Semi-Supervised Learning: From Gaussian Fields to Gaussian Processes

Xiaojin Zhu John Lafferty Zoubin Ghahramani

August 18, 2003
CMU-CS-03-175

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We show that the Gaussian random fields and harmonic energy minimizing function framework for semi-supervised learning can be viewed in terms of Gaussian processes, with covariance matrices derived from the graph Laplacian. We derive hyperparameter learning with evidence maximization, and give an empirical study of various ways to parameterize the graph weights.

This research was partially supported by the Advanced Research and Development Activity in Information Technology (ARDA), contract number MDA904-00-C-2106, and by the National Science Foundation (NSF), grant CCR-0122581.

Keywords: I.2.6 [Artificial Intelligence] Learning; I.5.1 [Pattern Recognition] Models–Statistical, I.5.2 [Pattern Recognition] Design Methodology–Classifier design and evaluation; General Terms: Algorithms; Additional Key Words: semi-supervised learning, kernel, Gaussian processes.

1 Introduction

Semi-supervised learning uses unlabeled data in addition to labeled data to improve classification [See01]. Recently there has been significant interest in formulating the problem in terms of learning on graphs [BC01, SJ01, BN02b]. In a previous paper [ZGL03] we propose a Gaussian random field framework on graphs for semi-supervised learning. In this paper we will show the connection between the Gaussian random field framework and familiar kernel machines, in particular Gaussian processes. In addition, we derive hyperparameter learning for our Gaussian fields by evidence maximization, and provide an empirical study of different ways to parameterize the graph from data.

Let $\{(x_1, t_1) \cdots (x_l, t_l)\}$ be the labeled data. In this paper we assume binary classification and binary labels $t_i \in \{-1, 1\}$. Let $\{x_{l+1} \cdots x_{l+u}\}$ be the unlabeled data, and set $n = l + u$. We will often use L and U to denote the labeled and unlabeled data. We assume distance scores d_{ij} between any two instances x_i, x_j. For example, d_{ij} might be the Euclidean distance if $x_i \in \mathbb{R}^m$. The distance scores need not form a metric, but they must be symmetric and non-negative.

2 From Gaussian Fields to Gaussian Processes

In previous work [ZGL03], we propose the use of Gaussian random fields for semi-supervised learning. We form a graph over the labeled and unlabeled data, where the nodes of the graph represent instances $x_i, i = 1 \ldots n$, and where the edge weights represent the "local similarity" between pairs of instances. For instance, in [ZGL03] the graph is fully connected with the following edge weight function:

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\alpha^2}\right)$$

We denote the $n \times n$ edge weight matrix by $\mathbf{W}$. For each node i in the graph we assign a hidden continuous variable $y_i \in \mathbb{R}$, which can be thought of as a soft label. We define a conditional Gaussian random field[1] on the soft labels $\mathbf{y}$, conditioned on the covariates x and the constraints that for labeled data, $y_i = t_i, i \in L$:

$$p(\mathbf{y}) \quad \propto \quad \exp\left(-\frac{\beta}{4}\sum_{i,j} w_{ij}(y_i - y_j)^2\right) \;=\; \exp\left(-\frac{\beta}{2}\mathbf{y}^\top \Delta \mathbf{y}\right) \tag{1}$$

where β is an "inverse temperature parameter," and Δ is the *combinatorial Laplacian*, given in matrix form as $\Delta = \mathbf{D} - \mathbf{W}$ where $\mathbf{D} = \mathrm{diag}(d_i)$ is the diagonal matrix with entries $d_i = \sum_j w_{ij}$. The Gaussian field favors soft labels that change slowly across the graph. The mean of the Gaussian field is

$$\bar{\mathbf{y}_U} \quad = \quad {\Delta_{UU}}^{-1}\mathbf{W}_{UL}\mathbf{y}_L \tag{2}$$

conditioned on the soft labels $\mathbf{y}_L$, where Δ_{UU} is the sub-matrix of Laplacian for unlabeled data. Classification can be carried out by simply thresholding the mean.

[1] A Gaussian random field is simply a Markov random field with continuous states and a joint Gaussian distribution over those states. The edges in the graph correspond to non-zeros in the inverse covariance matrix, and represent the statistical dependence (and independence) relations in the field. For details see [Whi90].

Note that equation (1) is a Gaussian distribution. A Gaussian process restricted to finite data $x_1 \cdots x_n$ is simply a multivariate Gaussian distribution [Mac98]. This indicates a connection between Gaussian random fields and Gaussian processes; we will discuss this connection first under the finite case, and later propose extensions to the infinite case.

Equation (1) can be viewed as a Gaussian prior over $\mathbf{y}$ with covariance matrix $(\beta\Delta)^{-1}$. However it is an improper prior: Δ by definition has a zero eigenvalue (perhaps more than one) with eigenvector $\mathbf{1}$ (to see this note that the degree matrix $\mathbf{D}$ is the row sum of $\mathbf{W}$). This is undesirable—since Δ is singular, we can not compute the covariance matrix; moreover, any $\mathbf{y}$ shifted by a constant c will have the same likelihood. To make a proper prior out of the Laplacian, we can regularize its spectrum to remove the zero eigenvalues, as suggested in [SK03]. In particular, we choose to transform the eigenvalues according to the function

$$r(\lambda) = \lambda + 1/\sigma^2 \tag{3}$$

which results in the *regularized Laplacian* $\Delta + \mathbf{I}/\sigma^2$. Using the regularized Laplacian we define the zero mean Gaussian process prior as

$$p(\mathbf{y}) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \tilde{\Delta}\mathbf{y}\right) \tag{4}$$

which corresponds to a kernel with Gram matrix (covariance matrix) $\mathbf{G} = \tilde{\Delta}^{-1} = [\beta(\Delta + \mathbf{I}/\sigma^2)]^{-1}$. We note several important aspects of the resulting Gaussian process:

- $\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\Delta}^{-1}\right)$;
- $\tilde{\Delta}$ is non-singular, unlike Δ, resulting in a proper prior.
- The parameter β controls the overall sharpness of the distribution; large β means $p(\mathbf{y})$ is more peaked around its mean.
- The parameter σ^2 controls the amount of regularization; large σ regularizes less.
- The kernel, or covariance matrix $\mathbf{G} = \tilde{\Delta}^{-1}$ is the inverse of a function of the Laplacian Δ. Therefore the covariance between any two point i, j in general depends on *all* points—all of the unlabeled data is used to define the prior.

The last point is worth emphasizing. In many familiar Gaussian process kernels the entries are "local." For example, in a radial basis function (RBF) kernel $\mathbf{K}$, the matrix entry $k_{ij} = \exp\left(-d_{ij}^2/\alpha^2\right)$ only depends on the distance between i, j and *not any other points*. In this case unlabeled data is useless because the influence of such data in $\mathbf{K}$ is marginalized out. In contrast, the matrix entries of the kernel $\mathbf{G}_{ij}$ in (4) depends on all entries in Δ, and therefore on the distances between all pairs of instances. Thus, distribution of unlabeled data will strongly influence the kernel, which is desirable for semi-supervised learning. In other words, in RBF (and many other) kernels we parameterize the covariance matrix directly, while here we parameterize the *inverse* covariance matrix.

In moving from Gaussian fields to Gaussian processes, we no longer assume that the soft labels $\mathbf{y}_L$ for the labeled data are fixed to the observed labels $\mathbf{t}_L$. Instead we now assume the data generation process is $\mathbf{x} \to \mathbf{y} \to \mathbf{t}$, where $\mathbf{y} \to \mathbf{t}$ is a noisy label generation process with a sigmoid noise output model between the hidden soft labels y_i and observed labels t_i:

$$P(t_i|y_i) = \frac{e^{\gamma y_i t_i}}{e^{\gamma y_i t_i} + e^{-\gamma y_i t_i}} = \frac{1}{1 + e^{-2\gamma y_i t_i}} \tag{5}$$

where γ is a hyperparameter which controls the steepness of the sigmoid. This assumption allows us to handle noise in training labels, and is a common practice in Gaussian process classification.

We are interested in $p(\mathbf{t}_U|\mathbf{t}_L)$, the labels for unlabeled data. This is easy once we know the posterior distribution $p(\mathbf{y}_L, \mathbf{y}_U|\mathbf{t}_L)$. By Bayes' theorem,

$$p(\mathbf{y}_L, \mathbf{y}_U|\mathbf{t}_L) = \frac{\prod_{i=1}^{l} P(t_i|y_i)p(\mathbf{y}_L, \mathbf{y}_U)}{P(\mathbf{t}_L)} \tag{6}$$

Because of the noise model, the posterior is not Gaussian and has no closed form solution. We choose to use the Laplace approximation to find the approximate $p(\mathbf{y}_L, \mathbf{y}_U|\mathbf{t}_L)$. The derivation is given in Appendix A, which largely follows [Her02] (B.7).

Bayesian classification is based on the posterior distribution $p(\mathbf{y}_U|\mathbf{t}_L)$. Since under the Laplace approximation this distribution is also Gaussian, the classification rule depends only on the sign of the mode $\hat{\mathbf{y}_U}$.

3 The Choice of Kernel

The most important aspect of Gaussian process inference lies in the choice of prior; that is, in the choice of kernel. It is rarely the case that we are simply "given" the correct kernel; rather, the kernel will often need to be learned from data. There are several design decisions one has to make.

The first set of design decisions involve how the kernel should be parameterized. Our kernel $\tilde{\Delta}^{-1}$ is ultimately derived from the graph weights $\mathbf{W}$. Given the distances d_{ij}, we would like the edge weights to decrease as the distances increases. Some possible ways to parameterize the graph include the following.

- Unweighted kNN graph: nodes i, j are connected by an edge of weight 1, if i is in j's k-nearest-neighborhood or vice versa. k is a hyperparameter that controls the density of the graph. Although small k may result in disconnected graphs (which means the graph Laplacian Δ has multiple zero eigenvalues), it does not pose a problem for us since these zero eigenvalues go away in the smoothed Laplacian $\tilde{\Delta}$. kNN has the nice property of "adaptive scales," because the distance with a edge is different in low and high density regions.

- Unweighted ϵNN graph: nodes i, j are connected by an edge of weight 1, if $d_{ij} \le \epsilon$. The hyperparameter ϵ controls neighborhood radius. Although ϵ is continuous, the search for the optimal hyperparameter only need to take place at at most n^2 values (the edge lengths in the graph).

- tanh-weighted graph: $w_{ij} = (\tanh(\alpha_1(d_{ij} - \alpha_2)) + 1)/2$. The intuition is to create a soft cutoff around length α_2, so that close examples (presumably from the same class) are connected and examples from different classes (presumably with large distance) are nearly disconnected. The hyperbolic tangent function simulates ϵNN in that when $d_{ij} \gg \alpha_2$, $w_{ij} \approx 0$; $d_{ij} \ll \alpha_2$, $w_{ij} \approx 1$. The hyperparameters α_1, α_2 controls the slope and cutoff value respectively. Unlike ϵNN, tanh-weighted graph is continuous with respect to α_1, α_2 and is amenable to learning with gradient methods.

- exp-weighted graph: $w_{ij} = \exp(-d_{ij}^2/\alpha^2)$. Again this is a continuous weighting scheme similar to tanh-weights, but the cutoff is not as clear as tanh(). Hyperparameter α controls the decay rate.

Once a graph is parameterized and the Laplacian Δ computed, another set of design decisions involves manipulating the the eigensystem of Δ to create a family of kernels. For example, one can use only the first few eigenvector as basis for regression, as in [BN02a], or one can apply a regularization function on the eigenvalues, as in [CWS02] and more explicitly [SK03]. As mentioned earlier, our kernel $\tilde{\Delta}^{-1}$ employs a particular regularization function (3) on the eigenvalues. In this paper we will focus on the learning of the graph weights $\mathbf{W}$, with this fixed regularization function.

To favor one kernel over another, we need an optimization criterion. Two possible criteria are the following.

- Maximize the likelihood of labeled data (evidence maximization). One can choose the hyperparameters that maximize the log likelihood of labeled data: $\Theta^* = \arg\max_\Theta \log p(\mathbf{t}_L|\Theta)$. $\log p(\mathbf{t}_L|\Theta)$ is known as the evidence and the procedure is also called evidence maximization. One can also assume a prior on Θ and find the maximum a posteriori (MAP) estimate $\Theta^* = \arg\max_\Theta \log p(\mathbf{t}_L|\Theta) + \log p(\Theta)$. The evidence can be multimodal and usually gradient methods are used to find a mode in hyperparameter space. This requires the derivatives $\partial \log p(\mathbf{t}_L|\Theta)/\partial\Theta$. We provide a complete derivation for the derivatives in Appendix B, which largely follows [WB98].

  In a full Bayesian setup, one would average over all hyperparameters (weighted by the posterior $p(\Theta|\mathbf{t}_L)$) instead of using the single best estimate Θ^*. This usually involves Markov Chain Monte Carlo techniques and is not pursued in this paper.

- Maximize the alignment to labeled data. Alternatively one can regard the matrix $\mathbf{y}_L{\mathbf{y}_L}^\top$ as the "target covariance matrix," and maximize the alignment to this target covariance matrix. The alignment is a generalized cosine similarity defined on matrices. For details see [CSTEK01]. The alignment is related to evidence maximization but is different. For example, the kernel $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ aligns better with the labels $[1 -1]$ than the kernel $\begin{bmatrix} 1 & -0.6 \\ -0.6 & 0.36 \end{bmatrix}$, but both maximize the evidence given a step function noise model. We do not follow this path in this paper, but note it as an interesting possible direction.

4 Experiments

We experiment on the following binary classification data sets. A detailed description for most datasets can be found in [ZLG03].

- **1/2**: Optical character recognition of handwritten digits '1' vs. '2'. d_{ij} is pixel-wise Euclidean distance between two images. There are 1100 positive examples ('1's) and 1100 negative examples ('2's). We randomly sample $L = 20$ labeled examples and use the rest as unlabeled data.

- **2/3**: Similarly '2' vs. '3'. 1100 positive examples ('2's) and 1100 negative examples ('3's). $L = 20$.

- **o/e**: Odd digits (1,3,5,7,9) vs. even digits (2,4,6,8,0). Each digit has 400 images, thus 2000 positive and 2000 negative examples. $L = 50$.

- **b/h**: Text document classification with newsgroups `rec.sport.baseball` (994 documents) vs. `rec.sport.hockey` (999 documents). $d_{ij} = 1 - CS(x_i, x_j)$ where CS is the cosine similarity on *tf.idf* document vectors. $L = 20$.

	kNN unweighted		ϵNN unweighted		tanh() weights			exp() weights	
	$\log p(\mathbf{t}_L)$	k	$\log p(\mathbf{t}_L)$	ϵ	$\log p(\mathbf{t}_L)$	α_1	α_2	$\log p(\mathbf{t}_L)$	α
1/2	-12.5	2	**-12.1**	347.2	-12.2	-0.036	261.1	-13.0	124.3
2/3	**-15.0**	2	-18.8	561.9	-16.8	-0.018	315.8	-16.9	196.4
o/e	-38.3	3	-40.2	492.8	**-37.3**	-0.027	313.5	-40.1	163.8
b/h	**-17.3**	2	-18.9	0.91	-18.7	-81.7	0.88	-19.2	0.57
p/m	-19.3	5	-19.4	0.91	**-17.8**	-53.3	0.80	-19.5	0.59
r/a	-39.3	4	-40.0	0.90	**-38.3**	-49.5	0.82	-40.7	0.86

Table 1: Evidence $\log p(\mathbf{t}_L)$ and optimal hyperparameters Θ^* under different parameterizations of the graph.

	kNN unweighted		ϵNN unweighted		tanh() weights		exp() weights	
	acc	tuned	acc	tuned	acc	tuned	acc	tuned
1/2	0.978	0.990	0.922	0.922	0.996	**0.997**	0.989	**0.997**
2/3	0.976	**0.993**	0.499	0.980	0.503	0.992	0.500	0.988
o/e	0.906	**0.904**	0.525	0.811	0.710	0.796	0.686	0.736
b/h	0.892	0.892	0.501	0.969	0.512	**0.977**	0.500	0.851
p/m	0.506	0.873	0.501	0.856	0.768	**0.897**	0.494	0.702
r/a	0.827	0.799	0.619	0.808	0.779	**0.815**	0.560	0.738

Table 2: Classification accuracy on unlabeled data, threshold at 0 (acc) or with class proportion knowledge (tuned). These are computed at Θ^* respectively.

- **p/m**: Document classification with newsgroups `comp.sys.ibm.pc.hardware` (982 documents) vs. `comp.sys.mac.hardware` (961 documents). $L = 20$.
- **r/a**: Newsgroups `talk.religion.misc` (628 documents) vs. `alt.atheism` (799 documents). $L = 50$.

To facilitate computation, we fix the regularization parameter $\sigma = 1000$, the temperature parameter $\beta = 0.1$ and the noise parameter $\gamma = 1$, and learn the remaining graph hyperparameters. These values for σ, β and γ are chosen by their good empirical results across different weight parameterizations. Table 1 and Table 2 compare the remaining graph hyperparameters which were learned. The best results are marked in bold font.

In Table 1 we show the optimal evidence with the corresponding learned hyperparameters Θ^* for the different graphs. No prior on Θ was used. To search for Θ^* for the kNN graphs we enumerate k from 1 to 128; for the ϵNN graphs we use a multi-resolution grid search on ϵ; for the tanh and exp graphs we use the conjugate gradient search program of [Ras] with arbitrary starting points.

Table 2 shows the classification accuracy on the unlabeled data under the optimal hyperparameters Θ^*. The standard Bayesian classification rule is to threshold $\hat{\mathbf{y}_U}$ at 0. But in practice we find that $\hat{\mathbf{y}_U}$ is often biased–for example, all of the values can be less than zero. To solve the problem, we use the additional knowledge of class proportions, e.g. "50% of unlabeled data are class 1." We tune the threshold on $\hat{\mathbf{y}_U}$ so that the class proportions are met. This heuristic greatly stabilizes the accuracy, as shown in the "tuned" columns.

To get a better idea of the effect of hyperparameter learning under the different graph parameterizations, we plot the evidence and (tuned) classification accuracy surfaces by systematically varying the hyperparameters.

- The evidence and tuned accuracy for kNN unweighted graphs is shown in Figure 1; note the log-scaled x-axis. The vertical lines mark the optimal k. A large range of small k's achieves high evidence and high accuracy. There is good correlation between evidence and tuned accuracy. It appears that small k is good in general.

- The evidence and tuned accuracy for ϵNN unweighted graphs is shown in Figure 2. The evidence curves have a more zig-zag shape. Another observation is that the number of edges under the optimal ϵ is much larger (sometimes by a few orders of magnitude) than that of the kNN graphs.

- The evidence for tanh()-weighted graphs is shown in Figure 3. The vertical lines mark the position of α_1, α_2 found by gradient ascent. The corresponding tuned classification accuracy surface is in Figure 4.

- The evidence and tuned accuracy for exp() weighted graphs is shown in Figure 5. This seems to be the worst among the four. We suspect this is because the exp() function cannot quickly kill off unwanted edges between different classes. The graph kernel is sensitive to such spurious edges.

5 Extension to Unseen Examples

We have so far restricted the Gaussian process to the nodes in the graph. In this finite case Gaussian processes are nothing but n-dimensional joint normal distributions; that is, Gaussian processes and Gaussian random fields are in this case equivalent. However Gaussian fields, by definition, cannot handle unseen instances, since any new instances need to become additional nodes in the graph, resulting in changes in the Laplacian and kernel matrices.

To define a proper Gaussian process, we need to extend the framework to the infinite space of possible $\mathbf{X}$ values. Equivalently, this is the problem of handling new input data in a transductive framework. Let x be a new instance that is in neither the labeled nor the unlabeled dataset. One simple strategy is to extend the kernel as

$$\mathbf{K}(x, x_i) \;\equiv\; \mathbf{G}(x^*, x_i),\; i \in L \tag{7}$$

Note that we only need to define the extended kernel $\mathbf{K}$ with respect to labeled instances since these are sufficient for computation in Gaussian processes. $\mathbf{G} = \tilde{\Delta}^{-1}$ is the graph kernel on the finite labeled and unlabeled set, and x^* is the example in the labeled and unlabeled set that is closest to x:

$$x^* \;=\; \arg\min_{z \in L \cup U} d_{xz} \tag{8}$$

Essentially we divide the input space into Voronoi cells generated by instances in $L \cup U$, and map any new instance x to its Voronoi cell representative. This mapping, however, is done with the original distance measure d_{xz}, which might be the Euclidean distance in the embedding space X, for example. This is an approximation to the graph distance (or manifold geodesic) we should have been using. Nonetheless, when the unlabeled data size is large the approximation is reasonable. From an algorithmic point of view, we classify x the same as its 1-nearest-neighbor in $L \cup U$.

One can take one step further and map x to the closest point in the space spanned by instances in $L \cup U$, i.e. as a linear combination of points in $L \cup U$. This is the approach taken by [CWS02]. Here closeness is again measured in the embedding space. This might give a better approximation, but the computation involved is more expensive, as one needs to invert a $n \times n$ matrix. When U is large the cost may not justify the gain compared to the simple Voronoi method.

6 Discussion

The empirical experiments presented above suggest that hyperparameters learned from evidence maximization correspond reasonably well to classification accuracy. However the computation involved is significant, because at each step we have to re-compute the covariance matrix $\mathbf{G} = \tilde{\Delta}^{-1}$ for the updated hyperparameters. Unless a fast algorithm is used, this will be the bottleneck that limits the application of graph weight learning. This is in general true for Gaussian process classifiers. Approximate methods exist, see e.g. [Gib97].

In contrast, methods that work under fixed graph weights but manipulate the graph spectrum [SK03] learn the optimal eigenvalues efficiently while keeping the eigenvectors fixed. But they lack the flexibility that changing graph weights provides. Such flexibility is important for irrelevant feature detection, for example. It would be interesting to combine the two.

From a practical point of view, our experiments indicate that tanh() weights are overall (although not always) the better graph parameterization, although only slightly. But for simplicity, kNN unweighted graphs are preferred, as finding a good k is much easier.

Appendix A: Laplace Approximation for Gaussian Processes

This section largely follows [Her02] (B.7). The Gaussian process model, restricted to the labeled and unlabeled data, is

$$\mathbf{y} \sim \mathcal{N}\left(\mu, \tilde{\Delta}^{-1}\right) \tag{9}$$

We will use $\mathbf{G} = \tilde{\Delta}^{-1}$ to denote the covariance matrix (i.e. the Gram matrix). Let $\mathbf{t} \in \{-1, +1\}$ be the observed discrete class labels. The hidden variable $\mathbf{y}$ and labels $\mathbf{t}$ are connected via a sigmoid noise model

$$P(t_i|y_i) = \frac{e^{\gamma y_i t_i}}{e^{\gamma y_i t_i} + e^{-\gamma y_i t_i}} = \frac{1}{1 + e^{-2\gamma y_i t_i}} \tag{10}$$

where γ is a hyperparameter which controls the steepness of the sigmoid. Given the prior and the noise model, we are interested in the posterior $p(\mathbf{y}_L, \mathbf{y}_U|\mathbf{t}_L)$. By Bayes theorem,

$$p(\mathbf{y}_L, \mathbf{y}_U|\mathbf{t}_L) = \frac{\prod_{i=1}^{l} P(t_i|y_i) p(\mathbf{y}_L, \mathbf{y}_U)}{P(\mathbf{t}_L)} \tag{11}$$

Because of the noise model, the posterior is not Gaussian and has no closed form solution. We use the Laplace approximation.

First we find the mode of the posterior (6):

$$\begin{aligned}
(\hat{\mathbf{y}_L}, \hat{\mathbf{y}_U}) &= \arg\max_{\mathbf{y}_L,\mathbf{y}_U} \frac{\prod_{i=1}^{l} P(t_i|y_i) p(\mathbf{y}_L, \mathbf{y}_U)}{P(\mathbf{t}_L)} && (12)\\
&= \arg\max_{\mathbf{y}_L,\mathbf{y}_U} \sum_{i=1}^{l} \ln P(t_i|y_i) + \ln p(\mathbf{y}_L, \mathbf{y}_U) && (13)\\
&= \arg\max_{\mathbf{y}_L,\mathbf{y}_U} Q_1 + Q_2 && (14)
\end{aligned}$$

Note $\mathbf{y}_U$ only appears in Q_2, and we can maximize $\hat{\mathbf{y}_U}$ independently given $\hat{\mathbf{y}_L}$. Q_2 is the log likelihood of the Gaussian (9). Therefore given $\hat{\mathbf{y}_L}$, $\mathbf{y}_U$ follows the conditional distribution of Gaussian:

$$p(\mathbf{y}_U|\hat{\mathbf{y}_L}) = \mathcal{N}\left(\mathbf{G}_{UL}\mathbf{G}_{LL}{}^{-1}\hat{\mathbf{y}_L}, \mathbf{G}_{UU} - \mathbf{G}_{UL}\mathbf{G}_{LL}{}^{-1}\mathbf{G}_{LU}\right) \tag{15}$$

and the mode is the conditional mean

$$\hat{\mathbf{y}_U} = \mathbf{G}_{UL}\mathbf{G}_{LL}{}^{-1}\hat{\mathbf{y}_L} \tag{16}$$

It's easy to see (16) has the same form as the solution for Gaussian Fields (2): Recall $\mathbf{G} = \tilde{\Delta}^{-1}$. From partitioned matrix inversion theorem,

$$\tilde{\Delta}_{UU} = S_A^{-1}$$

$$\tilde{\Delta}_{UL} = -S_A^{-1}\mathbf{G}_{UL}\mathbf{G}_{LL}^{-1}$$

where $S_A = \mathbf{G}_{UU} - \mathbf{G}_{UL}(\mathbf{G}_{LL})^{-1}\mathbf{G}_{LU}$ is the Schur complement of $\mathbf{G}_{LL}$. This gives us

$$-(\tilde{\Delta}_{UU})^{-1}\tilde{\Delta}_{UL} = S_A S_A^{-1}\mathbf{G}_{UL}\mathbf{G}_{LL}^{-1} = \mathbf{G}_{UL}\mathbf{G}_{LL}^{-1}$$

Thus we have

$$\begin{aligned}
\hat{\mathbf{y}_U} &= -\tilde{\Delta}_{UU}^{-1}\tilde{\Delta}_{UL}\hat{\mathbf{y}_L} && (17)\\
&= \tilde{\Delta}_{UU}^{-1}\mathbf{W}_{UL}\hat{\mathbf{y}_L} && (18)
\end{aligned}$$

which has the same form as the harmonic energy minimizing function in [ZGL03]. In fact the latter is the limiting case when $\sigma^2 \to \infty$ and there is no noise model.

Substitute (16) back to Q_2, using partitioned inverse of a matrix it can be shown that (not surprisingly)

$$Q_2 = -\frac{1}{2}\mathbf{y}_L^\top \mathbf{G}_{LL}{}^{-1}\mathbf{y}_L + c \tag{19}$$

Now go back to Q_1. The noise model can be written as

$$\begin{aligned}
P(t_i|y_i) &= \frac{e^{\gamma y_i t_i}}{e^{\gamma y_i t_i} + e^{-\gamma y_i t_i}} && (20)\\
&= \left(\frac{e^{\gamma y_i}}{e^{\gamma y_i} + e^{-\gamma y_i}}\right)^{\frac{t_i+1}{2}} \left(1 - \frac{e^{\gamma y_i}}{e^{\gamma y_i} + e^{-\gamma y_i}}\right)^{\frac{1-t_i}{2}} && (21)\\
&= \pi(y_i)^{\frac{t_i+1}{2}}(1 - \pi(y_i))^{\frac{1-t_i}{2}} && (22)
\end{aligned}$$

therefore

$$Q_1 = \sum_{i=1}^{l} \ln P(t_i|y_i) \tag{23}$$

$$= \sum_{i=1}^{l} \frac{t_i+1}{2} \ln \pi(y_i) + \frac{1-t_i}{2} \ln(1-\pi(y_i)) \tag{24}$$

$$= \gamma(\mathbf{t}_L - \mathbf{1})^\top \mathbf{y}_L - \sum_{i=1}^{l} \ln(1+e^{-2\gamma y_i}) \tag{25}$$

Put it together,

$$\hat{\mathbf{y}_L} = \arg\max Q_1 + Q_2 \tag{26}$$

$$= \arg\max \gamma(\mathbf{t}_L - \mathbf{1})^\top \mathbf{y}_L - \sum_{i=1}^{l} \ln(1+e^{-2\gamma y_i}) - \frac{1}{2}\mathbf{y}_L^\top \mathbf{G}_{LL}{}^{-1}\mathbf{y}_L \tag{27}$$

To find the mode, we take the derivative,

$$\frac{\partial(Q_1+Q_2)}{\partial \mathbf{y}_L} = \gamma(\mathbf{t}_L - \mathbf{1}) + 2\gamma(1-\pi(\mathbf{y}_L)) - \mathbf{G}_{LL}{}^{-1}\mathbf{y}_L \tag{28}$$

Because of the term $\pi(\mathbf{y}_L)$ it is not possible to find the root directly. We solve it with Newton-Raphson algorithm,

$$\mathbf{y}_L^{(t+1)} \leftarrow \mathbf{y}_L^{(t)} - \mathbf{H}^{-1} \left.\frac{\partial(Q_1+Q_2)}{\partial \mathbf{y}_L}\right|_{\mathbf{y}_L{}^{(t)}} \tag{29}$$

where $\mathbf{H}$ is the Hessian matrix,

$$\mathbf{H} = \left[\left.\frac{\partial^2(Q_1+Q_2)}{\partial y_i \partial y_j}\right|_{\mathbf{y}_L} \right] \tag{30}$$

Note $\frac{d}{dy_i}\pi(y_i) = 2\gamma\pi(y_i)(1-\pi(y_i))$, we can write $\mathbf{H}$ as

$$\mathbf{H} = -\mathbf{G}_{LL}{}^{-1} - \mathbf{P} \tag{31}$$

where $\mathbf{P}$ is a diagonal matrix with elements $P_{ii} = 4\gamma^2\pi(y_i)(1-\pi(y_i))$.

Once Newton-Raphson converges we compute $\hat{\mathbf{y}_U}$ from $\hat{\mathbf{y}_L}$ with (16). Classification can be done with $\mathrm{sgn}(\hat{\mathbf{y}_U})$ noting this is the Bayesian classification rule with Gaussian distribution and sigmoid noise model.

To compute the covariance matrix of the Laplace approximation, note by definition the inverse covariance matrix of the Laplace approximation is

$$\Sigma^{-1} = \left[\left.\frac{\partial^2 - \ln p(\mathbf{y}|\mathbf{t})}{\partial y_i \partial y_j}\right|_{\hat{\mathbf{y}_L}, \hat{\mathbf{y}_U}} \right] \tag{32}$$

From (6) it is straightforward to confirm

$$\Sigma^{-1} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \mathbf{G}^{-1} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \tilde{\Delta} \tag{33}$$

Therefore the covariance matrix is

$$\Sigma = \left(\begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \tilde{\Delta} \right)^{-1} \tag{34}$$

where $\mathbf{P}$ is evaluated at the mode $\hat{\mathbf{y}_L}$.

Appendix B: Hyperparameter Learning by Evidence Maximization

This section largely follows [WB98]. We want to find the MAP hyperparameters Θ which maximize the posterior

$$p(\Theta|\mathbf{t}_L) \propto p(\mathbf{t}_L|\Theta)p(\Theta)$$

The prior $p(\Theta)$ is usually chosen to be simple, and so we focus on the term $p(\mathbf{t}_L|\Theta)$, known as the *evidence.* The definition

$$p(\mathbf{t}_L|\Theta) = \int p(\mathbf{t}_L|\mathbf{y}_L)p(\mathbf{y}_L|\Theta)\; d\mathbf{y}_L$$

is hard to compute analytically. But notice

$$p(\mathbf{t}_L|\Theta) \quad = \quad \frac{p(\mathbf{t}_L|\mathbf{y}_L)p(\mathbf{y}_L|\Theta)}{p(\mathbf{y}_L|\mathbf{t}_L,\Theta)}, \forall \mathbf{y}_L \tag{35}$$

Since it holds for all $\mathbf{y}_L$, it holds for the mode of the Laplace approximation $\hat{\mathbf{y}_L}$:

$$p(\mathbf{t}_L|\Theta) = \frac{p(\mathbf{t}_L|\hat{\mathbf{y}_L})p(\hat{\mathbf{y}_L}|\Theta)}{p(\hat{\mathbf{y}_L}|\mathbf{t}_L,\Theta)}$$

The terms on the numerator are straight forward to compute; the denominator is tricky. However we can use the Laplace approximation, i.e. the probability density at the mode: $p(\hat{\mathbf{y}_L}|\mathbf{t}_L,\Theta) \approx \mathcal{N}(\hat{\mathbf{y}_L}|\hat{\mathbf{y}_L},\Sigma_{LL})$. Recall

$$\Sigma = \left(\begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{G}_{LL} & \mathbf{G}_{LU} \\ \mathbf{G}_{UL} & \mathbf{G}_{UU} \end{bmatrix}^{-1} \right)^{-1} \tag{36}$$

By applying Schur complement in block matrix decomposition twice, we find

$$\Sigma_{LL} = (\mathbf{P} + \mathbf{G}_{LL}^{-1})^{-1} \tag{37}$$

So the evidence is

$$\begin{aligned} p(\mathbf{t}_L|\Theta) \quad &\approx \quad \frac{p(\mathbf{t}_L|\hat{\mathbf{y}_L})p(\hat{\mathbf{y}_L}|\Theta)}{\mathcal{N}(\hat{\mathbf{y}_L}|\hat{\mathbf{y}_L},\Sigma_{LL})} && (38) \\ &= \quad \frac{p(\mathbf{t}_L|\hat{\mathbf{y}_L})p(\hat{\mathbf{y}_L}|\Theta)}{(2\pi)^{-\frac{n}{2}}|\Sigma_{LL}|^{-\frac{1}{2}}} && (39) \\ &= \quad \frac{p(\mathbf{t}_L|\hat{\mathbf{y}_L})p(\hat{\mathbf{y}_L}|\Theta)}{(2\pi)^{-\frac{n}{2}}|(\mathbf{P} + \mathbf{G}_{LL}^{-1})^{-1}|^{-\frac{1}{2}}} && (40) \end{aligned}$$

Switching to log domain, we have

$$\log p(\mathbf{t}_L|\Theta) \approx \Psi(\hat{\mathbf{y}_L}) + \frac{n}{2}\log 2\pi + \frac{1}{2}\log|\Sigma_{LL}| \tag{41}$$

$$= \Psi(\hat{\mathbf{y}_L}) + \frac{n}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{P} + \mathbf{G}_{LL}^{-1}| \tag{42}$$

where $\Psi(\mathbf{y}_L) = \log p(\mathbf{t}_L|\mathbf{y}_L) + \log p(\mathbf{y}_L|\Theta)$. Since $\mathbf{y} \sim \mathcal{N}\left(\mu, \tilde{\Delta}^{-1}\right) = \mathcal{N}(\mu, \mathbf{G})$, we have $\mathbf{y}_L \sim \mathcal{N}(\mu_L, \mathbf{G}_{LL})$. Therefore

$$\Psi(\hat{\mathbf{y}_L}) = \log p(\mathbf{t}_L|\hat{\mathbf{y}_L}) + \log p(\hat{\mathbf{y}_L}|\Theta) \tag{43}$$

$$= -\sum_{i=1}^{L}\log(1+\exp(-2\gamma\hat{y_i}t_i))$$
$$-\frac{n}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{G}_{LL}| - \frac{1}{2}(\hat{\mathbf{y}_L}-\mu_L)^\top\mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L}-\mu_L) \tag{44}$$

Put it together,

$$\log p(\mathbf{t}_L|\Theta) \approx -\sum_{i=1}^{L}\log(1+\exp(-2\gamma\hat{y_i}t_i))$$
$$-\frac{1}{2}\log|\mathbf{G}_{LL}| - \frac{1}{2}(\hat{\mathbf{y}_L}-\mu_L)^\top\mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L}-\mu_L) - \frac{1}{2}\log|\mathbf{P}+\mathbf{G}_{LL}^{-1}|$$
$$= -\sum_{i=1}^{L}\log(1+\exp(-2\gamma\hat{y_i}t_i))$$
$$-\frac{1}{2}(\hat{\mathbf{y}_L}-\mu_L)^\top\mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L}-\mu_L) - \frac{1}{2}\log|\mathbf{G}_{LL}\mathbf{P}+\mathbf{I}| \tag{45}$$

This gives us a way to (approximately) compute the evidence.

To find the MAP estimate of Θ (which can have multiple local maxima), we use gradient methods. This involves the derivatives of the evidence $\partial\log p(\mathbf{t}_L|\Theta)/\partial\theta$, where θ is the hyperparameter β, σ, γ or the ones controlling W.

We start from

$$\frac{\partial}{\partial\theta}\pi(\hat{y_i}) = \frac{\partial}{\partial\theta}\frac{1}{1+e^{-2\gamma\hat{y_i}}} \tag{46}$$

$$= 2\pi(\hat{y_i})(1-\pi(\hat{y_i}))(\hat{y_i}\frac{\partial\gamma}{\partial\theta} + \gamma\frac{\partial\hat{y_i}}{\partial\theta}) \tag{47}$$

To compute $\partial\hat{\mathbf{y}_L}/\partial\theta$, note the Laplace approximation mode $\hat{\mathbf{y}_L}$ satisfies

$$\left.\frac{\partial\Psi(\mathbf{y}_L)}{\partial\mathbf{y}_L}\right|_{\hat{\mathbf{y}_L}} = \gamma(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) - \mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L}-\mu_L) = 0 \tag{48}$$

which means

$$\hat{\mathbf{y}_L} = \gamma\mathbf{G}_{LL}(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) + \mu_L \tag{49}$$

Taking derivatives on both sides,

$$\frac{\partial\hat{\mathbf{y}_L}}{\partial\theta} = \frac{\partial}{\partial\theta}\gamma\mathbf{G}_{LL}(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) \tag{50}$$

$$= \frac{\partial\gamma\mathbf{G}_{LL}}{\partial\theta}(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) - 2\gamma\mathbf{G}_{LL}\frac{\partial\pi(\hat{\mathbf{y}_L})}{\partial\theta} \tag{51}$$

$$= \frac{\partial\gamma\mathbf{G}_{LL}}{\partial\theta}(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) - \frac{1}{\gamma}\mathbf{G}_{LL}\mathbf{P}\hat{\mathbf{y}_L}\frac{\partial\gamma}{\partial\theta} - \mathbf{G}_{LL}\mathbf{P}\frac{\partial\hat{\mathbf{y}_L}}{\partial\theta} \tag{52}$$

which gives

$$\frac{\partial \hat{\mathbf{y}_L}}{\partial \theta} = (\mathbf{I} + \mathbf{G}_{LL}\mathbf{P})^{-1} \left[\frac{\partial \gamma \mathbf{G}_{LL}}{\partial \theta}(\mathbf{t}_L + \mathbf{1} - 2\pi(\hat{\mathbf{y}_L})) - \frac{1}{\gamma}\mathbf{G}_{LL}\mathbf{P}\hat{\mathbf{y}_L}\frac{\partial \gamma}{\partial \theta} \right] \tag{53}$$

Now it is straight forward to compute the gradient with (45):

$$\begin{aligned} & \frac{\partial}{\partial \theta} \log p(\mathbf{t}_L|\Theta) \\ \approx\ & \frac{\partial}{\partial \theta}\left[-\sum_{i=1}^{L} \log(1+\exp(-2\gamma \hat{y}_i t_i)) - \frac{1}{2}(\hat{\mathbf{y}_L} - \mu_L)^\top \mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L} - \mu_L) - \frac{1}{2}\log|\mathbf{G}_{LL}\mathbf{P} + \mathbf{I}| \right] \\ =\ & -\sum_{i=1}^{L} \frac{\exp(-2\gamma \hat{y}_i t_i)(-2t_i)}{1+\exp(-2\gamma \hat{y}_i t_i)}(\hat{y}_i \frac{\partial \gamma}{\partial \theta} + \gamma \frac{\partial \hat{y}_i}{\partial \theta}) \\ & -\frac{1}{2}\left[2(\mathbf{G}_{LL}^{-1}(\hat{\mathbf{y}_L} - \mu_L))^\top \frac{\partial \hat{\mathbf{y}_L}}{\partial \theta} + (\hat{\mathbf{y}_L} - \mu_L)^\top \frac{\partial \mathbf{G}_{LL}^{-1}}{\partial \theta}(\hat{\mathbf{y}_L} - \mu_L) \right] \\ & -\frac{1}{2} tr\left((\mathbf{G}_{LL}\mathbf{P} + \mathbf{I})^{-1} \frac{\partial \mathbf{G}_{LL}\mathbf{P}}{\partial \theta} \right) \end{aligned} \tag{54}$$

where we used the fact

$$\frac{\partial \log |A|}{\partial \theta} = tr\left(A^{-1}\frac{\partial A}{\partial \theta} \right) \tag{55}$$

For example, if $\theta = \gamma$, the gradient can be computed by noting $\frac{\partial \gamma \mathbf{G}_{LL}}{\partial \gamma} = \mathbf{G}_{LL}$, $\frac{\partial \gamma}{\partial \gamma} = 1$, $\frac{\partial \mathbf{G}_{LL}^{-1}}{\partial \gamma} = 0$, and $\frac{\partial \mathbf{G}_{LL}\mathbf{P}}{\partial \gamma} = \mathbf{G}_{LL}\frac{\partial \mathbf{P}}{\partial \gamma}$ where $\frac{\partial \mathbf{P}_{ii}}{\partial \gamma} = 8\gamma\pi(\hat{y}_i)(1-\pi(\hat{y}_i)) + 4\gamma^2(1-2\pi(\hat{y}_i))\frac{\partial \pi(\hat{y}_i)}{\partial \gamma}$.

For $\theta = \beta$, we have $\frac{\partial \gamma \mathbf{G}_{LL}}{\partial \beta} = \gamma(-1/\beta)\mathbf{G}_{LL}$, $\frac{\partial \gamma}{\partial \beta} = 0$, $\frac{\partial \mathbf{G}_{LL}^{-1}}{\partial \beta} = \mathbf{G}_{LL}^{-1}/\beta$, and $\frac{\partial \mathbf{G}_{LL}\mathbf{P}}{\partial \beta} = -\mathbf{G}_{LL}\mathbf{P}/\beta + \mathbf{G}_{LL}\frac{\partial \mathbf{P}}{\partial \beta}$ where $\frac{\partial \mathbf{P}_{ii}}{\partial \beta} = 8\gamma^3\pi(\hat{y}_i)(1-\pi(\hat{y}_i))(1-2\pi(\hat{y}_i))\frac{\partial \hat{y}_i}{\partial \beta}$.

For $\theta = \sigma$, the computation is more intensive because the complex dependency between $\mathbf{G}$ and σ. We start from $\frac{\partial \mathbf{G}_{LL}}{\partial \sigma} = \left[\frac{\partial \mathbf{G}}{\partial \sigma}\right]_{LL}$. Using the fact $\frac{\partial A^{-1}}{\partial \theta} = -A^{-1}\frac{\partial A}{\partial \theta}A^{-1}$ and $\mathbf{G} = \tilde{\Delta}^{-1}$, we get $\frac{\partial \mathbf{G}}{\partial \sigma} = \beta/\sigma^3\mathbf{G}^2$. Note the computation involves the multiplication of the *full* matrix $\mathbf{G}$ and is thus more demanding. Once $\frac{\partial \mathbf{G}_{LL}}{\partial \sigma}$ is computed the rest is easy.

If we parameterize the weights $\mathbf{W}$ in Gaussian Fields with radial basis functions (for simplicity we assume a single length scale parameter α for all dimensions. Extension to multiple length scales is simple),

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\alpha^2} \right) \tag{56}$$

where d_{ij} is the Euclidean distance between x_i, x_j in the original feature space, we can similarly learn the hyperparameter α. Note $\frac{\partial w_{ij}}{\partial \alpha} = w_{ij}\frac{d_{ij}^2}{\alpha^3}$, $\frac{\partial \Delta}{\partial \alpha} = \frac{\partial \mathbf{D}}{\partial \alpha} - \frac{\partial \mathbf{W}}{\partial \alpha}$, $\frac{\partial \tilde{\Delta}}{\partial \alpha} = \beta\frac{\partial \Delta}{\partial \alpha}$. The rest is the same as for σ above.

Similarly with a tanh()-weighted weight function $w_{ij} = (\tanh(\alpha_1(d_{ij} - \alpha_2)) + 1)/2$, we have $\frac{\partial w_{ij}}{\partial \alpha_1} = (1 - \tanh^2(\alpha_1(d_{ij} - \alpha_2)))(d_{ij} - \alpha_2)/2$ and $\frac{\partial w_{ij}}{\partial \alpha_2} = -(1 - \tanh^2(\alpha_1(d_{ij} - \alpha_2)))\alpha_1/2$, and the rest follows.

References

[BC01] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, 2001.

[BN02a] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on manifolds. Technical Report TR-2002-12, University of Chicago, 2002.

[BN02b] Mikhail Belkin and Partha Niyogi. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems, 15*, volume 15, 2002.

[CSTEK01] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. In *Advances in NIPS*, 2001.

[CWS02] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems, 15*, volume 15, 2002.

[Gib97] Mark N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, University of Cambridge, 1997.

[Her02] Ralf Herbrich. *Learning Kernel Classifiers.* The MIT press, 2002.

[Mac98] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998.

[Ras] Carl Edward Rasmussen. http://www.gatsby.ucl.ac.uk/ edward/code/minimize/.

[See01] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[SJ01] Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems, 14*, volume 14, 2001.

[SK03] A. Smola and R. Kondor. Kernels and regularization on graphs. In *Conference on Learning Theory, COLT/KW*, 2003.

[WB98] Christopher K. I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[Whi90] J.L. Whittaker. *Graphical Models in Applied Multivariate Statistics.* John Wiley and Sons, 1990.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML-03, 20th International Conference on Machine Learning*, 2003.

[ZLG03] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

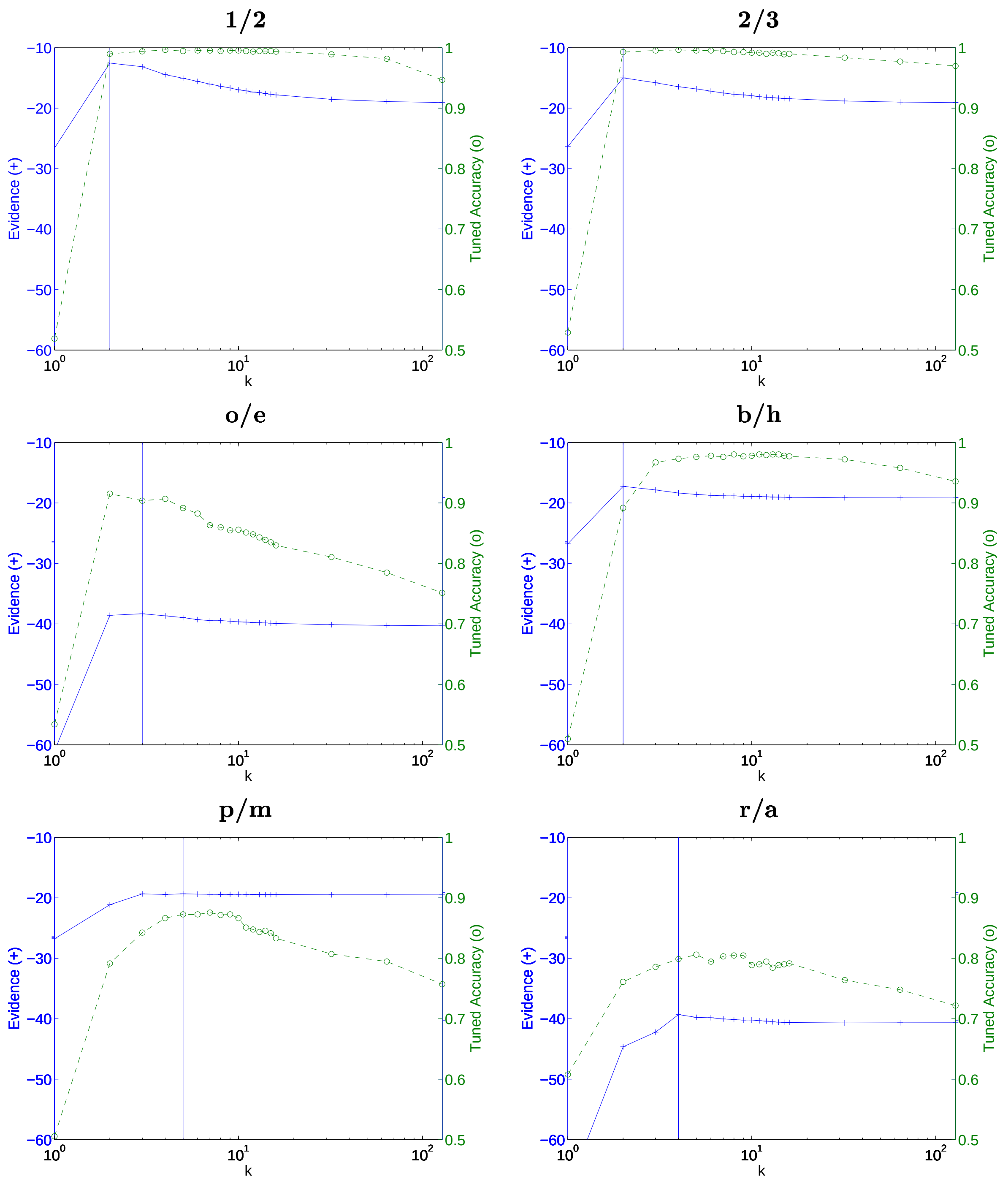

Figure 1: Evidence and tuned accuracy curves of kNN unweighted graphs, with k ranging from 1 to 128. The vertical lines mark the optimal k.

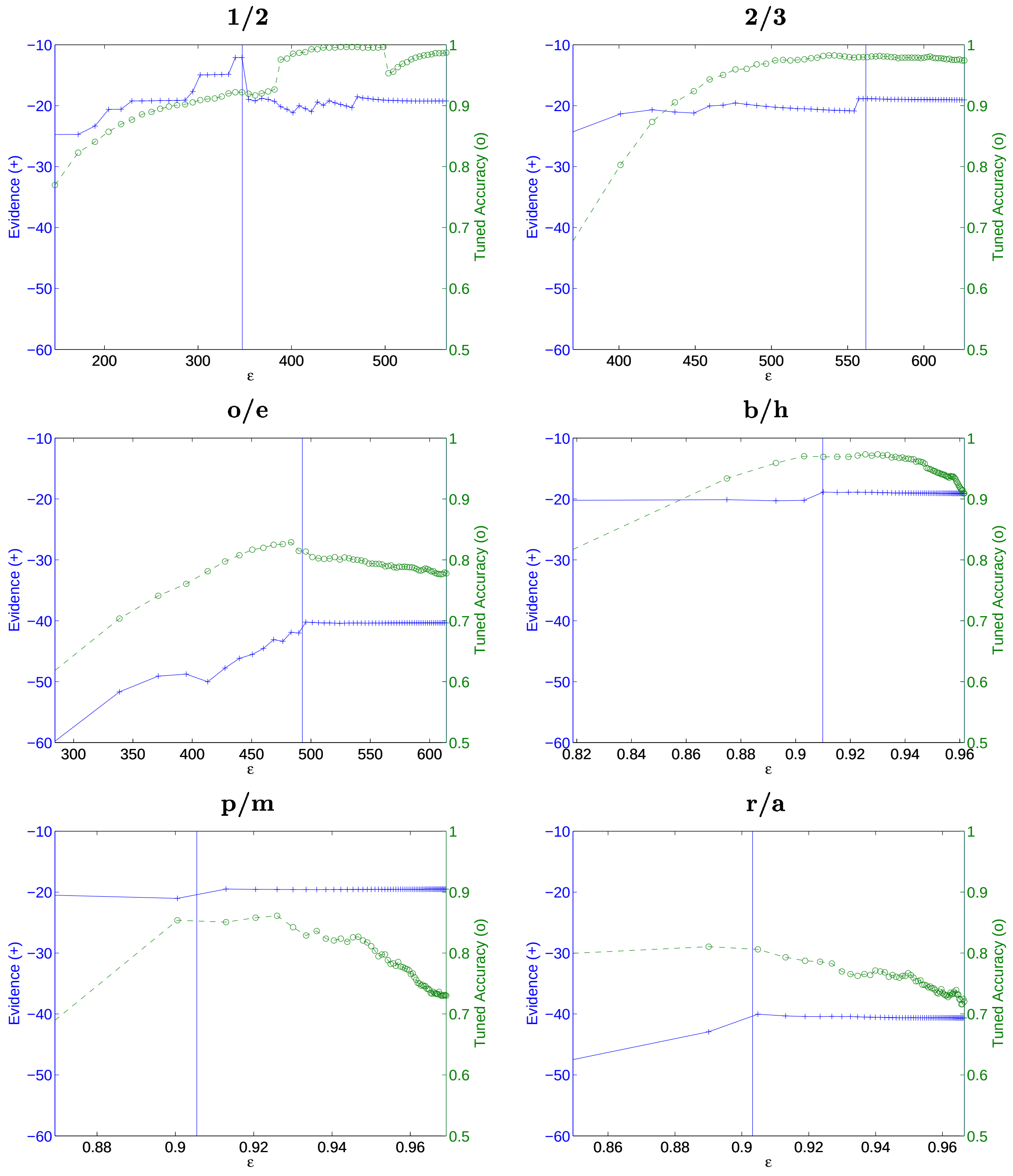

Figure 2: Evidence and tuned accuracy curves of ϵNN unweighted graphs. The vertical lines mark the optimal ϵ.

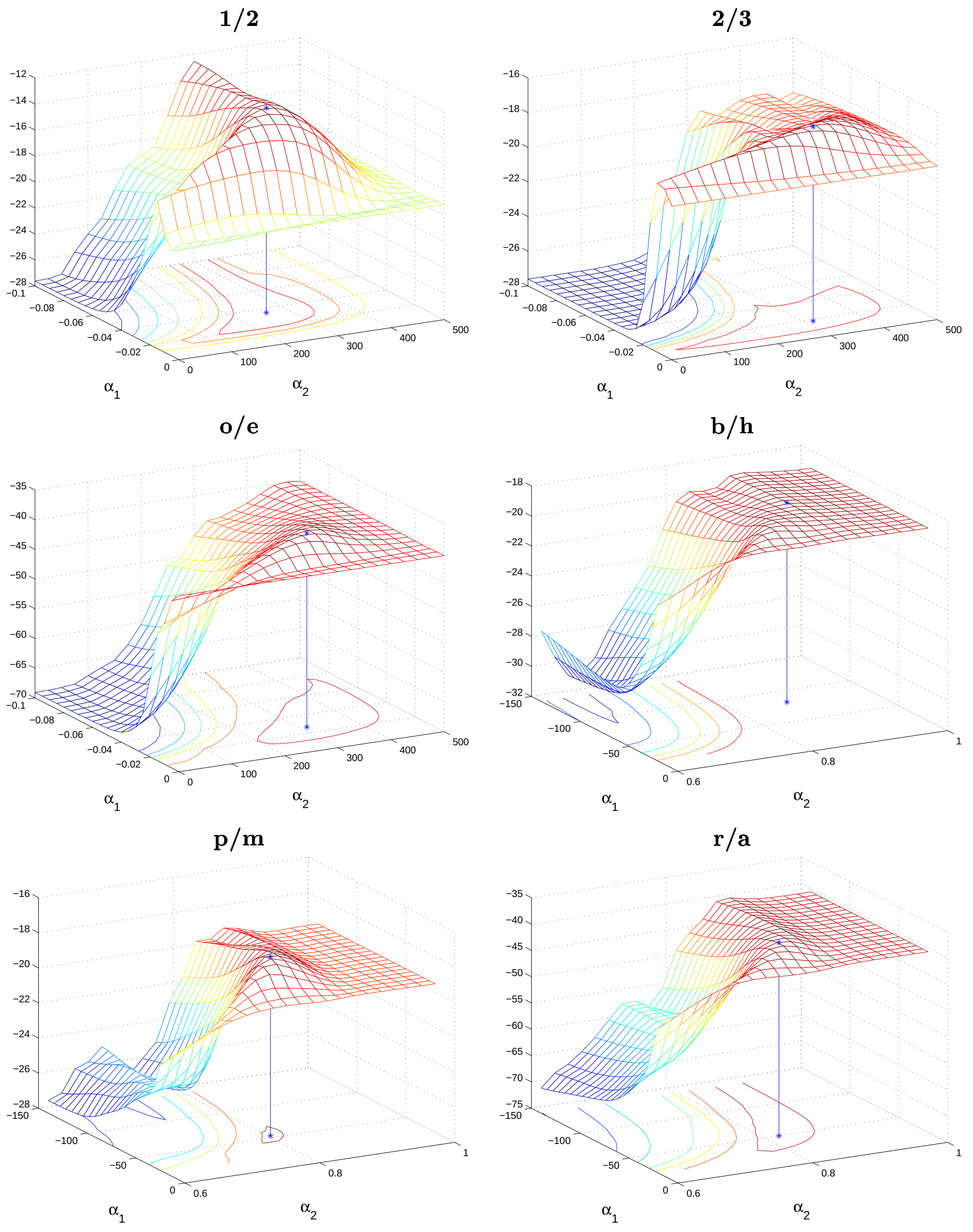

Figure 3: Evidence surface of tanh()-weighted graphs. The vertical lines mark the position of Θ^* found by gradient ascent.

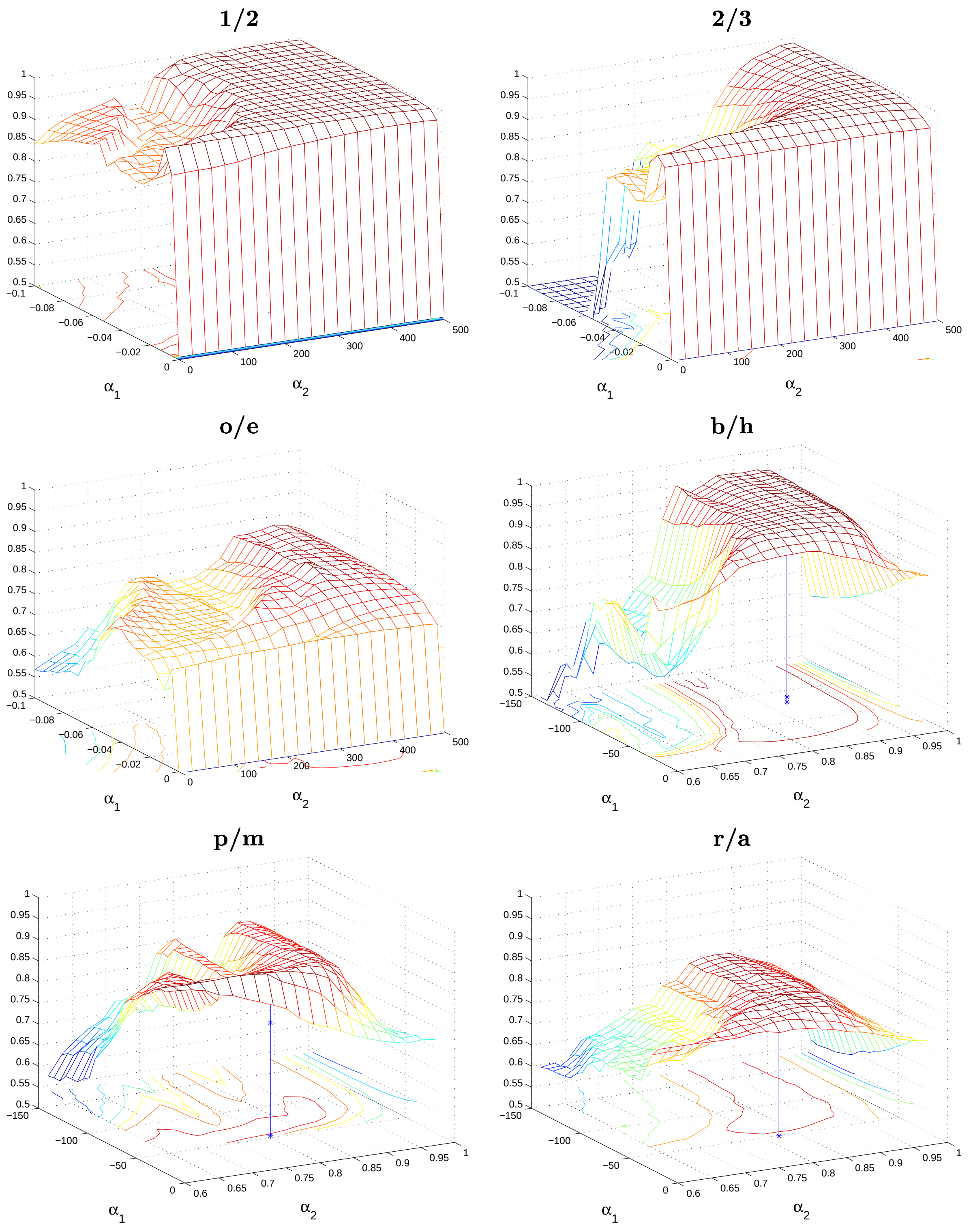

Figure 4: Tuned classification accuracy surface of tanh()-weighted graphs. The vertical lines mark the position of Θ^* found by gradient ascent.

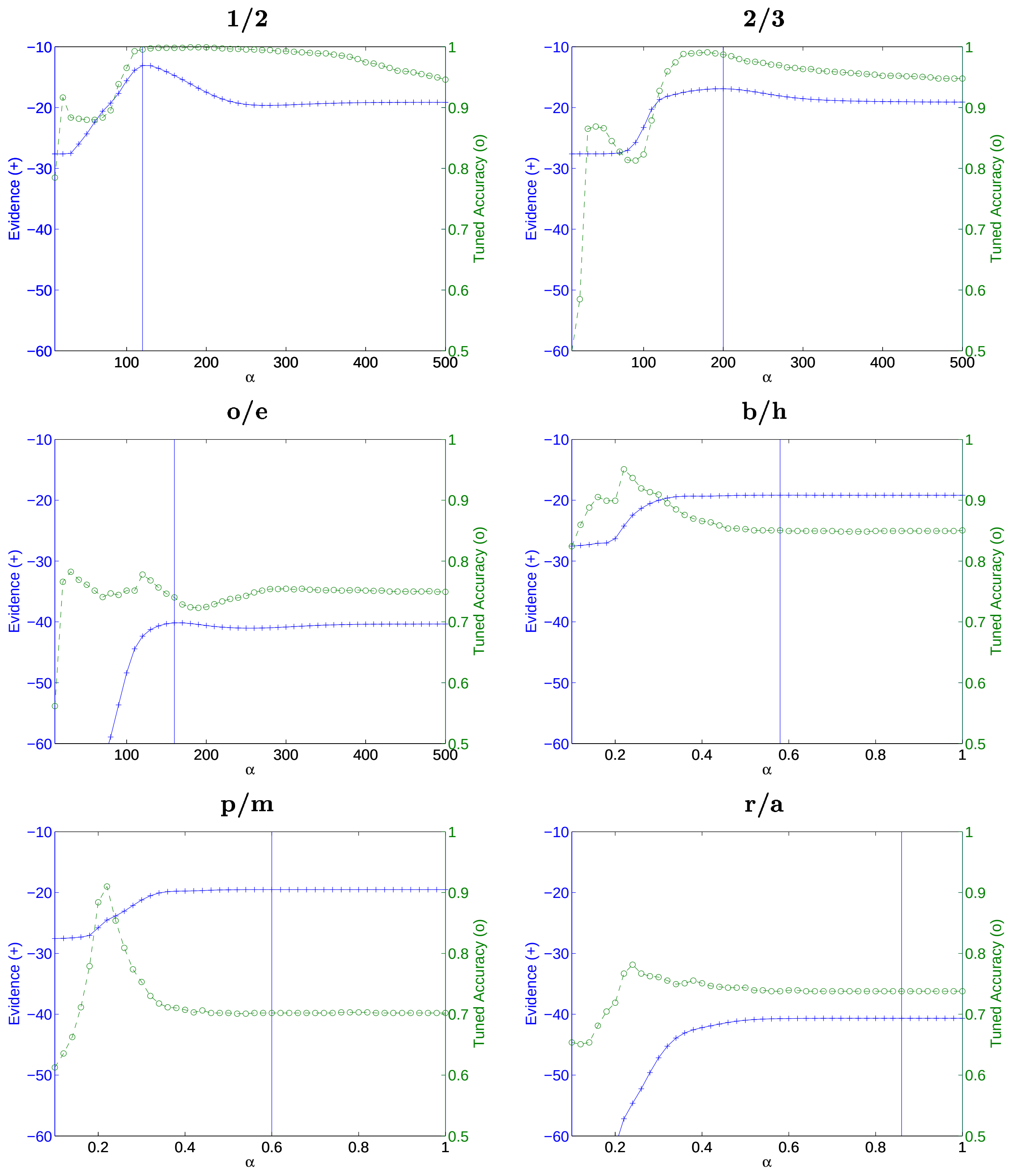

Figure 5: Evidence and tuned accuracy curves of exp() weighted graphs. The vertical lines mark the optimal α.