Lecture 7

IDS575: Statistical Models and Methods
Theja Tulabandhula

Notes derived from the book titled "Elements of Statistical Learning [2nd edition] (Sections 8.5-8.6, 9.1-9.2)

1 Inference using Expectation Maximization

The Expectation Maximization (EM) algorithm is a very useful tool to simplify MLE problems that are difficult to solve.

Lets understand it for a specific problem first: the problem of density estimation.

Density estimation is the problem of estimating the distribution function (density function) using a dataset.

Consider the 1-dimensional data ($N = 20$) shown in Figure 1 and enumerated in Figure 2.

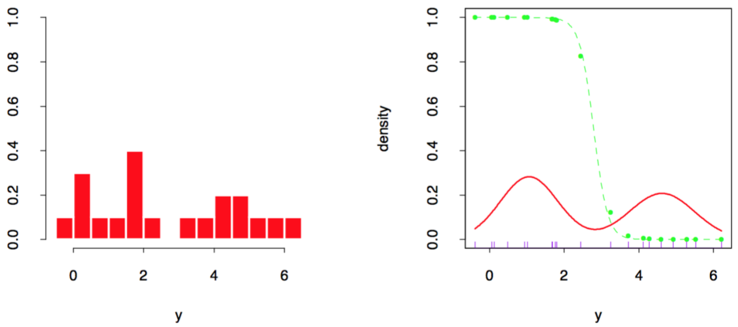
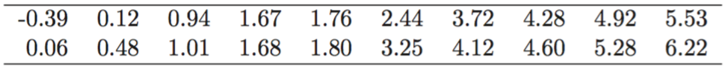


Figure 1: Dataset of 20 1-dimensional points, drawn as a histogram on the left. MLE fit on the right (solid red, more explanation in text).

Eyeballing the data, it looks like there is *bi-modality*, so we hypothesize that the data came from a *mixture* of two Gaussian distributions in the following way:

1. $Y_1 \sim N(\mu_1, \sigma_1^2)$



-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22

Figure 2: Data used for the histogram in Figure 1.

2. $Y_2 \sim N(\mu_1, \sigma_1^2)$

3. $\Delta \sim \text{Bernoulli}(\pi)$

4. $Y = (1 - \Delta)Y_1 + \Delta Y_2$.

The above is called a *generative* description. We flip a coin with bias π and then depending on the outcome, make Y equal to Y_1 or Y_2. This is a very popular model in statistics and machine learning.

The density of Y is $g_Y = (1 - \pi)N(Y; \mu_1, \sigma_1^2) + \pi N(Y; \mu_2, \sigma_2^2)$ and the parameter vector is $\theta = [\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2]^T$.

The log likelihood is

$$l(\theta; \mathbf{Y}) = \sum_{i=1}^{N} \log[(1 - \pi)N(y_i; \mu_1, \sigma_1^2) + \pi N(y_i; \mu_2, \sigma_2^2)].$$

This optimization problem is difficult because there is an additive term inside the log. It turns out that there are multiple *local maxima* for this function[1].

Lets introduce the *unobserved/latent* variables δ_i (realizations of $\Delta \sim \text{Bernoulli}(\pi)$ taking values 0 and 1 resp.). Then the new log-likelihood would be

$$l_0(\theta; \mathbf{Y}, \mathbf{\Delta}) = \sum_{i=1}^{N} \left\{(1 - \delta_i)\left(\log N(y_i; \mu_1, \sigma_1^2) + \log(1 - \pi)\right) + \delta_i \left(\log N(y_i; \mu_2, \sigma_2^2) + \log \pi\right)\right\}.$$

Maximizing the above likelihood is easy because the terms involving one parameter can be decoupled with terms involving other parameters.

Since δ_i are not observed, we could substitute their conditional expected values, which are defined as:

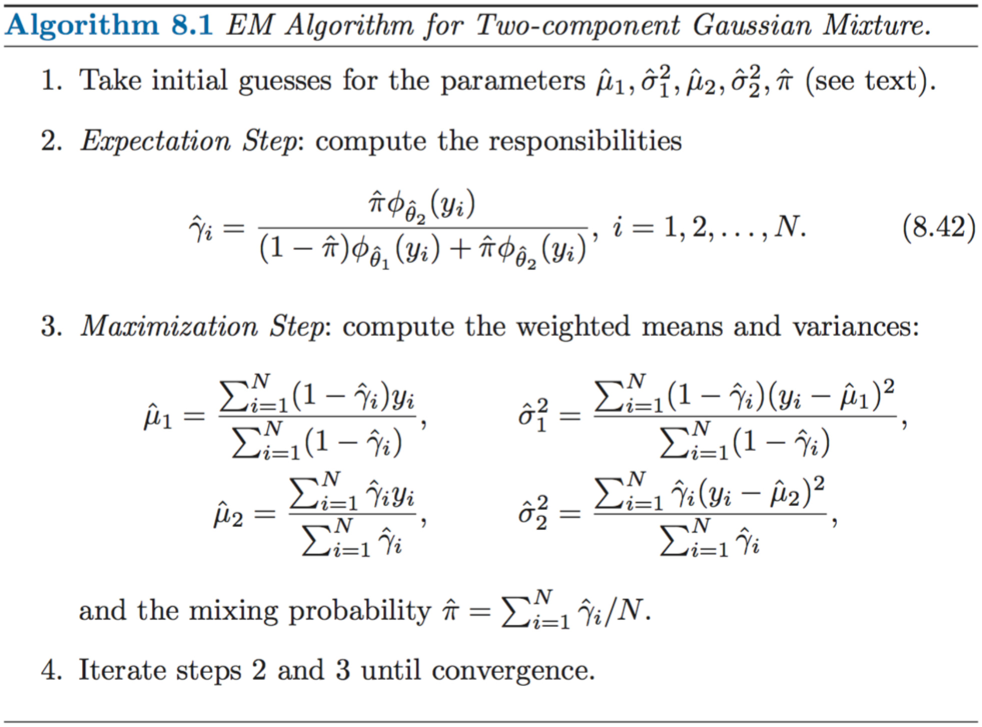
$$\gamma_i = E[\Delta_i | \theta, \mathbf{Y}],$$

giving us the idea for an iterative algorithm shown in Figure 3.

In each iteration, there are two steps:

- *The expectation step* (E): We find γ_i, a soft assignment of each y_i to one of the two Gaussians.

[1] A point θ is a local maximum if it maximizes the value of the given function in its neighborhood. It is a global maximum if it maximizes the function across all possible points.



Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \ldots, N. \qquad (8.42)$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.

Figure 3: The Expectation-Maximization algorithm for the 2-component mixture model.

- *The maximization step* (M): Substituting these γ_i in place of δ_i and maximize $l_0(\theta; \mathbf{Y}, \boldsymbol{\Delta})$.

Example 1. For the 1-dimensional data above, starting with several initial values for θ, the EM algorithm was run. Figure 4 shows one of these runs. Figure 5 shows how the estimate $\widehat{\pi}$ changes over iterations. Finally, the right panel of Figure 1 shows the fitted Gaussians.

1.1 The General Trick

The idea behind EM is to make the likelihood maximization problem easier by enlarging the data with latent data. This latent data can be any data that was missing, corrupted, or just unobserved. The general algorithm is shown in Figure 6. Let the observed data be $\mathbf{Z}$ and the latent data be $\mathbf{Z}^m$ and let $\mathbf{T} = (\mathbf{Z}, \mathbf{Z}^m)$. Let the original and new log-likelihoods be $l(\theta; \mathbf{Z})$ and $l_0(\theta; \mathbf{T})$. Then, in the j^{th} iteration, we do:

- The E step: compute $Q(\theta', \widehat{\theta}(j))$ by computing the conditional distribution of the latent variables given the observed and the current parameter $\widehat{\theta}(j)$ and substituting these in the expression for $l_0(\theta'; \mathbf{T})$ (call it $Q(\theta', \widehat{\theta}(j))$).

- The M step: Maximize the expression $Q(\theta', \widehat{\theta}(j))$ over θ' to get $\widehat{\theta}(j+1)$. And repeat.

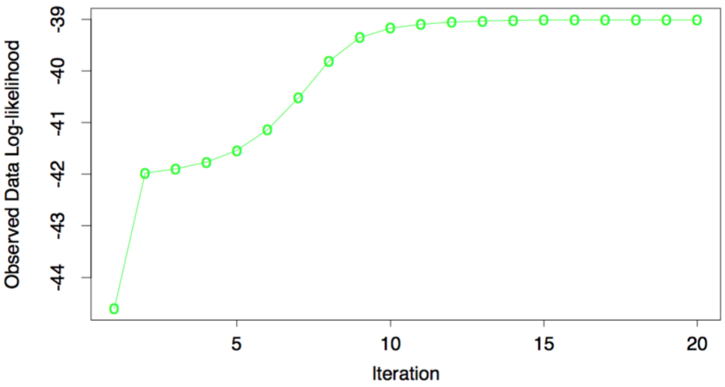
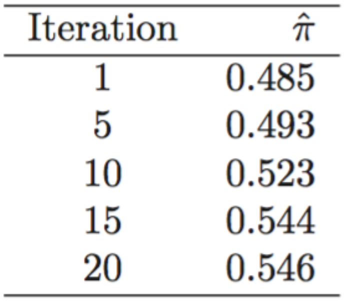


Figure 4: Progress of the Expectation-Maximization algorithm for the 1-dimensional dataset.



Iteration	$\hat{\pi}$
1	0.485
5	0.493
10	0.523
15	0.544
20	0.546

Figure 5: Estimates of π across select iterations of the Expectation-Maximization algorithm for the 1-dimensional dataset.

1.2 Why does EM Work?

Lets look at the following conditional distribution relation, that related the two likelihoods:

$$\begin{aligned} Pr(\mathbf{Z}\,|\theta') &= \frac{Pr(\mathbf{T}\,|\theta')}{Pr(\mathbf{Z}^m\,|\,\mathbf{Z},\theta')} \\ \Rightarrow l(\theta';\mathbf{Z}) &= l_0(\theta';\mathbf{T}) - l_1(\theta';\mathbf{Z}^m\,|\,\mathbf{Z}). \end{aligned}$$

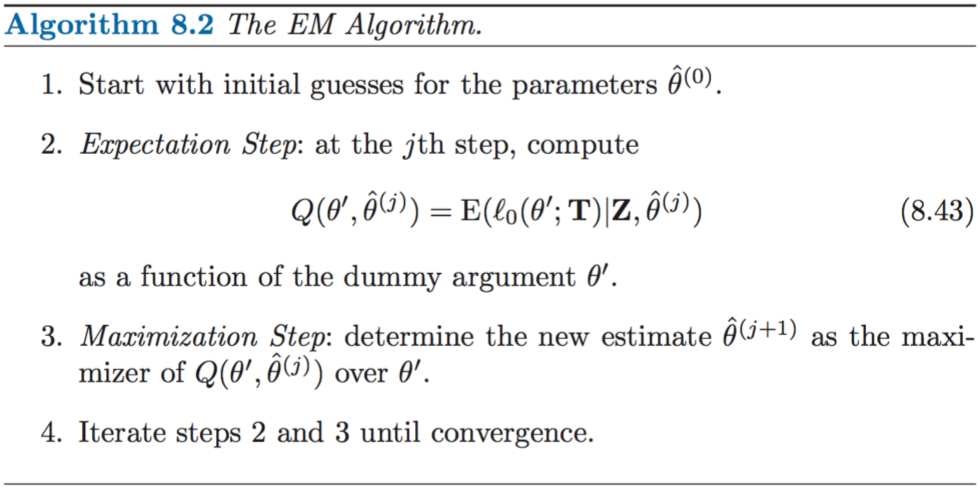
Take conditional expectation of the above equality with respect to $Pr(\mathbf{T}\,|\,\mathbf{Z},\theta)$, we get:

$$\begin{aligned} l(\theta';\mathbf{Z}) &= E[l_0(\theta';\mathbf{T})|\,\mathbf{Z},\theta] - E[l_1(\theta';\mathbf{Z}^m\,|\,\mathbf{Z})|\,\mathbf{Z},\theta] \\ &= Q(\theta',\theta) - R(\theta',\theta). \end{aligned}$$

So in the M step, we maximize the first term above with respect to θ'. But we still succeed in maximizing $l(\theta';\mathbf{Z})$, which is our original log-likelihood.

This is because the second term is maximized[2] when $\theta = \theta'$. What does this imply, lets look at two different parameter values, θ' and θ, where θ' maximizes $Q(\theta',\theta)$. Then, the

[2] We will assume that this is true here. You can verify this claim yourself.



Algorithm 8.2 *The EM Algorithm.*

1. Start with initial guesses for the parameters $\hat{\theta}^{(0)}$.
2. *Expectation Step*: at the jth step, compute
$$Q(\theta', \hat{\theta}^{(j)}) = \mathrm{E}(\ell_0(\theta'; \mathbf{T})|\mathbf{Z}, \hat{\theta}^{(j)}) \tag{8.43}$$
as a function of the dummy argument θ'.
3. *Maximization Step*: determine the new estimate $\hat{\theta}^{(j+1)}$ as the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over θ'.
4. Iterate steps 2 and 3 until convergence.

Figure 6: The Expectation-Maximization algorithm in general.

difference in the original log-likelihood will be:

$$\begin{aligned} l(\theta'; \mathbf{Z}) - l(\theta; \mathbf{Z}) &= Q(\theta', \theta) - R(\theta', \theta) - (Q(\theta, \theta) - R(\theta, \theta)) \\ &= (Q(\theta', \theta) - Q(\theta, \theta)) - (R(\theta', \theta) - R(\theta, \theta)) \\ &\geq 0. \end{aligned}$$

The above inequality just means that we are always improving our original objective values in each iteration. Hence, EM will at least get to the local maxima.

In summary, EM is a very powerful technique to fit complex models to many different type of data: from fitting mixture models to community detection in social network analysis. for every application, the E step and the M steps may need to be customized.

2 Sampling from the Posterior

Lets go back to Bayes rule. if you remember MAP estimation, it includes the prior on the parameter θ and computes the mode of the posterior distribution over θ given data.

Here we discuss ways to compute the mode, or other functions of the posterior via the technique of *sampling*.

Example 2. Say you have a random variable Z. Then one way to estimate $E[Z]$ is to sample say N realizations and take a *Monte Carlo average* $\frac{1}{N}\sum_{i=1}^{N} z_i$. Of course, $E[Z]$ can be computed analytically given the distribution/density if the latter is a well known distribution/density like the Multinomial or the Gaussian.

Sampling from certain distributions is easy, especially if you have access to samples from the uniform distribution $U[0, 1]$.

Example 3. Say we want to sample from a continuous increasing distribution function F which is defined as $F(z) = Pr(Z \leq z)$. Let F^{-1} be its inverse function. Then, Z can be sampled in two steps:

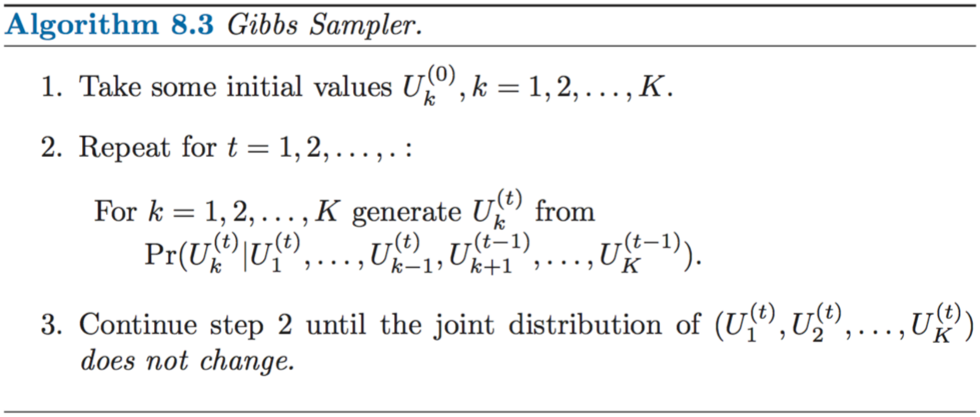
- Sample U from $U[0, 1]$.
- Return $Z = F^{-1}(U)$.

For example, $Z = -\frac{1}{\lambda}\log(U)$ is exponentially distributed with parameter λ.

If we wish to sample from a more complex distribution, say that of a p-dimensional random vector $U = [U_1, ..., U_p]^T$, there are a family of techniques called Markov Chain Monte Carlo, one of which we will discuss: the Gibbs Sampler.

The key assumption for the Gibbs sampler is that it is easy to sample from $P(U_j|U_1, ..., U_{j-1}, U_{j+1}, ..., U_p)$ for every $j = 1, ..., p$.

The key idea of Gibbs sampling is to sample from each of the conditional distributions in some sequence, many times, and finally bundle p of them together to get a sample. This is illustrated in Figure 7. It turns out that this repeated sampling is related to an object called the *Markov chain*. And the relation is this: when we are sampling coordinate U_j, it depends on $p-1$ samples that we obtained previously and everything before that is irrelevant (this is called the Markov property). Gibbs sampler stabilizes after some time and actually produces realizations u that are from the distribution $P(U_1, ..., U_p)$ [3].



Algorithm 8.3 *Gibbs Sampler.*

```
1. Take some initial values $U_k^{(0)}, k = 1, 2, \ldots, K$.
2. Repeat for $t = 1, 2, \ldots, .$ :
      For $k = 1, 2, \ldots, K$ generate $U_k^{(t)}$ from
         $\Pr(U_k^{(t)}|U_1^{(t)}, \ldots, U_{k-1}^{(t)}, U_{k+1}^{(t-1)}, \ldots, U_K^{(t-1)})$.
3. Continue step 2 until the joint distribution of $(U_1^{(t)}, U_2^{(t)}, \ldots, U_K^{(t)})$
   does not change.
```

Figure 7: The Gibbs Sampler.

Note 1. There are many other sampling techniques, and Gibbs sampler is just one of them. They are very relevant in Bayesian statistical modeling.

[3] One can verify this claim, although we will skip this here.

In the next several sections/lectures, we will get back to supervised learning. In particular, we will look at classification and regression methods that work with different (sometimes lesser) assumptions than linearity, but still give great predictive performance. These are:

1. Generalized Additive Models (GAMs),
2. *(future)* Tree-based methods,
3. *(future)* Multivariate Adaptive Regression Splines (MARS),
4. *(future)* Adaboost and Gradient Boosting Methods,
5. *(future)* Random Forest, and
6. *(future)* Support Vector Machines.

3 Generalized Additive Models

In many situations, Y does not depend linearly on X. To counter this, a GAM assumes the following:

$$E[Y|X] = \alpha + f_1(X_1) + f_2(X_2) + ... + f_p(X_p),$$

where f_j can be specified arbitrarily. For example, each such function can be a cubic smoothing spline[4].

Example 4. The GAM version of logistic regression for two classes would be:

$$\log(\frac{P(G=1|X)}{1-P(G=1|X)}) = \alpha + f_1(X_1) + f_2(X_2) + ... + f_p(X_p).$$

It turns out that $P(G=1|X) = \mu(X)$ is the conditional mean of response Y. In general, this mean is related to the right hand side above via a *link* function. Example link functions include:

- Identity link: $g(\mu) = \mu$.
- Logit link: $g(\mu) = \log(\frac{\mu}{1-\mu})$.
- Probit link: $g(\mu) = \Phi^{-1}(\mu)$.
- log-linear link: $g(\mu) = \log(\mu)$.

[4] Such a function is the minimizer (over the class of twice differentiable functions) of $\sum^N (y_i - f(x_i))^2 + \lambda \int f^{''}(x)^2 dx$.

These (and other links) define the family of Generalized Linear Models (GLMs) when $f_j(X_j) = X_j$.

Note 2. We can have nonlinear functions of more than one X_j as well.

The way these functions are estimated from data, is by solving the following optimization problem:

$$PRSS(\alpha, f_1, ..., f_p) = \sum_{i=1}^{N}(y - \alpha - \sum^{p} f_j(x_i j))^2 + \sum_{j=1}^{p} \lambda_j \int f_j^{''}(t_j)^2 dt_j,$$

where x_{ij} is the j^{th} component of the i^{th} observation. It turns out that this is not very difficult, although we will skip it to focus on other models.

Example 5. An example set of functions for a spam classification task is shown in Figure 8.

Note 3. Unequal Loss Functions: In a spam classification task, it may be more important to not classify a genuine email as spam, compared to classifying a spam as a genuine email. In order to do so, the $0-1$ loss function can be changed to penalize one type of mis-classification more than the other.

4 Summary

We learned the following things:

- Learning/inference in general settings can be achieved using maximum likelihood and the expectation maximization (EM) methods.
- Sampling as a way to infer/estimate quantities of interest, especially when the probability distribution function is complex.
- More models for regression and classification including: Generalized Additive Models and Tree-based methods.

A Sample Exam Questions

1. When is the EM algorithm useful? How is it different from MLE?
2. In what situations would sampling be useful?
3. What are the key differences between General Additive Models and linear models? Which family is more flexible?

B Sensitivity and Specificity

In certain 2-class classification applications (say, disease (1) and no-disease (0)), one is not only interested in a single mis-classification metric, but in two additional related metrics:

1. Sensitivity: It is the probability of the model predicting disease when the true label is disease.

2. Specificity: It is the probability of the model predicting no-disease when the true label is no-disease.

These can be empirically computed over training/validation data just like any other performance measure. By introducing unequal loss values (say $L_{kk'} \neq L_{k'k}$; here $L_{kk'}$ is the loss for predicting true label k as label k') we can change these numbers. For example, if we want specificity to be high, then we can set L_{01} high.

The receiver operating characteristic curve (ROC) is used to shown this trade-off between sensitivity and specificity. It plots sensitivity vs specificity.

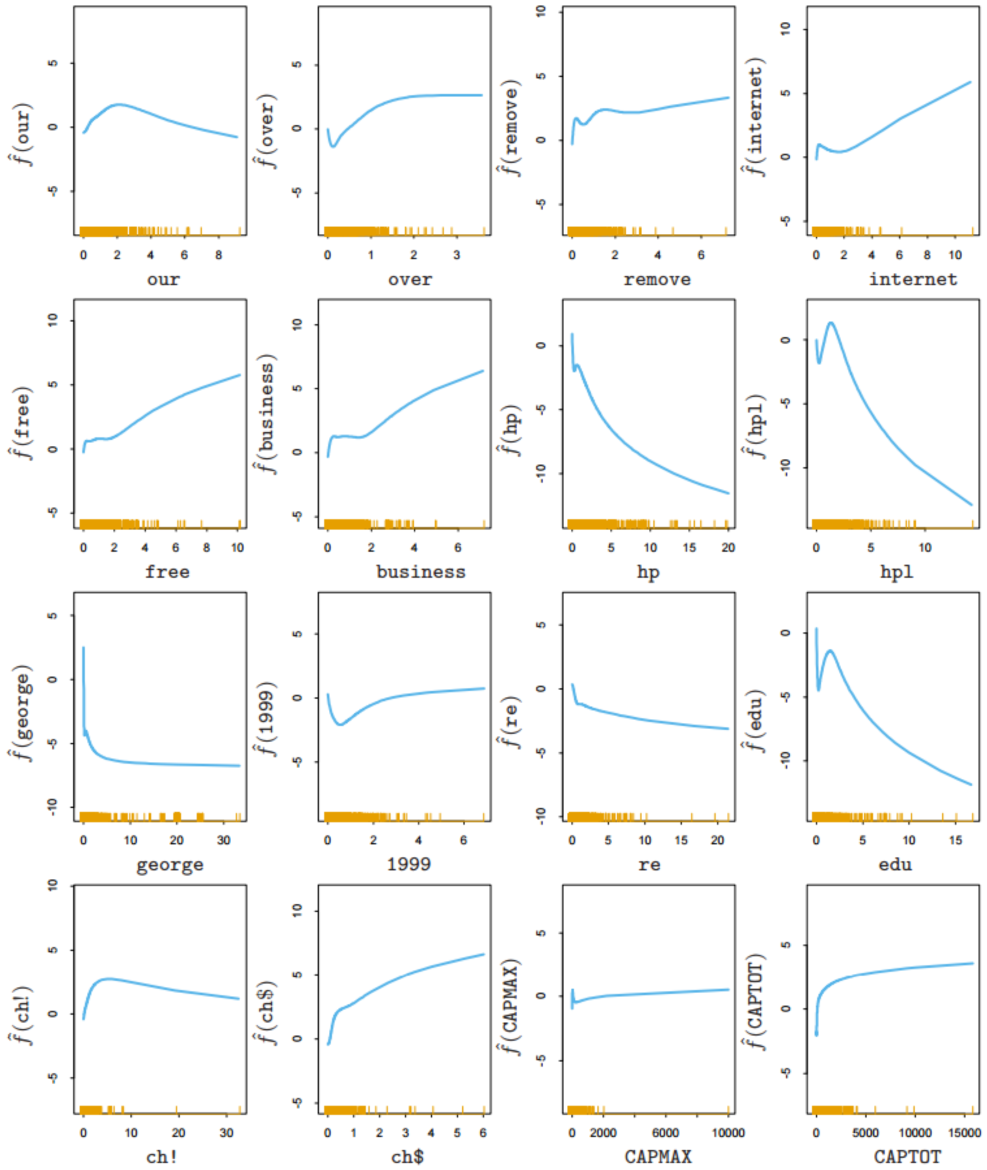


Figure 8: Estimated functions for a GAM model for a spam classification dataset.