Carefully Appoximated Bayes Factors for Feature Selection in MaxEnt Models

Hal Daumé III

15 Nov 2004

1 Introduction

Feature selection is essentially a model selection problem. If we take a frequentist maximum likelihood approach, we will, in the limit, select all features (unless, as is typical, we apply some sort of "early stopping" critereon). Additionally, basing the next feature to selected solely on standard measures such as likelihood gain, we fail to account for the *variance* of the estimate of this feature. In this note, I carefully derive an approximation to the *Bayes factor* in the feature/model selection problem for maximum entropy models. See [2] for an introduction to the use of maximum entropy models in the natural language processing domain. The advantages to using a Bayesian criterea for model selection are numerous, but the two strongest are that (a) it enables us to take into account uncertainty in likelihood when adding new features and (b) it allows us to decide when to *stop* adding features.

2 Background

Suppose we have data $\mathcal{D} = \langle x_n, y_n \rangle_{n=1}^N \in (\mathcal{X}^F \times \mathcal{Y})^N$, where $\mathcal{X}$ is typically either $\mathbb{R}$ or 2, and $\mathcal{Y}$ is some small finite set. We have F-many "features" and N-many data points. We model the relationship between $\mathcal{X}$ and $\mathcal{Y}$ through a log-linear model with parameter $\lambda \in \mathbb{R}^{F \times |\mathcal{Y}|}$:

$$p\left(y \mid x\right) = \int_{\Lambda} \mathrm{d}F^{\pi(\lambda)} \frac{\exp \lambda_y{}^\top x}{\sum_{y' \in \mathcal{Y}} \lambda_{y'}{}^\top x} \tag{1}$$

We assume a prior on λ, which is typically a Gaussian with zero mean and constant diagonal covariance. Typically, the problem is separated into "training" and "prediction", where we approximate:

$$p\left(y_{n+1} \mid x_{n+1}, \mathcal{D}\right) = \int_{\Lambda} \mathrm{d}F^{\pi(\lambda \mid \mathcal{D})} \frac{\exp \lambda_y{}^{\top} x}{\sum_{y' \in \mathcal{Y}} \lambda_{y'}{}^{\top} x} \tag{2}$$

$$\approx \frac{\exp \hat{\lambda}_y{}^{\top} x}{\sum_{y' \in \mathcal{Y}} \hat{\lambda}_{y'}{}^{\top} x} \tag{3}$$

$$\text{where } \hat{\lambda} = \max_{\lambda} \mathrm{d}F^{\pi(\lambda \mid \mathcal{D})}$$

The training portion therefore is tasked with finding the optimal $\hat{\lambda}$, and the prediction simply uses this estimated value to predict the new data point. This methodology works well in practice – indeed, we shall use this model as our classifier – but for the purpose of *model selection*, the maximum approximation is a bad idea: we do wish to take into account the variance of λ.

In order to perform model selection, one typically computes and compares the Bayes factor. Given R models, $\mathcal{M}_1, \ldots, \mathcal{M}_R$ with prior distribution $\pi(\cdot)$, we compute $p\left(\mathcal{D} \mid \mathcal{M}_r\right)$ (the *evidence* of $\mathcal{M}_r$) and combine this with the prior, to obtained the Bayes factor, α_r:

$$\alpha_r = \frac{\pi(\mathcal{M}_r) p\left(\mathcal{D} \mid \mathcal{M}_r\right)}{\sum_{r'=1}^{R} \pi(\mathcal{M}_{r'}) p\left(\mathcal{D} \mid \mathcal{M}_{r'}\right)} \tag{4}$$

The evidence, $p\left(\mathcal{D} \mid \mathcal{M}_r\right)$, is computed by integrating over all possible model parameters (in our case, the λs):

$$p\left(\mathcal{D} \mid \mathcal{M}_r\right) = \int_{\Lambda} \mathrm{d}F^{\pi(\lambda \mid \mathcal{M}_r)} p\left(\mathcal{D} \mid \lambda, \mathcal{M}_r\right) \tag{5}$$

The model with the highest Bayes factor, α_r is selected as the "correct" model. See [3, 5] for more discussion of the Bayes factor and issues in Bayesian learning in general.

3 Feature Selection

3.1 Problem Definition

In the feature selection problem, we will assume that $1 \leq G < F$ features have already been selected (as the base case, we can simply select the "bias") and we wish to select one more feature from the set of $F - G$ many available (or, if the model is already optimal, we wish to stop). This can be considered a model selection problem, where we define $\mathcal{M}_0$ to be the current model with G features, and $\mathcal{M}_g$ to be the current model plus feature g for $1 \leq g \leq F - G$. The feature selection task is the equivalent to the model selection task with these models. When $\mathcal{M}_0$ is chosen, it means that we have exhausted the useful features. Intuitively, a feature is good if (a) it is correlated with the class labels and (b) it is not correlated with existing fatures. In our analysis, being correlated with class labels will increase evidence (because the correct labels will become more likely under any λ), while being correlated with existing features will decrease evidence (because the variance of the parameters will increase, and the prior preference for small models will factor in).

Like most Bayesian methods, the procedure outlined above is simple. We compute the model evidence under each proposed model according to Eq (5), combine these with the model priors as in Eq (4) and select the largest on (the denominator in Eq (4) doesn't even need to be computed). Of course, the problem lies in the fact that Eq (5) is analytically intractable.

When faced with analytically intractable integrals (a commonplace occurance in Bayesian learning), one basically has three choices for how to approximate the integral. One can use a variational approximation, which typically amounts to an EM-like iterative optimization procedure; one can use an MCMC technique, which requires the computation of many samples over the desired posterior; or one can use the *Laplace* (or "saddle-point" approximation), in which the posterior is approximated to be Gaussian, and the normalizing constant for the Gaussian (which is available analytically) is used for the evidence. (See [6] for a discussion of all of these options.) In this note, we use the Laplace approximation, since it is deterministic, very efficient (in this case) and gives a reasonable approximation.

3.2 Laplace Approximation

To compute the Laplace approximation, we need to be able to manipulate the posterior $q(\lambda) = \pi(\lambda \mid \mathcal{M}_r) p(\mathcal{D} \mid \lambda, \mathcal{M}_r)$. In particular, we need to be able to compute the λ which maximizes $q(\lambda)$, and we need to be able to compute the matrix of second derivatives of $-\log q$ at this maximum:

$$A_{ij} = -\frac{\partial^2}{\partial \lambda_i \partial \lambda_j} \log q(x) \Bigg|_{\lambda=\lambda^0} \tag{6}$$

where λ^0 maximizes q. In this case, we can approximate the evidence, Eq (5) as:

$$p(\mathcal{D} \mid \mathcal{M}_r) \approx q(\lambda^0) \sqrt{\frac{(2\pi)^K}{\det A}} \tag{7}$$

where K is the dimensionality of A (in our case, $F-G$ for $\mathcal{M}_0$ and $F-G+1$ for $\mathcal{M}_g$) and $\det A$ is the matrix determinant of A.

3.3 Approximating the Laplace Approximation

Following Eq (4), we define β_r as the log, unnormalized Bayes factor:

$$\beta_r = \log \pi(\mathcal{M}_r) + \log p(\mathcal{D} \mid \mathcal{M}_r) \tag{8}$$

This, under the Laplace approximation, becomes:

$$\beta_r = \log \pi(\mathcal{M}_r) + \log \pi(\lambda_0^r) + \log p\left(\mathcal{D} \mid \lambda_0^r\right) + \frac{K}{2}\log(2\pi) - \frac{1}{2}\log\det A^r \tag{9}$$

Where λ_0^r maximizes q and A^r is the corresponding negative log Hessian matrix for model $\mathcal{M}_r$. Unfortunately, when millions or billions of features are used, computing λ_0^r for each possible feature becomes computationally impossible. This, we make our first approximation:

Assumption 1: Suppose λ_0^0 maximizes q for the current G-many features, and we are considering the addition of feature g. Then λ_0^g equals λ_0^0 in all places *except* position g. In other words, when a feature is added, we hold all other weights constant. This is known as the mean-field approximation in statistical physics.

This assumption is reasonable in the case of feature selection because in the feature selection problem, we are explicitly trying to add features that are uncorrelated, which makes it likely that this assumption will give a reasonable approximation (of course, once a feature has been chosen and added, we will re-estimate the entire λ vector).

Under this assumption, we can rewrite the β values, ignoring common terms, as (I have also assume that the prior for feature values factors over the different features, as it does in all cases I know of):

$$\begin{aligned}
\beta_0 &= \log \pi(\mathcal{M}_0) + \log p\left(\mathcal{D} \mid \lambda^0\right) - \frac{1}{2}\log\det A^0 \qquad (10)\\
\beta_g &= \log \pi(\mathcal{M}_g) + \log \pi(\mu_g) + \log p\left(\mathcal{D} \mid \lambda^0, \mu_g\right) + \frac{1}{2}\log(2\pi) - \frac{1}{2}\log\det A^g
\end{aligned}$$

Where $\lambda^0 = \operatorname{argmax}_\lambda \pi(\lambda) p\left(\mathcal{D} \mid \lambda\right)$, $\mu_g = \operatorname{argmax}_\mu \pi(\mu) p\left(\mathcal{D} \mid \lambda^0, \mu\right)$, A^0 is the Hessian for λ^0 and A^g is the Hessian for λ^0 and μ_g.

While this is becoming more tractable, computing the determinant of a $G+1 \times G+1$ matrix for each feature remains prohibitive. We thus approximate the A matrix by a diagonal matrix.

Assumption 2: The off-diagonal elements of the A matrix are 0.

Under this assumption, we need only compute $A_{ii} = a_i$, which can be computed in our case as:

$$a_i = -\sum_{n=1}^{N} \left[\left(\frac{\sum_y x_{ni}^y \exp\left[\lambda_y{}^\top x_n + \mu x_{ni}\right]}{\sum_y \exp\left[\lambda_y{}^\top x_n + \mu x_{ni}\right]} \right)^2 - \frac{\sum_y \left(x_{ni}\right)^2 \exp\left[\lambda_y{}^\top x_n + \mu x_{ni}\right]}{\sum_y \exp\left[\lambda_y{}^\top x_n + \mu x_{ni}\right]} \right] \tag{11}$$

The error introduced by this approximation is likely negligible: as stated before, we are interested in computing features that are not correlated with eachother. If a feature g is not correlated with any existing feature, then this approximation is good (since inductively the "old" A matrix can be approximated by the diagonal, the "new" matrix can also be approximated by the diagonal). Since the log determinant of a diagonal matrix is simply the sum of its elements, under this new assumption, we get:

$$\begin{aligned} \beta_0 &= \log \pi(\mathcal{M}_0) + \log p\left(\mathcal{D} \mid \lambda^0\right) \\ \beta_g &= \log \pi(\mathcal{M}_g) + \log \pi(\mu_g) + \log p\left(\mathcal{D} \mid \lambda^0, \mu_g\right) + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log a_g \end{aligned} \tag{12}$$

Eq (12) has a nice intuitive interpretation. A feature g has a high factor if (a) it makes the data log posterior high and (b) has low variance. This means that a feature both (a) has to be useful for predicting the classes and (b) not be underspecified by the data.

3.4 Model Priors

The only aspect of the feature selector not yet described is the form of the model priors $\pi(\mathcal{M}_r)$. Note that although not written as such, this is actually conditional on the *current* model, $\pi(\mathcal{M}_r \mid \hat{\mathcal{M}})$, where $\hat{\mathcal{M}}$ is the currently selected set of features. The simplest prior available simply places uniform weights on each possible model:

$$\pi(\mathcal{M}_r \mid |\hat{\mathcal{M}}| = G) = \frac{1}{F - G + 1} \tag{13}$$

where $F - G$ is the number of inactive features, and the one is added to account for $\mathcal{M}_0$. Other model priors are possible when prior information is actually available: indeed, one might know *a priori* that some features are more or less good, given the currently selected features, and a carefully crafted prior could account for this knowledge.

3.5 Selecting Multiple Features

For efficiency's sake, it is often desirable to select $1 < S < F - G$ features at a time. One could do so by computing the Bayes factor for all subsets of features, but this is computationally way too expensive. Alternatively, one could simply compute the approximate Bayes factor β_g for all unadded features, and add the top S at once. The problem with this approach is that in problems with many highly correlated features, the top S features are likely to be very similar. From the perspective of data modeling, this is not so bad. We will have redundant features to compute, but the model is able to deal reasonably well with redundant features. The primary concern is that, as more redundant features are added, our approximation to the Hessian matrix becomes worse.

To fix this problem without requiring a quadratic explosing in the number of optimizations required, we recommend the following approach. Suppose the two best features are g' and g, respectively, according to the approximate Bayes factors. We will always at feature g'. Then, based on that we have added feature g', we recompute β_g according to:

$$\log \pi(\mathcal{M}_g) + \log \pi(\mu_g) + \log p\left(\mathcal{D} \mid \lambda^0, \mu_{g'}, \mu_g\right) + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log a_g \tag{14}$$

If g and g' are highly correlated, then the model in which their values have been approximated according to the mean field assumption will likely overfit the data, thus resulting in a *decrease* in the data log likelihood. This will cause the approximate Bayes factor to be lower than that for a feature with little correlation to g'.

3.6 Corrective Feature Selection

As mentioned before, since we make the mean field approximation, to compute the optimal μ for a new feature g, we need only consider data points in which feature g actually appears. Moreover, the data log likelihood will also only change at those points. Thus:

$$\log p\left(\mathcal{D} \mid \lambda^0, \mu_g\right) = \sum_{n=1}^{N}\left[{\lambda^0_{y_n}}^\top x_n + \mu_g x_{ng} - \log \sum_{y' \in \mathcal{Y}} \exp\left({\lambda^0_{y'}}^\top x_n + \mu_g x_{ng}\right)\right] \tag{15}$$

We can make the maximum approximation (also known as the Viterbi approximation) in Eq (15), by replacing the sum over y' with the max over y'. In this case, the $\log$ "eats" the $\exp$, and we obtain:

$$\log p\left(\mathcal{D} \mid \lambda^0, \mu_g\right) \approx \sum_{n=1}^{N}\left[{\lambda^0_{y_n}}^\top x_n + \mu_g x_{ng} - \max_{y'}\left({\lambda^0_{y'}}^\top x_n + \mu_g x_{ng}\right)\right] \tag{16}$$

The computation of the gradient of this approximation is not possible since it contains discontinuities and thus this approximation is useless for the purpose of optimization. However, it should be noted that under this approximation, the gradient of μ_g will be non-zero only if $y \neq y'$. In other words, the gradient will be non-zero only when example n is *misclassified*. We can thus improve the efficiency of our algorithm by considering the addition of features that are present only at *misclassified* points.

3.7 Implementation

The implementation of the technique described in this note is straightforward. Pseudocode for the method is shown in Figure 1. In this, we assume that we are able to find the MAP parameters (the call to OptMAP) for a model with a certain set of features: in practice, I use LM-BFGS to optimize maxent models [7, 1, 4].

```
Algorithm ApproxBFFS
Active ← {Bias}
while true do
  λ^0 ← OptMAP(D, Active)
  β_0 ← log p (D | λ^0, Active)
  Proposed ← ⋃_{n|y_n≠ŷ_n} x_n − Active
  for g ∈ Proposed do
    μ_g ← OptSingle(D, λ^0, g)
  end for
  Current ← ∅
  while |Current| ≤ S and Proposed ≠ ∅ do
    for g ∈ Proposed do
      a_g ← − Σ_{n=1}^{N} [ ( Σ_y x_ni exp[λ_y^⊤ x_n + μ x_ni] / Σ_y exp[λ_y^⊤ x_n + μ x_ni] )^2 − Σ_y (x_ni)^2 exp[λ_y^⊤ x_n + μ x_ni] / Σ_y exp[λ_y^⊤ x_n + μ x_ni] ]
      β_g ← log π(μ_g) + log p (D | λ^0, μ_g, Active ∪ Current ∪ {g})
            + 1/2 log(2π) + 1/2 log a_g
    end for
    if β_0 > β_g for all g ∈ Proposed then
      break
    end if
    ĝ ← argmax_g β_g
    β_0 ← β_g
    Current ← Current ∪ {ĝ}
    Proposed ← Proposed − {ĝ}
  end while
  if Current = ∅ then
    break
  end if
  Active ← Active ∪ Current
end while
return Active
```

Figure 1: Approximate Bayes factor feature selection algorithm.

The algorithm also assumes we can optimize μ for a single parameter under the factorial assumption. Given λ, the derivative of the data log likelihood for feature g is:

$$\frac{\mathrm{d}}{\mathrm{d}\mu} \log p\left(\mathcal{D} \mid \lambda, g\right) = \sum_{n=1}^{N} \left[x_{ng} - \log \sum_{y'} \frac{p_{n,y'} x_{ng} \exp\left(\mu x_{ng}\right)}{\sum_{y'} p_{n,y'} \exp\left(\mu x_{ng}\right)} \right] \tag{17}$$

Here, $p_{n,y'}$ is the current probability of class y' for data point n under λ: $p_{n,y'} \propto \exp \lambda_{y'}{}^{\top} x_n$. Using Eq (17), we can perform simple Newton steps to find μ. We can do a bit better by considering the second derivative, which we must calculate anyway, as they are exactly the a_g values from the Hessian approximation.

4 Conclusion

Using two carefully chosen assumptions (first, the mean-field approximation; second, that features are not correlated) together with the Laplace approximation, we have obtained an efficient feature selection algorithm for maximum entropy models based on Bayes factors. This algorithm is implemented in the MEGAM toolkit: `http://www.isi.edu/~hdaume/megam/`.

References

[1] Brett M. Averick and Jorge J. Moré. Evaluation of large-scale optimization problems on vector and parallel architectures. *SIAM Journal of Optimization*, 4, 1994.

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

[3] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer-Verlag, second edition, September 1985.

[4] Hal Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://www.isi.edu/~hdaume/docs/daume04cg-bfgs.ps`, implementation available at `http://www.isi.edu/~hdaume/megam/`, August 2004.

[5] Robert E. Kass and Adrian E. Raftery. Bayes factors and model uncertainty. *Journal of the American Statistical Association*, 90:773–795, 1995.

[6] David J.C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, first edition, June 2002.

[7] S.G. Nash and J. Nocedal. A numerical study of the limited memory BFGS method and the truncated Newton method for large scale optimization. *SIAM Journal of Optimization*, 1:358–372, 1991.