Notes on the framework of Ando and Zhang (2005)

Karl Stratos

1 Beyond learning good functions: learning good spaces

1.1 A single binary classification problem

Let X denote the problem domain. Suppose we want to learn a binary hypothesis function $h : X \to \{-1, +1\}$ defined as

$$h(x) := \begin{cases} 1 & \text{if } f(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where $f : X \to \mathbb{R}$ assigns a real-valued score to a given $x \in X$. We assume some loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that can be used to evaluate the prediction $f(x) \in \mathbb{R}$ against a given label $y \in \{-1, +1\}$. Here are some example loss functions:

$$L(y, y') := \log(1 + \exp(-yy')) \qquad \text{(logistic loss)}$$
$$L(y, y') := \max(0, 1 - yy') \qquad \text{(hinge loss)}$$

In a risk minimization approach, we seek a function f^{**} within a space of allowed functions $\mathcal{H}$ that minimizes the *expected* loss over the distribution $\mathcal{D}$ of instances $(x, y) \in X \times \{-1, +1\}$:

$$f^{**} := \arg\min_{f \in \mathcal{H}} \mathbf{E}_{(X,Y) \sim \mathcal{D}} \, L(f(X), Y)$$

Of course, we don't know the distribution $\mathcal{D}$. In practice, we seek a function f^* that minimizes the *empirical* loss over the training data $(x^{(1)}, y^{(1)}) \dots (x^{(n)}, y^{(n)})$:

$$f^* := \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(f(x^{(i)}), y^{(i)}) \tag{1}$$

this approach is called "empirical risk minimization" (ERM). One can easily find f^* if L is defined so that the objective in Eq. (1) is convex with respect to the function parameters.

1.2 Multiple binary classification problems

Suppose we now have m binary classification problems. Let X_l denote the problem domain of the l-th problem. We want to lean m binary hypothesis functions $h_1 \dots h_m$ defined as

$$h_l(x) := \begin{cases} 1 & \text{if } f_l(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where $f_l : X_l \to \mathbb{R}$ assigns a real-valued score to a given input $x \in X_l$ from the l-th domain. A simple way to learn $f_1 \dots f_m$ is to assume that each problem is independent so that there is no interaction between hypothesis spaces $\mathcal{H}_1 \dots \mathcal{H}_m$. Then following the usual ERM approach, we simply solve for each function separately. Assuming we have n_l training examples $(x^{(1;l)}, y^{(1;l)}) \dots (x^{(n_l;l)}, y^{(n_l;l)}) \in X_l \times \{-1, +1\}$ for the l-th problem drawn iid from distribution $\mathcal{D}_l$, find:

$$f_l^* := \arg\min_{f_l \in \mathcal{H}_l} \sum_{i=1}^{n_l} L(f_l(x^{(i;l)}), y^{(i;l)}) \tag{2}$$

for $l = 1 \dots m$.

A more interesting setting is that the problems share a certain common structure. To realize this setting, we introduce an abstract parameter $\Theta \in \Gamma$ for some parameter space Γ and assume that all the m hypothesis spaces are influenced by Θ. To mark this influence, we now denote the spaces by $\mathcal{H}_{1,\Theta} \dots \mathcal{H}_{m,\Theta}$ and functions by $f_{1,\Theta} \dots f_{m,\Theta}$. If we know Θ, then this new setting is reduced to the previous setting since the problems are conditionally independent given Θ. Again, we simply solve for each function separately in a manner similar to Eq. (2):

$$f^*_{l,\Theta} := \operatorname*{arg\,min}_{f_{l,\Theta} \in \mathcal{H}_{l,\Theta}} \sum_{i=1}^{n_l} L(f_{l,\Theta}(x^{(i;l)}), y^{(i;l)})$$

But instead of assuming a known $\Theta \in \Gamma$, we will optimize it jointly with the model parameters $\{w_l, v_l\}_{l=1}^m$. In this approach, we seek to minimize the sum of average losses across the m problems:

$$\Theta^*, \{f^*_{l,\Theta}\}_{l=1}^m := \operatorname*{arg\,min}_{\Theta \in \Gamma, f_{l,\Theta} \in \mathcal{H}_{l,\Theta}} \sum_{l=1}^{m} \frac{1}{n_l} \sum_{i=1}^{n_l} L(f_{l,\Theta}(x^{(i;l)}), y^{(i;l)}) \tag{3}$$

Note that for each problem $l = 1 \dots m$, we minimize the average loss (normalized by the number of training examples n_l) since the problems may have vastly different sizes of training data.

2 Learning strategies

To make the exposition in this section concrete, we will adopt the following setup of Ando and Zhang (2005). We assume a single feature function ϕ that maps an input $x \in X_l$ from any of the l problems to a vector $\phi(x) \in \mathbb{R}^d$. We consider the structural parameter space $\Gamma \subseteq \mathbb{R}^{k \times d}$ for some k. For the l-th problem and for a given structural parameter $\Theta \in \mathbb{R}^{k \times d}$, we consider a hypothesis space $H_{l,\Theta}$ of linear functions $f_{l,\Theta} : X_l \to \mathbb{R}$ parameterized by $w_l \in \mathbb{R}^d$ and $v_l \in \mathbb{R}^k$:

$$f_{l,\Theta}(x) = (w_l^\top + v_l^\top \Theta)\phi(x) \quad \forall x \in X_l \tag{4}$$

Here is how to interpret the setup. If Θ is a zero matrix, then the hypothesis function for each problem independently makes its own predictions as a standard linear classifier: $f_{l,\Theta}(x) = w_l^\top \phi(x)$. Otherwise, Θ serves as a global force that influences the predictions of all m hypothesis functions $f_{1,\Theta} \dots f_{m,\Theta}$.

Example 2.1. *Let m denote the number of distinct words in the vocabulary. Given an occurrence of word x in a corpus, we use $\phi(x) \in \mathbb{R}^d$ to represent the context of x. For example, for the sentence* `the dog barked`*, $\phi($`dog`$) \in \mathbb{R}^{2m}$ might be a two-hot encoding of words* `the` *and* `barked`*. Note that the word x itself is not encoded in $\phi(x)$.*

We have m binary predictors $f_{1,\Theta} \dots f_{m,\Theta}$ where $f_{l,\Theta}(x) = (w_l^\top + v_l^\top \Theta)\phi(x)$ corresponds to predicting whether x is the l-th word type given only the context $\phi(x)$.

2.1 Joint optimization on training data

Assume $R(\Theta)$ and $r_l(w_l, v_l)$ are some regularizers for matrices $\Theta \in \mathbb{R}^{k \times d}$ and vectors $w_l \in \mathbb{R}^d, v_l \in \mathbb{R}^k$. Then we can implement the abstract training objective in Eq. (3) by defining $\mathcal{H}_{l,\Theta}$ as a space of regularized linear functions parametrized by $\Theta \in \mathbb{R}^{k \times d}$, $w_l \in \mathbb{R}^d$, and $v_l \in \mathbb{R}^k$ as in Eq. (4):

$$\Theta^*, \{w_l^*, v_l^*\}_{l=1}^m := \operatorname*{arg\,min}_{\Theta, \{w_l, v_l\}_{l=1}^m} R(\Theta) + \sum_{l=1}^{m} \left(r_l(w_l, v_l) + \frac{1}{n_l} \sum_{i=1}^{n_l} L((w_l^\top + v_l^\top \Theta)\phi(x^{(i;l)}), y^{(i;l)}) \right) \tag{5}$$

We assume that r_l and L are chosen so that Eq. (5) is convex with respect to w_l, v_l for a fixed Θ. But this is not convex with respect to w_l, v_l, Θ jointly, so one cannot hope to find globally optimal $\Theta^*, \{w_l^*, v_l^*\}_{l=1}^m$. A natural optimization method is alternating minimization:

- Initialize Θ to some value.
- Repeat until convergence:

1. Optimize the objective in Eq. (5) with respect to w_l, v_l, with fixed Θ.
2. Optimize the objective in Eq. (5) with respect to Θ, with fixed w_l, v_l.

But Ando and Zhang (2005) modify the objective slightly so that the alternating minimization can use singular value decomposition (SVD) as a subroutine.

2.2 Alternating minimization using SVD

Here is a variant of the objective in Eq. (5):

$$\Theta^*, \{w_l^*, v_l^*\}_{l=1}^m := \underset{\Theta, \{w_l, v_l\}_{l=1}^m}{\arg\min} \sum_{l=1}^m \left(\lambda_l \left|\left|w_l\right|\right|^2 + \frac{1}{n_l} \sum_{i=1}^{n_l} L((w_l^\top + v_l^\top \Theta)\phi(x^{(i;l)}), y^{(i;l)}) \right) \tag{6}$$
$$\text{subject to } \Theta\Theta^\top = I_{k,k}$$

In other words, we set $r_l(w_l, v_l) = \lambda_l \left|\left|w_l\right|\right|^2$ (where $\lambda_l \geq 0$ is a regularization hyperparameter) and introduced an orthogonality constraint $\Theta\Theta^\top = I_{k,k}$. This constraint can be seen as implicitly regularizing the parameter Θ.

Now a key step: a change of variables. Define $u_l := w_l + \Theta^\top v_l \in \mathbb{R}^d$. We can then rewrite Eq. (6) as:

$$\Theta^*, \{u_l^*, v_l^*\}_{l=1}^m := \underset{\Theta, \{u_l, v_l\}_{l=1}^m}{\arg\min} \sum_{l=1}^m \left(\lambda_l \left|\left|u_l - \Theta^\top v_l\right|\right|^2 + \frac{1}{n_l} \sum_{i=1}^{n_l} L(u_l^\top \phi(x^{(i;l)}), y^{(i;l)}) \right) \tag{7}$$
$$\text{subject to } \Theta\Theta^\top = I_{k,k}$$

Once we have $\Theta^*, \{u_l^*, v_l^*\}_{l=1}^m$, we can recover $w_l^* = u_l^* - (\Theta^*)^\top v_l^*$. This non-obvious step is to achieve the following effect. When u_l is fixed for $l = 1 \ldots m$, Eq. (7) reduces to:

$$\Theta^*, \{v_l^*\}_{l=1}^m := \underset{\Theta, \{v_l\}_{l=1}^m}{\arg\min} \sum_{l=1}^m \lambda_l \left|\left|u_l - \Theta^\top v_l\right|\right|^2 \tag{8}$$
$$\text{subject to } \Theta\Theta^\top = I_{k,k}$$

One more key step: inspect what the optimal parameters $\{v_l^*\}_{l=1}^m$ are given a fixed Θ in Eq. (8). That is, analyze the form of

$$\{v_l^*\}_{l=1}^m := \underset{\{v_l\}_{l=1}^m}{\arg\min} \sum_{l=1}^m \lambda_l \left|\left|u_l - \Theta^\top v_l\right|\right|^2 \tag{9}$$

for some $\Theta \in \mathbb{R}^{k \times d}$ such that $\Theta\Theta^\top = I_{k,k}$. Differentiating the right hand side with respect to v_l and setting it to zero, we see that $v_l^* = \Theta u_l$. Note that this is the case for *any* given Θ, in particular the optimal Θ^* in Eq. (8). This is good, because we can now pretend there is no v_l in Eq. (8) by plugging in $v_l = \Theta u_l$. Eq. (8) is now formulated entirely in terms of Θ:

$$\Theta^* := \underset{\Theta}{\arg\max} \sum_{l=1}^m \lambda_l \ \ u_l^\top \Theta^\top \Theta u_l \tag{10}$$
$$\text{subject to } \Theta\Theta^\top = I_{k,k}$$

This form is almost there. To see what the solution Θ^* is more clearly, we take a final step: organize vectors $u_1 \ldots u_m$ and regularizing parameters $\lambda_1 \ldots \lambda_m$ into a matrix $M = [\sqrt{\lambda_1} u_1 \ldots \sqrt{\lambda_m} u_m] \in \mathbb{R}^{d \times m}$. By construction, the diagonal entries of $\Theta M M^\top \Theta^\top \in \mathbb{R}^{k \times k}$ have the following form: for $l = 1 \ldots k$,

$$[\Theta M M^\top \Theta^\top]_{l,l} = \lambda_l u_l^\top \Theta^\top \Theta u_l$$

Let $\theta_l \in \mathbb{R}^d$ denote the l-th row of $\Theta \in \mathbb{R}^{k \times d}$. Then Eq. (10) can be rewritten as:

$$\{\theta_1^* \ldots \theta_k^*\} := \underset{\theta_1 \ldots \theta_k}{\arg\max} \sum_{l=1}^m \lambda_l \ \ \theta_l^\top M M^\top \theta_l \tag{11}$$
$$\text{subject to } \theta_i^\top \theta_j = 1 \text{ if } i = j \text{ and } 0 \text{ otherwise}$$

At last, it is clear in Eq. (11) that the solution $\theta_1^* \dots \theta_k^*$ is the k eigenvectors of $MM^\top \in \mathbb{R}^{d \times d}$ that correspond to the largest k eigenvalues. Equivalently, they are the k left singular vectors of $M \in \mathbb{R}^{d \times m}$ that correspond to the largest k singular values.

Based on this derivation, we have an alternating minimization algorithm that uses SVD as a subroutine to approximate the solution in Eq. (6):

SVD-based Alternating Optimization of Ando and Zhang (2005)
Input: for $l = 1 \dots m$: training examples $(x^{(1;l)}, y^{(1;l)}) \dots (x^{(n_l;l)}, y^{(n_l;l)}) \in X_l \times \{-1, +1\}$, regularization parameters $\lambda_l \geq 0$; a suitable loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
Output: approximation of the parameters in Eq. (6): $\hat{w}_l \in \mathbb{R}^d$ and $\hat{v}_l \in \mathbb{R}^k$ for $l = 1 \dots m$, structurual parameter $\hat{\Theta} \in \mathbb{R}^{k \times d}$ across the m problems

- Initialize $\hat{u}_l \leftarrow 0 \in \mathbb{R}^d$ for $l = 1 \dots m$ and $\hat{\Theta} \in \mathbb{R}^{k \times d}$ randomly.
- Until convergence:
  1. For each $l = 1 \dots m$, set $\hat{v}_l = \hat{\Theta}\hat{u}_l$ and optimize the convex objective (e.g., using subgradient descent)

$$\hat{w}_l = \arg\min_{w_l \in \mathbb{R}^d} \lambda_l ||w_l||^2 + \frac{1}{n_l} \sum_{i=1}^{n_l} L((w_l^\top + \hat{v}_l^\top \hat{\Theta})\phi(x^{(i;l)}), y^{(i;l)})$$

  2. For each $l = 1 \dots m$, set $\hat{u}_l = \hat{w}_l + \hat{\Theta}^\top \hat{v}_l$ and set

$$\hat{M} = [\sqrt{\lambda_1}\hat{u}_1 \dots \sqrt{\lambda_m}\hat{u}_m] \in \mathbb{R}^{d \times m}$$

     Compute the k left singular vectors $\hat{\theta}_1 \dots \hat{\theta}_k \in \mathbb{R}^d$ of $\hat{M}$ corresponding to the k largest singular values and set $\hat{\Theta} = [\hat{\theta}_1 \dots \hat{\theta}_k]^\top$.
- Return $\hat{\Theta}$, $\{\hat{w}_l, \hat{v}_l\}_{l=1}^m$ where $\hat{v}_l \leftarrow \hat{\Theta}\hat{u}_l$.

3 Application to semi-supervised learning

Ando and Zhang (2005) propose applying this framework to semi-supervised learning, i.e., leveraging unlabeled data on top of the labeled data one already has. The basic idea is the following. Given labeled data for some supervised task, one creates a bunch of auxiliary labeled data for some *related* task. By finding a common structure Θ shared between the original task and the artificially constructed task, one can hope to utilize the information in unlabeled data.

How to artificially construct a task such that (1) it is related to the original task and (2) its labeled data can be easily obtained from unlabeled data is best illustrated with an example. Consider the task of associating a word in a sentence with a part-of-speech tag. For simplicity, we will not consider any structured predction such as viterbi decoding. Instead, we independently predict the part-of-speech tag of word x_j (the j-th word in sentence x) given some feature representation $\phi(x, j)$. We will probably want to include features such as the identity of words in a local context (x_{j-1}, x_j, x_{j+1}), the spelling of the considered word (prefixes and suffixes of x_j), and so on.

By observing that part-of-speech tags are intimiately correlated with word types themselves, we create an artificial problem of predicting the word in the current position given the previous and next words. This task is also illustrated in Example 2.1. Note that we can use unlabeled text for this problem. In summary, here is a description of how we might apply this framework to this semi-supervised task:

1. Apply the **SVD-based Alternating Optimization** algorithm to learn binary classifiers (corresponding to word types) that predict the word in the current position given the previous and next words. In the described setup, we can simply use the existing feature function $\phi(x, j)$ by setting all

irrelevant features (such as spelling features) to zero. Ando and Zhang (2005) report that iterating only once is sufficient in practice.

2. Fix the output structure $\hat{\Theta}$ from step 1. Throw away other parameters $\hat{w}_l, \hat{v}_l$ returned by the algorithm. Now, re-learn $\hat{w}_l, \hat{v}_l$ on the original task of predicting the part-of-speech tag of word x_j given $\phi(x, j)$, using the fixed $\hat{\Theta}$ from step 1 in the algorithm.

The output of this procedure is a set of binary classifiers for predicting the part-of-speech tag of a word in a sentence. But during training, these classifiers took advantage of the shared structure $\hat{\Theta}$ in the task of predicting the current word given neighboring words. Since these two tasks are very related, the original problem will benefit from using $\hat{\Theta}$.

Ando and Zhang (2005) report very good performance on sequence labeling tasks such as named-entity recognition. But because this framework is formulated entirely in terms of binary classification, they use their own sequence labeling method which also formulates the problem in terms of binary classification, which is out of the scope of this note.

References

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, **6**, 1817–1853.