9.520: Statistical Learning Theory and Applications March 10th, 2010

Regularization Methods for Multi-Output Learning

Lecturer: Lorenzo Rosasco *Scribe: Yi-Chieh Wu, Phillip Isola*

1 Introduction

Today we discuss the problem of *multi-output learning*, also known as multi-task learning (MTL), in which, as the name implies, the object of interest is modeled as a function with multiple outputs. We can view this problem from two perspectives: either we have a function with multiple outputs, or we have to learn multiple, correlated functions, each with a single output.

2 Examples

We start by providing a few practical examples in which multi-output learning might be useful.

- Suppose we are attempting to model the buying preferences of several consumers based on past purchases, e.g. as in the Netflix recommender system. We assume that people with similar tastes tend to buy similar items and their buying history is related. Inferring the preferences for a customer based only on his past purchases may be tough, because that customer may not have rated enough movies or made enough purchases. If we can add information from other customers with similar tastes, then we can essentially increase the number of data samples and hopefully also increase classification accuracy, thereby prompting customers to rent movies attuned to their tastes and be more satisfied with the overall Netflix service. In this case, each consumer is modeled as a task and their previous preferences are the corresponding training set.
- In pharmocological studies, we may be attempting to predict the blood concentration of a medicine at different times across multiple patients. Finding the best-fit function for a single patient based only on his measurements will likely overfit the training data. Instead, if we pool the measurements across the patients, we can find a time series function that will likely better generalize to the population at large.
- The lecture slides provide a much wider array of practical applications, in fields such as financial analysis, computer vision, and geophysics.

3 Framework

The goal of multi-output problems can be thought of as improving classification performance by exploiting relationships among the different tasks. Some data sets may be very small while others are large, and some may be more similar to others, but by combining all the information together and training the classifiers simultaneously, we can leverage shared information among the data sets (see Figure 1).

The framework for multi-task learning can be formally described as follows.

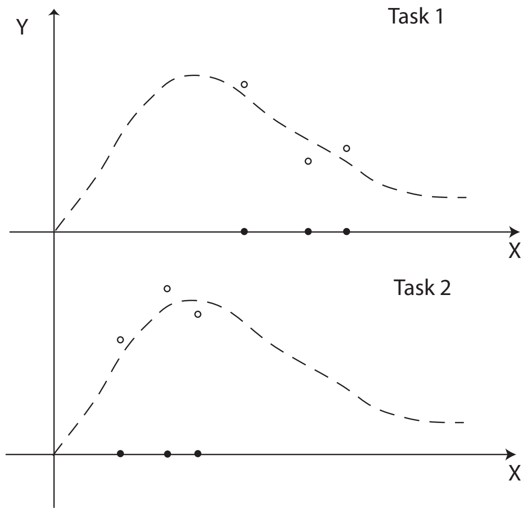

Figure 1: If we have samples from multiple tasks, all of which have a similar underlying functional form, we can exploit this similarity by somehow **borrowing strength** across the tasks during learning.

There are T single-output tasks. For each task $j = 1, \dots, T$, we are given a set of examples

$$S_j = (x_i^j, y_i^j)_{i=1}^{n_j}$$

sampled i.i.d. according to a distribution P_t. The goal is to find

$$f^t(x) \sim y, \qquad \text{for each } t = 1, \dots, T.$$

With the above notation, we usually let $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ so that x_i^j (the i^{th} vector in the j^{th} training set) is in $\mathbb{R}^d$. Now $f^t(x)$ is a single classification rule.

Before moving on, we note that the framework of multi-task learning makes the assumption that the multiple tasks are "similar". In the strongest case, we might assume the examples are all sampled from the same distribution P. This assumption might be valid in many real world problems, such as in many time series prediction problems, in which the tasks are very correlated. However, in designing MTL algorithms, it remains important to assess exactly what kinds of similarities across tasks that we can describe.

On the other hand, the framework for multi-task learning problems can be very general. That is, the input spaces can be different, the output spaces can be different, and/or the hypothesis spaces can be different. In this lecture, we will simplify the problem so that the input, output, and hypothesis spaces are the same (e.g. $X_j = X$, $Y_j = Y$, and $\mathcal{H}_j = \mathcal{H}$ for all $j = 1, \dots, T$), and furthermore, the hypothesis space is a RKHS with kernel K. This makes the problem easier to analyze, and it turns out that many results transfer directly to more general cases.

4 Algorithm

A possible way of solving the MTL problem is to minimize the penalized empirical risk, e.g.

$$\min_{f^1,\ldots,f^T} \ \mathrm{ERR}[f^1,\ldots,f^T] + \lambda\,\mathrm{PEN}(f^1,\ldots,f^T),$$

where typically, the error term is the sum of the empirical risks and the penalty term enforces similarity among the tasks.

4.1 Empirical Risk

We let

$$\mathrm{ERR}[f^1,\ldots,f^T] = \sum_{j=1}^{T} l_{S_j}[f^j]$$

where

$$l_{S_j}[f^j] = \frac{1}{n_j}\sum_{i=1}^{n_j}(y_i^j - f^j(x_i^j))^2$$

for the square loss. Note that taking the sum means giving equal importance to all data sets so that we are in some sense averaging the information from our data sets.

4.2 Penalty Term

The default choice for the regularizer in a single task classifier is $\|f\|_{\mathcal{H}}^2$. An easy choice (but perhaps too simple) for multi-output learning is to use summation for the penalty term as we did for the empirical risk, e.g. $\mathrm{PEN}(f^1,\ldots,f^T) = \sum_{j=1}^{T}\|f^j\|_K^2$. However, with this penalty term we will just end up finding the minimizing $f^1,\ldots,f^T$ for all the tasks independently. Instead, we would like to make use of some notion of correlation among the functions. We will do this by breaking up the penalty term into two components: one component acting as the usual regularizer in individual tasks, and the other component imposing similarity between tasks. The relative weighting of these two components will be controlled by two parameters, γ and λ. When $\gamma = 0$, we treat everybody as independent, and when $\lambda = 0$, we treat everybody as very close to the mean. We will consider three different regularizers:

- *Mixed Effect Regularizers.* Let us assume that the functions look like a common curve plus a (possibly task-specific) perturbation, and further assume that the common curve is the average. Then[1]

  $$\mathrm{PEN}(f^1,\ldots,f^T) = \lambda\sum_{j=1}^{T}\|f^j\|_K^2 + \gamma\sum_{j=1}^{T}\left\|f^j - \frac{1}{T}\sum_{s=1}^{T}f^s\right\|_K^2.$$

  We can further extend this concept if not all tasks are the same, but maybe they are divided into groups (see *Cluster Regularizers* below).

- *Graph Regularization Regularizers.* Sometimes we may know more specific correlations between the tasks. In graph regularization, we build a similarity graph that encodes these correlations. This graph can be represented as a $T\times T$ positive weight matrix M, which we can use to enforce stronger or weaker similarities specific to each pair of tasks.

  $$\mathrm{PEN}(f^1,\ldots,f^T) = \gamma\sum_{\ell,q=1}^{T}\|f^\ell - f^q\|_K^2 M_{\ell q} + \lambda\sum_{\ell=1}^{T}\|f^\ell\|_K^2 M_{\ell\ell}$$

[1]Note the $\frac{1}{T}$ factor in the average that is missing in the lecture slides.

- *Cluster Regularizers.* The components/tasks are partitioned into c clusters so that the components in the same cluster should be similar. For $r = 1, \dots, c$, let
  - m_r be the cardinality of each cluster, and
  - $I(r)$ be the index set of the components that belong to cluster r.

  Then
$$\text{PEN}(f^1, \dots, f^T) = \gamma \sum_{r=1}^{c} \sum_{l \in I(r)} \|f^l - \overline{f}_r\|_K^2 + \lambda \sum_{r=1}^{c} m_r \|\overline{f}_r\|_K^2,$$
  where $\overline{f}_r$, $r = 1, \dots, c$, is the mean in cluster r.

The above regularizers give us ways of handling MTL when all the tasks are the same, when the tasks are related by a known graph structure, and when the tasks fall into a number of known clusters. Determining which of these situations we are in, and what is the precise similarity structure of the tasks, remains a problem open for discussion. For now, we will assume we are given the correlations between the tasks.

4.3 Tikhonov Regularization

We would like to reframe the problem as Tikhonov regularization, so that we can apply the representer theorem and our standard techniques. To do this, we define a suitable RKHS $\mathcal{H}$ with kernel Q and re-index the sums in the error term (e.g. by stacking the matrices) so that

$$\min_{f_1, \dots, f_T} \Big\{ \sum_{j=1}^{T} \frac{1}{n_j} \Big(\sum_{i=1}^{n} (y_i^j - f^j(x_i))^2 \Big) + \lambda \text{PEN}(f_1, \dots, f_T) \Big\}$$

can be written as

$$\min_{f \in \mathcal{H}} \Big\{ \frac{1}{n_T} \sum_{i=1}^{n_T} (y_i - f(x_i, t_i))^2 + \lambda \|f\|_Q^2 \Big\},$$

where we consider the training set (x_i, y_i, t_i) for $i = 1, \dots, T$ and let $n_T = \sum_{j=1}^{T} n_j$. In the next section, we will show that by letting Q act simultaneously on a point and a task, we can reframe the problem as a scalar task and can apply previously discussed algorithms (e.g. SVM, RLS).

5 Joint Kernels

Remember that a kernel is simply a function that maps $X \times X \to \mathbb{R}$. To let the kernel act simultaneously on a point and a task, let the space be $X = (X, \Pi)$ to produce the (joint) kernel $Q : (X, \Pi) \times (X, \Pi) \to \mathbb{R}$, where $\Pi = \{1, \dots, T\}$ is the index set of the output components. Finally, simply use the reproducing property, where the function can be written as a linear combination of the kernel. Explicitly, in the single task case, we have

$$f(x) = \sum_{i=1}^{p} K(x, x_i) c_i$$

with norm

$$\|f\|_K^2 = \sum_{i,j=1}^{p} K(x_j, x_i) c_i c_j,$$

where $p \leq \infty$. To transition to the multi-task case, set $x = (x, t)$ and $K = Q$ so that

$$f(x,t) = \sum_{i=1}^{p} Q((x,t),(x_i,t_i))c_i$$

with norm

$$\|f\|_Q^2 = \sum_{i,j=1}^{p} Q((x_j,t_j),(x_i,t_i))c_i c_j.$$

A useful class of kernels comes from assuming that we can separate the kernel into a (scalar) kernel for points ($K : X \times X \to \mathbb{R}$) and a kernel for tasks (positive definite matrix A in $T \times T$). Now A acts on the indices of tasks so that $A_{t,t'}$ says how similar tasks t and t' are, giving rise to

$$Q((x,t),(x',t')) = K(x,x')A_{t,t'}$$

with norm

$$\|f\|_Q^2 = \sum_{i,j=1}^{p} K(x_i,x_j)A_{t_i t_j} c_i c_j.$$

Lemma 1 *If we fix t so that $f_t(x) = f(t,x)$ is one of the tasks, we can relate the norm to the scalar product among the tasks as*

$$\|f\|_Q^2 = \sum_{s,t} A^{\dagger}_{s,t}\langle f_s, f_t\rangle_K,$$

where $\dagger$ denotes the pseudo-inverse.

Now we have an inner product among tasks, and a weight matrix A that encodes relations among the outputs.[2] This also implies that

- A regularizer of the form $\sum_{s,t} A^{\dagger}_{s,t}\langle f_s, f_t\rangle_K$ defines a kernel Q. Thus, if we are able to rewrite a regularizer in this form, we will obtain a kernel, Q, that we can used to solve the problem as Tikhonov regularization (or use some other kernel regularization technique). We will take this approach to solve the regularization problems of Section 4.2.
- The norm induced by a kernel Q of the form $K(x,x')A$ can be seen as a regularizer.

Proof of Lemma 1: Note that if $f_t(x) = \sum_i K(x,x_i)A_{t,t_i}c_i$, then

$$\begin{aligned}\langle f_s, f_t\rangle_K &= \Big\langle \sum_i K(x,x_i)A_{s,t_i}c_i, \sum_j K(x,x_j)A_{t,t_j}c_j \Big\rangle \\ &= \sum_{i,j} \big\langle K(x,x_i), K(x,x_j)\big\rangle A_{s,t_i}A_{t,t_j}c_i c_j \\ &= \sum_{i,j} K(x_i,x_j)A_{s,t_i}A_{t,t_j}c_i c_j.\end{aligned}$$

Now multiply by $A^{-1}_{s,t}$ (or rather $A^{\dagger}_{s,t}$) and sum over s,t to arrive at

$$\begin{aligned}\sum_{s,t} A^{\dagger}_{s,t}\langle f_s, f_t\rangle_K &= \sum_{s,t}\sum_{i,j} K(x_i,x_j)A^{\dagger}_{s,t}A_{s,t_i}A_{t,t_j}c_i c_j \\ &= \sum_{i,j} K(x_i,x_j)\sum_{s,t}[A^{\dagger}_{s,t}A_{s,t_i}A_{t,t_j}]c_i c_j \\ &= \sum_{i,j} K(x_i,x_j)A_{t_i t_j}c_i c_j,\end{aligned}$$

which is exactly equal to $\|f\|_Q^2$ previously determined. $\square$

[2]In other words, A shows the correlation among the tasks. For example, think of $f = (f^1, ..., f^T)^\top$ as a vector of functions; then A is a weight matrix on the correlation of these functions.

5.1 Penalty Term

Now we relate the joint kernel to the three different regularizers of Section 4.2.

- *Mixed Effect Regularizers.* Let $\mathbf{1}$ be the $T \times T$ matrix whose entries are all equal to 1, and let $\mathbf{I}$ be the T-dimensional identity matrix. Set $A = \omega\mathbf{1} + (1-\omega)\mathbf{I}$ so that the functions f_s and f_t have correlation 1 when $s = t$ and correlation ω when $s \neq t$. (If $\omega = 1$, all tasks are similar, and if $\omega = 0$, all tasks are independent.) Then the kernel

$$Q((x,t),(x',t')) = K(x,x')(\omega\mathbf{1} + (1-\omega)\mathbf{I})_{t,t'}$$

  incurs a penalty

$$A_\omega \left(B_\omega \sum_{\ell=1}^{T} \|f^\ell\|_K^2 + \omega T \sum_{\ell=1}^{T} \left\| f^\ell - \frac{1}{T}\sum_{q=1}^{T} f^q \right\|_K^2 \right),$$

  where $A_\omega = \frac{1}{2(1-\omega)(1-\omega+\omega T)}$ and $B_\omega = (2 - 2\omega + \omega T)$.

- *Graph Regularization Regularizers.* The penalty

$$\frac{1}{2}\sum_{\ell,q=1}^{T} \|f^\ell - f^q\|_K^2 M_{\ell q} + \sum_{\ell=1}^{T} \|f^\ell\|_K^2 M_{\ell\ell}$$

  can be rewritten as

$$\sum_{\ell,q=1}^{T} \langle f^\ell, f^q\rangle_K L_{\ell q},$$

  where $L = D - M$, with $D_{\ell q} = \delta_{\ell q}(\sum_{h=1}^{T} M_{\ell h} + M_{\ell q})$. The kernel is $Q((x,t),(x',t')) = K(x,x')L^\dagger_{t,t'}$.

- *Cluster Regularizers.* The penalty

$$\epsilon_1 \sum_{c=1}^{r} \sum_{l \in I(c)} \|f^l - \overline{f}_c\|_K^2 + \epsilon_2 \sum_{c=1}^{r} m_c \|\overline{f}_c\|_K^2$$

  induces a kernel $Q((x,t),(x',t')) = K(x,x')G^\dagger_{t,t'}$ with

$$G_{lq} = \epsilon_1 \delta_{lq} + (\epsilon_2 - \epsilon_1)M_{lq}.$$

  The $T \times T$ matrix M is such that $M_{lq} = \frac{1}{m_c}$ if components l and q belong to the same cluster c with cardinality m_c, and $M_{lq} = 0$ otherwise.

5.2 Wrapup – Applying the Representer Theorem and Solving

In the previous section, we showed that if given a penalty, we can expand it to an inner product and a number, put the numbers into a matrix, and invert the matrix. Thus, if we know how to define a reasonable regularizer, or if we have a similarity matrix, the MTL problem becomes "easy" to frame in Tikhonov regularization.

The representor theorem tells us that

$$f(x,t) = f_t(x) = \sum_{i=1}^{n} Q((x,t),(x_i,t_i))c_i,$$

where the coefficients are given by

$$(\mathbf{Q} + \lambda I)\mathbf{C} = \mathbf{Y}.$$

and $\mathbf{C} = (c_1, \ldots, c_n)^T$, $\mathbf{Q}_{ij} = Q((x_i, t_i), (x_j, t_j))$ and $\mathbf{Y} = (y_1, \ldots, y_n)^T$. Note that even under the simplifying assumption that the matrices can just be concatenated, we still have a bottleneck in inverting the matrix $\mathbf{Q}$. Luckily, we can use L_2 boosting and find the coefficients by performing gradient descent on the empirical risk

$$\frac{1}{n_T} \|\mathbf{Y} - \mathbf{QC}\|^2_{n_T},$$

i.e. we set $\mathbf{C}^0 = 0$ and consider for $i = 1, \ldots, t-1$,

$$\mathbf{C}^i = \mathbf{C}^{i-1} + \eta(\mathbf{Y} - \mathbf{QC}^{i-1}),$$

where η is the step size (and the factors of 2, n_T, and $\mathbf{Q}$ are folded into the step size). The number of iterations plays the role of regularization parameter in the gradient descent.

6 Related Topics

6.1 Vector Fields

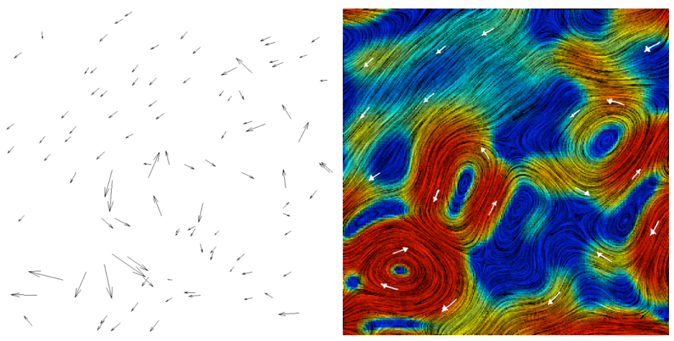

Figure 2: Example vector field data.
(figures from Macêdo and Castro 08)

Finally, we will discuss briefly how we can incorporate vector fields into MTL problems.[3] This is in some sense the most natural extension of the scalar setting, as we simply let y be a vector rather than a scalar. That is, we are now given a set of training points $S = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}^T$, and the points are sampled i.i.d. according to some probability distribution P. The goal is to find $f(x) \sim y$.

As a practical example, suppose that we sample the velocity fields of an incompressible fluid and want to recover the whole velocity field. Let the input x be a location, and the output be the velocity components at that location, $y = (v_1, v_2)$. This formulation allows us to average out noise or enforce other constraints, e.g. that the divergence of the vector field must be 0.

For vector field learning, the error term

$$\text{ERR}[f^1, \ldots, f^T] = \frac{1}{n} \sum_{j=1}^{T} \sum_{i=1}^{n} (y_i^j - f^j(x_i^j))^2$$

[3]A vector field simply associates a vector to each point in a space.

can be written as

$$\mathrm{ERR}[f] = \frac{1}{n}\sum_{i=1}^{n} \|y_i - f(x_i)\|_T^2$$
$$\|y - f(x)\|_T^2 = \sum_{j=1}^{T} (y^j - f^j(x))^2,$$

with $f : X \to \mathbb{R}^T$ and $f = f^1, \ldots f^T$.

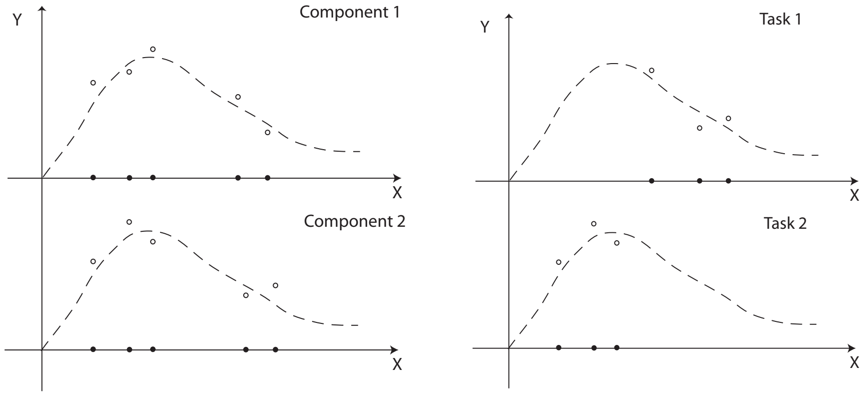

Figure 3: Relating vector fields and MTL. The multiple tasks in MTL correspond to the multiple components that make up the vectors in a vector field.

6.2 Multi-class and Multi-label

In the multi-class problem, each input can be assigned to one of T classes, and we can think of encoding each class with a vector, for example: class one can be $(1, 0 \ldots, 0)$, class 2 be $(0, 1 \ldots, 0)$, etc. In the multi-label problem, images contain at most T objects, and each input image is associated to a vector $(1, 0, 1 \ldots, 0)$, where 1/0 indicates presence/absence of an object.

In machine learning, the naïve multi-class classifier is to use the one-vs-all scheme with binary classifiers. This turns out to be identical to solving multi-label problems where we set correlations to 0. That is, solving for f_1 is equal to solving one-vs-all for task 1. Thus, it is actually hard to outperform the one-vs-all classifier.[4]

7 Final Remarks

Multi-output learning is a lively field, and we have discussed the problem in terms of kernels. The kernel/regularizer choice is crucial, particularly since multi-output problems tend to overfit due to the large number of model parameters and the small number of training samples. With this basic framework for MTL, we can extend the problem as we have for the scalar case and look at, for example, spectral, manifold, or sparsity-based regularization for multi-output problems.

[4]See Ryan Rifkin, "In defense of one-vs-all".