha_i$. In other words, we essentially just discount the observed counts by the constant α_i (which is the reciprocal of the standard deviation), subject to the constraint that λ_i is non-negative. This is much simpler and more intuitive than the constraints with the Gaussian prior (Equation 1), since those constraints change as the values of λ_i change. Furthermore, as we will describe in Section 3, discounting by a constant is a common technique for language model smoothing (Ney et al., 1994; Chen and Goodman, 1999), but one that has not previously been well justified; the exponential prior gives some Bayesian justification.

In Section 5 we will show that on two very different tasks – grammar checking and a collaborative filtering task – the exponential prior yields lower error rates than the Gaussian.

2 Learning algorithms and discounting

In this section we derive a learning algorithm for exponential priors, with provable convergence properties, and show that it leads to a simple discounting formula. Note that the simple learning algorithm is an important contribution: the algorithm for a Gaussian prior is quite a bit more complicated, and previous related work with the Laplacian prior (two-sided exponential) has had a difficult time finding learning algorithms; because the Laplacian does not have a continuous first derivative, and because the exponential prior is bounded at 0, standard gradient descent type algorithms may exhibit poor behavior. Williams (1995) devotes a full ten pages to describing a somewhat heuristic approach for solving this problem, and even this discussion concludes "In summary it is left to the reader to supply the algorithms for determining successive search directions and the initially preferred value of" the step size (page 130).[1] We show that a very simple variation on a standard algorithm, Generalized Iterative Scaling (GIS) (Darroch and Ratcliff, 1972), solves this problem. In particular, as we will show, while GIS uses an update rule of the form

$$\lambda_i := \lambda_i + \frac{1}{f\#} \log \frac{\textit{observed}[i]}{\textit{expected}[i]}$$

[1] Williams' algorithm is for the much more complex case of a multi-layer network, while ours is for the one layer case, but there are no obvious simplifications to his approach for the one layer case.

our modified algorithm uses a rule of the form

$$\lambda_i := \max\left(0, \left(\lambda_i + \frac{1}{f^\#}\log\frac{\textit{observed}[i]-\alpha_i}{\textit{expected}[i]}\right)\right) \quad (3)$$

Note that there are two different styles of model that one can use, especially in the common case that there are two outputs (values for y.) Consider a word sense disambiguation problem, trying to determine whether a word like "bank" means the river or financial sense, with questions like whether or not the word "water" occurs nearby. One could have a single indicator function $f_1(x,y) =$ *1 if water occurs nearby* and values in the range $-\infty < \lambda_1 < \infty$. We call this style "double sided" indicators. Alternatively, one could have two indicator functions, $f_1(x,y) =$ *1 if water occurs nearby and y=river* and $f_2(x,y) =$ *1 if water occurs nearby and y=financial.* In this case, one could allow either $-\infty < \lambda_1, \lambda_2 < \infty$ or $0 \le \lambda_1, \lambda_2 < \infty$. We call the style with two indicators "single sided." With a Gaussian prior, it does not matter which style one uses – one can show that by changing σ^2, exactly the same results will be achieved. With a Laplacian (double sided exponential), one could also use either style. With an exponential prior, only positive values are allowed, so one must use the double sided style, so that one can learn that some indicators push towards one sense, and some push towards the other – that is, rather than having one weight which is positive or negative, we have two weights which are positive or zero, one of which pushes towards one answer, and the other pushing towards the other.

We derive our constraints and learning algorithm by very closely following the derivation of the algorithm by Chen and Rosenfeld. It will be convenient to maximize the log of the expression in 2 rather than the expression itself. This leads to an objective function:

$$\begin{aligned} L(\Lambda) =& \sum_j \log P_\Lambda(y_j|\overline{x_j}) - \sum_{i=1}^F \alpha_i\lambda_i + \log\alpha_i \\ =& \sum_j\sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y_j) \\ & -\sum_j \log\sum_y \exp\left(\sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y)\right) - \sum_{i=1}^F \alpha_i\lambda_i + \log\alpha_i \end{aligned} \quad (4)$$

Note that this objective function is convex (since it is the sum of two convex functions.) Thus, there is a global maximum value for this objective function.

Now, we wish to find the maximum. Normally, we would do this by setting the derivative to 0, but the bound of $\lambda_k \ge 0$ changes things a bit. The maximum can then occur at the discontinuity in the derivative ($\lambda_k = 0$) or when $\lambda_k > 0$. We can explicitly check the value of the objective function at the point $\lambda_k = 0$. When there is a maximum with $\lambda_k > 0$ we know that the partial derivative with respect to λ_k will be 0.

$$\begin{aligned} \frac{\partial}{\partial\lambda_k} & \sum_j \sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y_j) \\ & - \sum_j \log\sum_y \exp\left(\sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y)\right) \\ & - \sum_{i=1}^F \alpha_i\lambda_i + \textit{const}(\Lambda) \\ = & \sum_j f_k(\overline{x_j}, y_j) \\ & - \sum_j \frac{\sum_y f_k(\overline{x_j}, y)\exp(\sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y))}{\sum_y \exp\left(\sum_{i=1}^F \lambda_i f_i(\overline{x_j}, y)\right)} - \alpha_k \\ = & \sum_j f_k(\overline{x_j}, y_j) - \sum_j \sum_y f_k(\overline{x_j}, y) P_\Lambda(y|\overline{x_j}) - \alpha_k \end{aligned}$$

This implies that at the optimum, when $\lambda_k > 0$,

$$\sum_j f_k(\overline{x_j}, y_j) - \sum_j\sum_y f_k(\overline{x_j}, y) P_\Lambda(y|\overline{x_j}) - \alpha_k = 0$$

$$\textit{observed}[k] - \textit{expected}[k] - \alpha_k = 0$$

$$\textit{observed}[k] - \alpha_k = \textit{expected}[k] \quad (5)$$

In other words, we discount the observed count by α_k – the absolute discounting equation. Sometimes it is better for λ_k to be set to 0 – another possible optimal point is when $\lambda_k = 0$ and $\textit{observed}[k] - \alpha_k < \textit{expected}[k]$. One of these two cases must hold at the optimum.

Notice an important property of exponential priors (analogous to a similiar property for Laplace priors (Williams, 1995; Tibshirani, 1994)): they often favor parameters that are exactly 0. This leads to a kind of natural pruning for exponential priors, not found in Gaussian priors, which are only very rarely 0. (Note, however, that one should not increase the α's in the exponential prior to control pruning, as this may lead to oversmoothing. If additional pruning is needed for speed or memory savings, feature selection techniques should be used, such as pruning small or infrequent parameters, instead of a strengthened prior.)

Now, we can derive the update equations. The derivation is exactly the same as Chen and Rosenfeld's (2000) with the minor change of an exponential prior instead of a Gaussian prior (we include it in the appendix.) Let $f^\#(x,y) = \sum_i f_i(x,y)$. Then, in the end, we get an update equation of the form

$$\lambda_k := \max\left(0, \lambda_k + \frac{1}{f^\#}\log\frac{\textit{observed}[k]-\alpha_k}{\textit{expected}[k]}\right)$$

Compare this equation to the corresponding equation with a Gaussian prior (Chen and Rosenfeld., 2000). With a Gaussian prior, one can derive an equation of the form

$$\textit{observed}[k] - \frac{\lambda_k}{\sigma_k^2} = \textit{expected}[k]\exp\left(f^\#\delta_k\right)$$

and then solve for δ_k. There is no closed form solution: it must be solved using numerical methods, such as Newton's method, making this update much more complex and time consuming than the exponential prior.

One can also derive variations on this update. For instance, in the Appendix, we derive an update for Improved Iterative Scaling (Della Pietra et al., 1997) with an exponential prior. Perhaps more importantly, one can also derive updates for Sequential Conditional Generalized Iterative Scaling (SCGIS) (Goodman, 2002), which is several times faster than GIS. The SCGIS update for binary features with an exponential prior is simply

$$\lambda_k := \max\left(0, \lambda_k + \log \frac{\textit{observed}[k] - \alpha_k}{\textit{expected}[k]}\right)$$

One might wonder why we simply don't use conjugate gradient (CG) methods, which have been shown to converge quickly for maxent. There has not been a formal comparison of SCGIS to conjugate gradient methods. In our own pilot studies, SCGIS is sometimes faster and sometimes slower. Also, some versions of CG use heuristics for the step size, and lose convergence guarantees. Finally, SCGIS is simpler to implement than conjugate gradient, and even for those with a conjugate gradient library, because of the parameter constraints (for an exponential prior) or discontinuous derivatives (for a Laplacian) standard conjugate gradient techniques need to be at least somewhat modified.

Good-Turing discounting has been used or suggested for language modeling several times (Rosenfeld, 1994, page 38), (Lau, 1994). In particular, it has been suggested to use an update of the form

$$\lambda_k := \lambda_k + \frac{1}{f\#} \log \frac{\textit{observed}[k]^*}{\textit{expected}[k]}$$

where $\textit{observed}[k]^*$ is the Good-Turing discounted value of $\textit{observed}[k]$. This update has a problem, as noted by its proponents: the constraints are probably now inconsistent – there is no model that can simultaneously satisfy them. We note that a simple variation on this update, inspired by the exponential prior, does not have these problems:

$$\lambda_k := \max 0, \left(\lambda_k + \frac{1}{f\#} \log \frac{\textit{observed}[k]^*}{\textit{expected}[k]}\right)$$

In particular, this can be thought of as picking an $\alpha_{\textit{observed}[k]}$ for each $\textit{observed}[k]$. This does not constitute a Bayesian prior, since the value is picked after the counts are observed, but it does lead to a convex objective function with a global maximum, and the update function will converge towards this maximum. Variations on the constraints of Equation 5 will apply for this modified objective function. Furthermore, in the experimental results section, we will see that on a language modeling task, this modified update function outperforms the traditional update. By using a well motivated approach inspired by exponential priors, we can find a simple variation that has better performance both theoretically and empirically.

3 Previous Work

There has been a fair amount of previous work on regularization of maxent models. Early approaches focused on feature selection (Della Pietra et al., 1997) or, similarly, count cutoffs (Rosenfeld, 1994). By not using all features, there is typically extra probability left-over for unobserved events, which is distributed in a maximum entropy fashion. The problem with this approach is that it ignores useful information – although low count or low discrimination features may cause overfitting, they do contain valuable information. Because of this, more recent approaches (Rosenfeld, 1994, page 38), (Lau, 1994) have tried techniques such as Good-Turing discounts (Good, 1953).

There are a number of other approaches (Khudanpur, 1995; Newman, 1997) based on the fuzzy maxent framework (Della Pietra and Della Pietra, 1993). Chen and Rosenfeld (Chen and Rosenfeld., 2000) give a more complete discussion of those approaches.

Chen and Rosenfeld (2000), following a suggestion of Lafferty, implemented a Gaussian prior for maxent models. They compared this technique to most of the previously discussed techniques, on a language modeling task, and concluded that it was consistently the best technique.

Tibshirani (1994) introduced Laplacian priors for linear models (linear regressions) and showed that an objective function that minimizes the absolute values of the parameters corresponds to a Laplacian prior. He called this the Least Absolute Shrinkage and Selection Operator (LASSO) and showed that it leads to a type of feature selection. Exponential priors are sometimes called single-sided Laplacians, so obviously, the two techniques are very closely related, so closely that we would not want to claim that either was better than the other.

Williams (1995) introduced a Laplacian prior for neural networks. Single layer neural networks with certain loss functions are equivalent to logistic regression/maximum entropy models. Williams' algorithm was for a more general case, and much more complex than the one we describe.

More recently, Figueiredo et al. (2003) in unpublished independent work also examined Laplacian priors for logistic regression, deriving a somewhat more complex algorithm than ours, but one that they extended to partially supervised learning. He has shown that the results are comparable to transductive SVMs.

Perkins and Theiler (2003) used logisitic regression with a Laplacian prior as well. Their learning algorithm was conjugate gradient descent. Normally, conjugate gradient methods are only used on data that has a continuous first derivative, so the code was modified to prune weights that go exactly to zero.

Our main contribution then is not the use of Laplacian

priors with logistic regression, nor even the first good learning algorithm for the model type. Our contributions are performing an explicit comparison to a Gaussian prior and showing improved performance on real data; noticing that the fixed point of the models leads to absolute discounting; showing that iterative-scaling style algorithms including GIS, IIS, and SCGIS can be trivially modified to use this prior; and explicitly showing that in at least one case, parameters are actually consistent with this prior.

4 Kneser-Ney smoothing

In this section, we help explain the excellent performance of Kneser-Ney smoothing, the best performing language model smoothing technique. This justification not only answers an important question – why do discounting methods work well – but also gives guidance for extending Kneser-Ney smoothing to problems with fractional counts, where solutions were not previously known.

Chen and Goodman (1999) performed an extensive comparison of different smoothing (regularization) techniques for language modeling. They found that a version of Kneser-Ney smoothing (Kneser and Ney, 1995) consistently was the best performing technique. Unfortunately, while there are partial theoretical justifications for Kneser-Ney smoothing, in terms of preserving marginals, one important part has previously had no justification: Kneser-Ney smoothing *discounts* counts, while most conventional regularization techniques, justified by Dirichlet priors, *add to* counts. Given that discounting was previously unjustified, it is exciting that we have found a way to explain it. In fact, we can show that the Backoff version of Kneser-Ney smoothing is an excellent approximation to a maximum entropy model with an exponential prior. In particular, Kneser-Ney smoothing is derived by assuming absolute discounting, and then finding the distribution that preserves marginals, i.e. making expected = observed - discount. The differences between Backoff Kneser-Ney smoothing and maxent models with exponential priors are twofold. First, the backoff version does not exactly preserve marginals – an approximation is made. Second, Backoff Kneser-Ney always performs discounting, even when this effectively results in lowering the probability, e.g. the equivalent of a negative value for λ. There is also an interpolated version of Kneser-Ney smoothing. The interpolated version of Kneser Ney smoothing works even better. It exactly preserves marginals (except for discounting.) Also, the secondary distribution is combined with the primary distribution; this has several effects, including that it is unlikely to get the equivalent of a large negative λ value.

5 Experimental Results

In this section, we detail our experimental results, showing that exponential priors outperform Gaussian priors on two different data sets, and inspire improvements for a third. For all experiments, except the language model experiments, we used a single variance for both the Gaussian and the exponential prior, rather than one per parameter, with the variance optimized on held out data. For the language modeling experiments, we used three variances, one each for the unigram, bigram, and trigram models, again optimized on held out data.

We were inspired to use an exponential prior by an actual examination of a data set. In particular, we used the grammar-checking data of Banko and Brill (2001). We chose this set because there are commonly used versions both with small amounts of data (which is when we expect the prior to matter) and with large amounts of data (which is required to easily see what the distribution over "correct" parameter values is.) For one experiment, we trained a model using a Gaussian prior, using a large amount of data. We then found those parameters (λ's) that had at least 35 training instances – enough to typically overcome the prior and train the parameter reliably. We then graphed the distribution of these parameters. While it is common to look at the distribution of data, the NLP and machine learning communities rarely examine distributions of model parameters, and yet this seems like a good way to get inspiration for priors to try, using those parameters with enough data to help guess the priors for those with less, or at least to determine the correct form for the prior, if not the exact values.[2] The results are shown in Figure 1, which is a histogram of λ's with a given value. If the distribution were Gaussian, we would expect this to look like an upside-down parabola. If the distribution were Laplacian, we would expect it to appear as a triangle (the bottom formed from the X-axis.) Indeed, it does appear to be roughly triangular, and to the extent that it diverges from this shape, it is convex, while a Gaussian would be concave. We don't believe that the exponential prior is right for every problem – our argument here is that based on both better accuracy (our next experiment) and a better fit to at least some of the parameters, that the exponential prior is better for some models.

We then tried actually using exponential priors with this application, and were able to demonstrate improve-

[2] Of course, those parameters with lots of data might be generated from a different prior than those with little data. This technique is meant as a form of inspiration and evidence, but not of proof. Similarly, all parameters may be generated by some other process, e.g. a mixture of Gaussians. Finally, a prior might be accurate but still behave poorly, because it might interact poorly with other approximations. For instance, it might interact poorly with the fact that we use $\arg\max$ rather than the true Bayesian posterior over models.

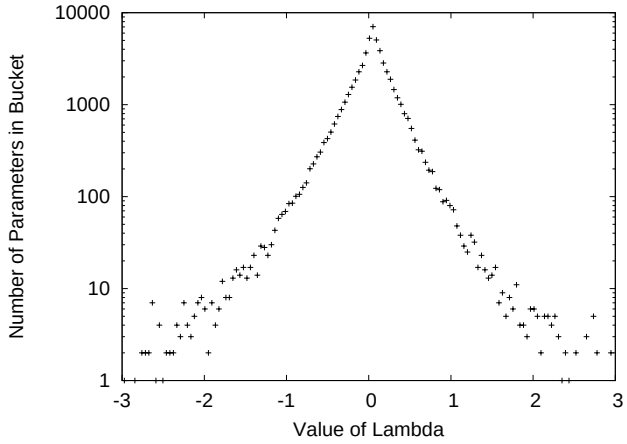

Figure 1: Histogram of λ values

ments in error rate. We used a small data set, 100,000 sentences of training data and ten different confusable word pairs. (Most training sentences did not contain examples of the confusable word pairs of interest, so the actual number of training examples for each word-pair was less than 100,000). We tried different priors for the Gaussian and exponential prior, and found the best single prior on average across all ten pairs. With this best setting, we achieved a 14.51% geometric average error rate with the exponential prior, and 15.45% with the Gaussian. To avoid any form of cheating, we then tried 10 different word pairs (the same as those used by Banko and Brill (2001)) with this best parameter setting. The results were 18.07% and 19.47% for the exponential and Gaussian priors respectively. (The overall higher rate is due to the test set words being slightly more difficult.) We also tried experiments with 1 million and 10 million words, but there were not consistent differences; improved smoothing mostly matters with small amounts of training data.

We also tried experiments with a collaborative-filtering style task, television show recommendation, based on Nielsen data. The dataset used, and the definition of a collaborative filtering (CF) score is the same as was used by Kadie *et al.* (2002), although our random train/test split is not the same, so the results are not strictly comparable. We first ran experiments with different priors on a held-out section of the training data, and then using the single best value for the prior (the same one across all features), we ran on the test data. With a Gaussian prior, the CF score was 42.11, while with an exponential prior, it was 45.86, a large improvement.

Finally, we ran experiments with language modeling, with mixed success. We used 1,000,000 words of training data (a small model, but one where smoothing matters) from the WSJ corpus and a trigram model with a cluster-based speedup (Goodman, 2001). We evaluated on test data using the standard language modeling measure, perplexity, where lower scores are better. We tried five experiments: using Katz smoothing (a widely used version of Good-Turing smoothing) (perplexity 238.0); using Good Turing discounting to smooth maxent (perplexity 224.8); using our variation on Good-Turing, inspired by exponential priors, where λ's are bounded at 0 (perplexity 204.5); using an exponential prior (perplexity 190.8); using a Gaussian prior (perplexity 183.7); and using interpolated modified Kneser-Ney smoothing (perplexity 180.2). On the one hand, an exponential prior is worse than a Gaussian prior in this case, and modified interpolated Kneser-Ney smoothing is still the best known smoothing technique (Chen and Goodman, 1999), within noise of a Gaussian prior. On the other hand, searching for parameters is extremely time consuming, and Good-Turing is one of the few parameter-free smoothing methods. Of the three Good-Turing smoothing methods, the one inspired by exponentials priors was the best.

Note that perplexity is $2^{entropy}$ and in general, we have found that exponential priors work slightly worse on entropy measures than the Gaussian prior, even when they are better on accuracy. This may be due to the fact that an exponential prior "throws away" some information, whenever the λ would be negative. (In a pilot experiment with a variation that does not throw away information, the entropies are closer to the Gaussian.)

6 Conclusion

We have shown that an exponential prior for maxent models leads to a simple update formula that is easy to implement, and to models that are easy to understand: observations are discounted, subject to the constraint that $\lambda \geq 0$. We have also shown that in at least one case, this prior better matches the underlying model, and that for two applications, it leads to improved accuracy. The prior also inspired an improved version of Good-Turing smoothing with lower perplexity. Finally, an exponential prior explains why models that discount by a constant can be Bayesian, giving an alternative to Dirichlet priors which add a constant. This helps justify Kneser-Ney smoothing, the best performing smoothing technique in language modeling. In the future, we would like to use our technique of examining the distribution of model parameters to see if other problems exhibit other priors besides Gaussian and Laplacian/exponential, and if performance on those problems can be improved through this observation.

Acknowledgments

Thanks to John Platt, who suggested looking at Laplacian priors, and to Chris Meek for helpful discussions, and Jeff Bilmes for reading an earlier version of this pa-

per. Finally, thanks to Stan Chen and Roni Rosenfeld: our derivation for exponential priors closely follows the text of their derivation for Gaussian priors.

References

M. Banko and E. Brill. 2001. Mitigating the paucity of data problem. In *HLT*.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, October.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Trans. on Speech and Audio Processing*, 8(2):37–50, January.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480.

Stephen Della Pietra and Vincent Della Pietra. 1993. Statistical modeling by maximum entropy. Unpublished Manuscript.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April.

Mario A. T. Figueiredo, Balaji Krishnapuram, Lawrence Carin, and Alexander J. Hartemink. 2003. Supervised and semi-supervised sparse Bayesian classification.

I.J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP 2001*.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *ACL '02*.

Carl M. Kadie, Christopher Meek, and David Heckerman. 2002. CFW: A collaborative filtering system using posteriors over weights of evidence. In *Proceedings of UAI*, pages 242–250.

S. Khudanpur. 1995. A method of maximum entropy estimation with relaxed constraints. In *1995 Johns Hopkins University Language Modeling Workshop*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*, volume 1, pages 181–184.

Raymond Lau. 1994. Adaptive statistical language modelling. Master's thesis, MIT.

W. Newman. 1997. Extension to the maximum entropy method. *IEEE Trans. on Information Theory*, IT-23(1):89–93, January.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech, and Language*, 8:1–38.

Simon Perkins and James Theiler. 2003. Online feature selection using grafting. August.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

J. Reynar and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *ANLP*.

Ronald Rosenfeld. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University, April.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. Technical report.

Peter M. Williams. 1995. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7:117–143.

A Derivation of Update Equation

In each iteration, we try to find $\Delta = \{\delta_i\}$ that maximizes the increase in the objective function (subject to the constraint that $\delta_i + \lambda_i \geq 0$). From Equation 4,

$$\begin{aligned} L(\Lambda + \Delta) - L(\Lambda) = & \\ & \sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) \\ & - \sum_j \log \sum_y P_\Lambda(y|\overline{x_j}) \exp\left(\sum_i \delta_i f_i(\overline{x_j}, y)\right) \\ & - \sum_i \alpha_i \delta_i \end{aligned}$$

As with the Gaussian prior, it is not clear how to maximize this function directly, so instead we use an auxiliary function, $B(\Delta)$, with three important properties: first, we can maximize it; second, it bounds this function from below; third, it is larger than zero whenever Λ is not at a local optimum, i.e. does not satisfy the constraints in Equation 5. Using the well-known inequality $\log x \leq x - 1$, which implies $-logx \geq 1 - x$, we get

$$L_X(\Lambda + \Delta) - L_X(\Lambda) \geq$$

$$\sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j)$$
$$+ \sum_j 1 - \sum_y P_\Lambda(y|\overline{x_j}) \exp\left(\sum_i \delta_i f_i(\overline{x_j}, y)\right)$$
$$- \sum_i \alpha_i \delta_i \quad (6)$$

Let $f^\#(x,y) = \sum_i f_i(x,y)$. Modify Equation 6 to:

$$L_X(\Lambda+\Delta) - L_X(\Lambda) \geq \sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) +$$
$$\sum_j 1 - \sum_y P_\Lambda(y|\overline{x_j}) \exp\left(f^\#(\overline{x_j}, y) \sum_i \delta_i \frac{f_i(\overline{x_j}, y)}{f^\#(\overline{x_j}, y)}\right)$$
$$- \sum_i \alpha_i \delta_i \quad (7)$$

Now, recall Jensen's inequality, which states that for a convex function g,

$$\sum_y p(x) g(x) \geq g\left(\sum_x p(x) x\right)$$

Notice that $\frac{f_i(x,y)}{f^\#(x,y)}$ is a probability distribution. Thus, we get

$$L_X(\Lambda+\Delta) - L_X(\Lambda) \geq \sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) +$$
$$\sum_j 1 - \sum_y P_\Lambda(y|\overline{x_j}) \sum_i \frac{f_i(\overline{x_j}, y)}{f^\#(\overline{x_j}, y)} \exp\left(f^\#(\overline{x_j}, y)\delta_i\right)$$
$$- \sum_i \alpha_i \delta_i \quad (8)$$

Now, we would like to find Δ that maximizes Equation 8. Thus, we take partial derivatives and set them to zero, remembering to also check whether a maximum occurs when $\delta_k = 0$.

$$\frac{\partial}{\partial \delta_k}\left(\sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) + \sum_j 1 \right.$$
$$\left. - \sum_y P_\Lambda(y|\overline{x_j}) \sum_i \frac{f_i(\overline{x_j}, y)}{f^\#(\overline{x_j}, y)} \exp\left(f^\#(\overline{x_j}, y)\delta_i\right) - \sum_i \alpha_i \delta_i \right)$$
$$= \sum_j f_k(\overline{x_j}, y_j)$$
$$+ \sum_j - \sum_y P_\Lambda(y|\overline{x_j}) \frac{f_k(\overline{x_j}, y)}{f^\#(\overline{x_j}, y)} \frac{\partial}{\partial \delta_k} \exp\left(f^\#(\overline{x_j}, y)\delta_k\right) - \alpha_k$$
$$= \sum_j f_k(\overline{x_j}, y_j)$$
$$- \sum_j \sum_y P_\Lambda(y|\overline{x_j}) f_k(\overline{x_j}, y) \exp\left(f^\#(\overline{x_j}, y)\delta_k\right) - \alpha_k$$
$$= 0$$

This gives us a version of Improved Iterative Scaling with an exponential Prior. In general, however, we prefer variations of Generalized Iterative Scaling, which may not converge as quickly, but lead to simpler algorithms. In particular, we set $f^\# = \max_{x,y} f^\#(x,y)$. Then, instead of Equation 7, we get

$$L_X(\Lambda+\Delta) - L_X(\Lambda)$$
$$\geq \sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) +$$
$$\sum_j 1 - \sum_y P_\Lambda(y|\overline{x_j}) \exp\left(f^\# \sum_i \delta_i \frac{f_i(\overline{x_j}, y)}{f^\#}\right)$$
$$- \sum_i \alpha_i \delta_i \quad (9)$$

We can follow essentially the same derivation from there. (Technically, we need to add in a slack parameter; the slack parameter can then be given a near-zero variance prior so that its value stays at 0, and thus in practice it can be ignored.) We thus get:

$$\frac{\partial}{\partial \delta_k}\left(\sum_j \sum_i \delta_i f_i(\overline{x_j}, y_j) + \right.$$
$$\left. \sum_j 1 - \sum_y P_\Lambda(y|\overline{x_j}) \sum_i \frac{f_i(\overline{x_j}, y)}{f^\#} \exp\left(f^\# \delta_i - \sum_i \alpha_i \delta_i\right)\right)$$
$$= \sum_j f_k(\overline{x_j}, y_j)$$
$$- \sum_j \sum_y P_\Lambda(y|\overline{x_j}) f_k(\overline{x_j}, y) \exp\left(f^\# \delta_k\right) - \alpha_k$$
$$= \textit{observed}[k] - \textit{expected}[k] \exp\left(f^\# \delta_k\right) - \alpha_k$$
$$= 0 \quad (10)$$

From Equation 10 we get

$$\delta_k = \frac{1}{f^\#} \log \frac{\textit{observed}[k] - \alpha_k}{\textit{expected}[k]}$$

Now, $\delta_k + \lambda_k$ may be less than 0; in this case, an illegal new value for λ_k would result. We know, however, from the monotonicity of all the equations with respect to δ_k that the lowest legal value of δ_k will be the best, and we thus arrive at

$$\delta_k = \max\left(-\lambda_k, \frac{1}{f^\#} \log \frac{\textit{observed}[k] - \alpha_k}{\textit{expected}[k]}\right)$$

or equivalently

$$\lambda_k := \max\left(0, \lambda_k + \frac{1}{f^\#} \log \frac{\textit{observed}[k] - \alpha_k}{\textit{expected}[k]}\right)$$