Notes / Lab Book

Isak Falk

December 10, 2019

Contents

1 Notes

1.1 What is Meta Learning

Before we can answer if there is any active learning to be done, we need to understand what meta learning actually is. Below is a short list of questions we should answer in the lit review

- What is meta learning
  - A Perspective View and Survey of Meta-Learning
  - MAML for deep learning
    - Multi-task learning
      - Multi-Task learning
      - Sparse coding for multitask and transfer learning

            - Consistent Multitask learning with nonlinear output
            - Convex Learning of multiple tasks and their structure
            - An Overview of Multi-Task Learning in Deep NNs
            - Multi-Task Feature learning
            - A survey on Multi-Task Learning
        * Transfer learning
            - A survey on Transfer Learning
            - Self-taught learning: Transfer Learning from unlabeled data
            - Transfer learning via Dimensionality Reduction
    - Neigbouring fields
        * Dataset Shift
        * Learning-to-learn
            - Incremental Learning to Learn with statistical guarantees
            - Learning to learn around a common mean
        * Lifelong learning
        * Curriculum learning
- How can it be formulated
    - SLT
    - Other
- Equivalent problems in other fields
    - Statistics
    - Control
    - ML
    - Signal Processing
    - Operations Research
    - etc.
- Current approaches to Active Meta-Learning

1.2 Current Meta Learning setup

1.2.1 Setup

We have a meta distribution ρ_μ over some space of distributions $\mathcal{D}_\mathcal{Z}$ such that $\rho_\mathcal{Z} \sim \rho_\mu$, where $\mathcal{Z}$ is our sample space. Normally in supervised learning we have that $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. For each sampled distribution $\rho_\mathcal{Z}$ we sample a dataset (which we will call *task* to stay consistent with literature) $\mathcal{T} = (z_i)_{i=1}^n \sim \rho_\mathcal{Z}^n$ iid, which is split up into n^{tr}, n^{val} sized dataset such that $\mathcal{T} = D^{tr} \cup D^{val} = (z_i)_{i=1}^{n^{tr}} \cup (z_j)_{j=n^{tr}+1}^{n^{tr}+n^{val}}$ where D^{tr}, D^{val} is the train and validation dataset respectively.

Now assume the following problem, we have m distributions $(\rho_i)_{i=1}^m \sim \rho_\mu^m$ sampled iid. Each of these distributions ρ_i gives rise to a task $\mathcal{T}_i = D_i^{tr} \cup D_i^{val}$, where we make the simplifying assumption that $n_i = n_i^{tr} + n_i^{val}$ is the same for any i, so that $n_i^{tr} = n^{tr}, n_i^{val} = n^{val}$ and $n_i = n^{tr} + n^{val}$.

Our problem looks as follows, we want to find an algorithm $\mathcal{A}_\varphi(\cdot) : \mathcal{Z}^* \to \mathcal{H}$ where $\mathcal{Z}^*$ is the set of all possible datasets of any size and $\mathcal{H}$ is our hypothesis space, we let φ denote the *hyperparameters* (we will also write $\varphi = \mathcal{T}_I$ where I is some index set, e.g. $I = [\![k]\!]$ to show that we have run the meta-learning algorithm on $(\mathcal{T}_1, \dots, \mathcal{T}_k)$) of the algorithm which we want to learn, as we are doing meta-learning rather than normal supervised learning. In this sense we may learn φ from the outer loop, and the algorithm then uses these to adjust its behaviour when mapping from the train set to a target function in the

hypothesis class. We assume that $\mathcal{H}$ is the same for all distributions, such that $f_i \in \mathcal{H}$ for all i, but f_i differ in general.

(Rewriting of previous paragraph, we can view this as a three step process as follows:

1. The hyper-meta algorithm takes sets of tasks $(\mathcal{T}_i)_{i=1}^m$ and maps to algorithms that are
2. Meta algorithms that map from training sets D^{tr} to supervised algorithms that
3. Maps from training sets D^{tr} to a hypothesis set

Not sure if this is a fruitful way to look at it, but at least it clarifies the different levels we are operating on here. One way to write this (sloppily) could be as follows, if T^* is the set of all sequences of tasks, we call $\mathcal{M}_1$ the hyper-meta, $\mathcal{M}_0$ the meta, and $\mathcal{A}$ the base algorithm, then we have the following nested way of coupling them

$$\mathcal{M}_1 : T^* \to \{\mathcal{M}_0 : \mathcal{Z}^* \to \{\mathcal{A} : \mathcal{Z}^* \to \mathcal{H}\}\}$$

)

We introduce the following notation, the loss function is a function

$$\ell : \mathcal{Z} \times \mathcal{H} \to \mathbb{R}_+$$

which takes as input a sample point $z \in \mathcal{Z}$ and a hypothesis $h \in \mathcal{H}$ getting a loss of choosing this hypothesis for this sample point, $\ell(h, z)$. Note that this way of writing the loss may be highly non-linear as can be seen in the following case when $z = (x, y)$, $\ell(z, h) = (h(x) - y)^2$. The actual loss can now be written in the following form, for a task $\mathcal{T} = D^{tr} \cup D^{val}$, we have that the loss of the algorithm $\mathcal{A}_\phi(\cdot)$ is

$$L(\mathcal{A}_\phi(\cdot), \mathcal{T}) = \frac{1}{|D^{val}|} \sum_{z \in D^{val}} \ell(\mathcal{A}_\phi(D^{tr}), z).$$

From this we can formulate the learning problem for meta learning. Given the above, we want to understand how to perform well on the *meta-risk*

$$\mathcal{E}_{\rho_\mu}(\mathcal{A}_\phi(\cdot)) = \mathbb{E}_{\rho \sim \rho_\mu}\left[\mathbb{E}_{\mathcal{T} \sim \rho^n}\left[L(\mathcal{A}_\phi(\cdot)), \mathcal{T})\right]\right].$$

It's clear that we can actually rewrite this further, since relying on the iid assumption we have that

$$\begin{aligned}\mathbb{E}_{\mathcal{T} \sim \rho^n}\left[L(\mathcal{A}_\phi(\cdot)), \mathcal{T})\right] &= \mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathbb{E}_{D^{val} \sim \rho^{n_{val}}}\left[\frac{1}{|D^{val}|} \sum_{z \in D^{val}} \ell(\mathcal{A}_\phi(D^{tr}), z)\right]\right] \\ &= \mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathbb{E}_{z \sim \rho}\left[\ell(\mathcal{A}_\phi(D^{tr}), z)\right]\right]\end{aligned}$$

from which we get that the meta-risk can be expressed as

$$\begin{aligned}\mathcal{E}_{\rho_\mu}(\mathcal{A}_\phi(\cdot)) &= \mathbb{E}_{\rho \sim \rho_\mu}\left[\mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathbb{E}_{z \sim \rho}\left[\ell(\mathcal{A}_\phi(D^{tr}), z)\right]\right]\right] \\ &= \mathbb{E}_{\rho \sim \rho_\mu}\left[\mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathcal{E}_\rho(\mathcal{A}_\phi(D^{tr}))\right]\right]\end{aligned}$$

A way to do this is to do well in probability over the train set which is what is usually done in SLT. Explicitly, assuming as above that we have a set of distributions (which we will call *base* distributions, where base correspond to the base level in contrast to *meta* which corresponds to the meta level) $(\rho_i)_{i=1}^m \sim \rho_\mu^m$ iid, and each ρ_i gives rise to a task $\mathcal{T}_i$ sampled iid, then we are interested in bounds of the form

$$\Pr_{(\mathcal{T}_i)_{i=1}^m}\left(\mathcal{E}_{\rho_\mu}(\mathcal{A}_\phi(\cdot)) - \mathcal{E}_{\rho_\mu}(\mathcal{A}_*) \geq \epsilon\right) \leq \delta,$$

where $\mathcal{A}_* = \inf_{\mathcal{A}} \mathcal{E}_{\rho_\mu}(\mathcal{A})$. We probably want to constrain this in the future, but leave this like this for now.

1.2.2 Approach to solving this

The general problem is hard to solve, instead we consider how the generalisation error for an algorithm behaves. Consider the following expression (which differs from the one above but taken from photos of what Carlo wrote on screen), we assume that we have m different training tasks $M = (\mathcal{T}_i)_{i=1}^m$ and will use the shorthand $\mathcal{T}_{1:m}$ to mean all the tasks in index set. For an active learning algorithm on a meta-level,

for each $t \leq m$ we let M_t be a subset of tasks of size t, $M_t \subseteq M, |M_t| = t$. We are interested in quantifying the following

$$\Pr_M \left(\mathbb{E}_{\mathcal{T} \sim \rho_T} [L(\mathcal{A}_M(\cdot), \mathcal{T}) - L(\mathcal{A}_{M_t}(\cdot), \mathcal{T})] \geq \epsilon \right) \leq \delta$$

We make the following additional assumptions

- The base loss $\ell(f(x), y)$ is Lipschitz with respect to the second argument with constant L.
- The meta-algorithm $\mathcal{A}_\phi(\cdot)$ exists in some vector-valued reproducing kernel hilbert space [1]. In particular this means the following (following [2]), there is some vvRKHS $\mathcal{G}$ consisting of functions mapping from $\mathcal{X} \to \mathcal{H}$ where $\mathcal{H}$ is some separable Hilbert space, we will assume that $\mathcal{H} \subseteq \mathbb{R}^d$ since instances of datapoints normally comes in column form.

The definition of an vvRKHS is a generalisation of the univariate case. In particular the vvRKHS $\mathcal{G}$ is characterised by a so called *kernel of positive type* which is an operator values bi-linear map $\Gamma : \mathcal{X} \times \mathcal{X} \to B(\mathcal{H}, \mathcal{H})$. Since we assume that $\mathcal{H}$ is a subspace of Euclidean space, Γ will map to positive semi-definite matrices. The vvRKHS is built in a similar way to the univariate case with first a pre-Hilbert space which gets completed by adding the limit points, with the inner product

$$\langle \Gamma(x, \cdot))c, \Gamma(x', \cdot)c' \rangle_{\mathcal{G}} = \langle \Gamma(x, x')c, c' \rangle_{\mathcal{H}}$$

which leads to the reproducing property, for any $x \in \mathcal{X}, c \in \mathcal{H}$ and $g \in \mathcal{G}$, we have that

$$\langle g(x), c \rangle_{\mathcal{H}} = \langle g, \Gamma(x, \cdot) \rangle \, c_{\mathcal{G}}$$

and that for each $x \in \mathcal{X}$, the function $\Gamma(x, \cdot) : \mathcal{G} \to \mathcal{H}$ is the evaluation function in x on $\mathcal{G}$, that is $\Gamma(x, \cdot)(g) = g(x)$ and $\Gamma(x, \cdot) \in \mathcal{G}$.

Consider now the expression in the expectation, the expected deviation of the meta-loss between the meta-learning algorithm trained on the full dataset and the subset of taska, we can write this as follows

$$\begin{aligned} |L(\mathcal{A}_M(\mathcal{T})) - L(\mathcal{A}_{M_t}(\mathcal{T}))| &\leq \frac{1}{|D^{val}|} \sum_{z \in D^{val}} \left| \ell(\mathcal{A}_M(D^{tr}), z) - \ell(\mathcal{A}_{M_t}(D^{tr}), z) \right| \\ &= \frac{L}{|D^{val}|} \sum_{x \in D^{val}} \left| \mathcal{A}_M(D^{tr})(x) - \mathcal{A}_{M_t}(D^{tr})(x) \right| \end{aligned}$$

Now in order to proceed we need to think about how we can decouple D^{tr} and M or M_t. If we let $\mathcal{A}_M(\cdot) \in \mathcal{H}_1 \otimes \mathcal{H}_2$ and both $\mathcal{H}_1, \mathcal{H}_2$ are RKHS's then we have that $\mathcal{H}_1 \otimes \mathcal{H}_2$ is also an RKHS. Consider now

$$\begin{aligned} \left| \mathcal{A}_M(D^{tr})(x) - \mathcal{A}_{M_t}(D^{tr})(x) \right| &= \left| \left\langle \mathcal{A}_M(D^{tr}) - \mathcal{A}_{M_t}(D^{tr}), K_{\mathcal{H}_2}(x, \cdot) \right\rangle \right| \\ &\leq \left\| \mathcal{A}_M(D^{tr}) - \mathcal{A}_{M_t}(D^{tr}) \right\|_{\mathcal{H}_2} \| K_{\mathcal{H}_2}(x, \cdot) \|_{\mathcal{H}_2} \\ &= \left\| \left\langle \mathcal{A}_M(\cdot) - \mathcal{A}_{M_t}(\cdot), \mu_{\mathcal{H}_2}(D^{tr}) \right\rangle \right\|_{\mathcal{H}_2} \| K_{\mathcal{H}_2}(x, \cdot) \|_{\mathcal{H}_2} \end{aligned}$$

1.3 Solving biased KRR

Using the semi-parametric representer theorem we can represent the objective function as

$$J(\alpha, \beta) = \frac{1}{n} \| \mathbf{K}\alpha + \mathbf{\Psi}\beta - \mathbf{Y} \|_{\mathbb{R}^n}^2 + \lambda \alpha^\top \mathbf{K} \alpha.$$

where $\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^P, \mathbf{K} \in \mathbb{R}^{n \times n}, \mathbf{\Psi} \in \mathbb{R}^{n \times P}$. We can rewrite this further by defining $\theta = [\alpha, \beta]^T \in \mathbb{R}^{n+P}$ and $\mathbf{L} = [\mathbf{K}, \mathbf{\Psi}] \in \mathbb{R}^{n \times (n+P)}$

$$\mathbf{R} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n+P) \times (n+P)}$$

then means that the objective function is the same as

$$J(\theta) = \frac{1}{n} \| \mathbf{L}\theta - \mathbf{Y} \|_{\mathbb{R}^n}^2 + \lambda \theta^\top \mathbf{R} \theta.$$

which can be solved using normal least squares (note that things are a bit different and the hessian is not clearly p.d. hence minimizer might not be unique), which gives for the Jacobian and the Hessian the expressions

$$\nabla_\theta J = \frac{2}{n}(\mathbf{L}^\top \mathbf{L}\theta - \mathbf{L}^\top \mathbf{Y} + n\lambda \mathbf{R}\theta) = \frac{2}{n}(\mathbf{L}^\top(\mathbf{L}\theta - \mathbf{Y}) + n\lambda \mathbf{R}\theta) = \frac{2}{n}((\mathbf{L}^\top \mathbf{L} + n\lambda \mathbf{R})\theta - \mathbf{L}^\top \mathbf{Y})$$
$$\nabla^2_\theta J = \frac{2}{n}(\mathbf{L}^\top \mathbf{L} + n\lambda \mathbf{R})$$

Solving this we see that the solution, which we get by finding where the gradient is zero, is equal to

$$\begin{aligned}\theta^* &= \left(\mathbf{L}^\top \mathbf{L} + n\lambda \mathbf{R}\right)^{-1} \mathbf{L}^\top \mathbf{Y} \\ &= \left(\begin{bmatrix} \mathbf{K}^2 & \mathbf{K}\mathbf{\Psi} \\ \mathbf{\Psi}^\top \mathbf{K} & \mathbf{\Psi}^\top \mathbf{\Psi} \end{bmatrix} + n\lambda \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{K}\mathbf{Y} \\ \mathbf{\Psi}^\top \mathbf{Y} \end{bmatrix}\end{aligned}$$

1.4 Relation between Curriculum Learning and Meta Learning

The formal definition of a *curriculum* according to [3] is the following, let $P(z)$ be the target distribution and let $Q_\lambda(z)$ be the relaxed distribution where $\lambda \in [0, 1]$, where $\lambda = 0$ is a simpler toy problem and $\lambda = 1$ is the original problem $Q_1 = P$. Let $0 \leq W_\lambda(z) \leq 1$. Then $Q_\lambda(z) \propto W_\lambda(z)P(z)$ and is normalised. For a monotonically increasing sequence of values $(\lambda_l)_{l=1}^L$ with $\lambda_1 = 0, \lambda_L = 1$, the sequence of distribution $(Q_{\lambda_l})_{l=1}^L$ is a *curriculum* if if the entropy of the sequence of distribution is increasing, $H(Q_{\lambda_l}) < H(Q_{\lambda_{l+1}})$ and $(W_\lambda)_{l=1}^L$ is non-decreasing for all z, $W_{\lambda_l}(z) \leq W_{\lambda_{l+1}}(z)$.

For active learning, we simply let $Q_l := Q_{\lambda_l}$ which is an empirical distribution of size l, thus $W_l(z) = \mathbb{1}_{\{z \in Q_l\}}$ and we have that the sequence of W_l is monotonically increasing, since $Q_l \subseteq Q_{l+1}$ and $H(Q_l) = -\sum_{i=1}^l l^{-1} \log(l^{-1}) = \log(l)$ which is increasing in l. Hence as long as we use a uniform weight over the active learning set, active learning is a curriculum learning (which extends to any setting, supervised or meta learning).

1.5 Datasets

For testing out the current active meta learning strategy of using the MMD to actively select in what order to train on the datasets provided in the meta-learning setup we need to make sure that the datasets fit into our conditions and assumptions. For regression, this is clear, the L2 error is infinitely differentiable and as long as the base-algorithm is too, GD will be as well (but what about SGD? Could do some initial tests on that as well, for now we just make sure that the datasets are small, less than some n which makes it feasible to run the gradient descent algorithm which runs in $O(ln^2)$ where l is the number of GD steps. This should be feasible, especially)

1.6 Kernels on distributions

Kernels on distributions have been investigated in [4, 5] and furthermore kernels restricted on sets can be found in [6, 7] where they define and derive results about convolutional kernels, although with less theoretical foundation with respect to characteristicity and universality than that of kernels on distributions.

From [4] we have sufficient conditions for a function $K : X \times X \to \mathbb{R}$ to be a universal kernel, where universality means that the corresponding RKHS of K is dense in the set $C(X)$ of bounded continuous function from X to $\mathbb{R}$ in the uniform norm and in extension dense in all $L_1(\mu)$ for which μ has dense support. The following gives kernels which are universal on the set of probability measures, as can be seen in [4, Example 1], expanded upon below.

Let $(\mathcal{X}, d_\mathcal{X})$ be a compact metric space and $A := M_1^+(\mathcal{X})$ the set of all Borel probability measures on this space. In the case that $\mathcal{X} \subseteq \mathbb{R}^d$ we let $d_\mathcal{X}$ be the usual euclidean distance restricted to $\mathcal{X}$. If we equip A with the Prohorov metric and call this d_A, then (A, d_A) is a compact metric space iff $(\mathcal{X}, d_\mathcal{X})$ is a compact metric space. By using properties of RKHS's and compactness of $\mathcal{X}$ we have that if the kernel $K_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is *characteristic*, and define as $KME_\mathcal{X}(\rho) = \mathbb{E}_{X \sim \rho}[K_\mathcal{X}(X, \cdot)]$ then the following function is a kernel, is bounded and universal on A

$$K_\sigma(\rho, \rho') = \exp\left(-\frac{1}{2\sigma^2}\|KME_\mathcal{X}(\rho) - KME_\mathcal{X}(\rho)\|_{\mathcal{H}_\mathcal{X}}\right)$$

where $\mathcal{H}_{\mathcal{X}}$ is the RKHS corresponding to kernel $K_{\mathcal{X}}$.

We have a choice in what base kernel to choose when building universal kernels for probability distributions over compact sets

- A characteristic kernel $K_{\mathcal{X}}$, for example [8] (examples other than radial kernels can also be found there)
  - Gaussian kernels $k(x,y) = \exp\Big(-\frac{1}{2\sigma^2}\|x-y\|_2^2\Big), \sigma > 0$
  - Inverse multiquadratic kernels $k(x,y) = (c^2 + \|x-y\|_2^2)^{-\beta}, \beta > 0, c > 0$

After choosing this kernel $K_{\mathcal{X}}$, we "lift this kernel" by considering the mean embedding $KME_{\mathcal{X}}$ on $\mathcal{X}$ as a kernel on A. Note that this means that the RKHS for A and $\mathcal{X}$ coincide since the mean embedding embeds into the same RKHS as the original kernel (by Riesz representation theorem). We can simplify this further be recognising this as the KME being a function decomposed into one radial part and one KME part as

$$K_A(\rho,\xi) = f(\|KME_{\mathcal{X}}(\rho) - KME_{\mathcal{X}}(\xi)\|_{\mathcal{H}_{\mathcal{X}}})$$

where

$$f : [0,\infty) \to \mathbb{R}$$

is the function of the radius. In the case of the gaussian $f(r) = \exp\big(-\frac{1}{2\sigma^2}r^2\big)$ and for the inverse multiquadratic $f(r) = (c^2 + r^2)^{-\beta}$.

For implementation purposes, when working with radial basis functions, we only need to compute the expression

$$\|KME_{\mathcal{X}}(\rho) - KME_{\mathcal{X}}(\xi)\|^2 = \|KME_{\mathcal{X}}(\rho)\|^2 + \|KME_{\mathcal{X}}(\xi)\|^2 - 2\langle KME_{\mathcal{X}}(\rho), KME_{\mathcal{X}}(\xi)\rangle$$

in the case that we have a convex combination of dirac deltas, $\rho = \sum_{i=1}^n \alpha_i \delta_{x_i}, \xi = \sum_{j=1}^m \beta_j \delta_{y_j}$ this can be simplified to

$$\begin{aligned}\|KME_{\mathcal{X}}(\rho) - KME_{\mathcal{X}}(\xi)\|^2 &= \sum_{i,j}^n \alpha_i\alpha_j K_{\mathcal{X}}(x_i,x_j) + \sum_{l,t}^m \beta_l\beta_t K_{\mathcal{X}}(y_l,y_t) - 2\sum_{i,l}^{n,m}\alpha_i\beta_l K_{\mathcal{X}}(x_i,y_l)\\ &= \alpha^\intercal \mathbf{K}_{n,n}\alpha + \beta^\intercal \mathbf{K}_{m,m}\beta - 2\alpha^\intercal \mathbf{K}_{n,m}\beta\end{aligned}$$

which for empirical distributions, $\alpha = \frac{1}{n}\mathbf{1}, \beta = \frac{1}{m}\mathbf{1}$ reduces to

$$\begin{aligned}\|KME_{\mathcal{X}}(\rho) - KME_{\mathcal{X}}(\xi)\|^2 &= \frac{1}{n^2}\sum_{i,j}^n K_{\mathcal{X}}(x_i,x_j) + \frac{1}{m^2}\sum_{l,t}^m K_{\mathcal{X}}(y_l,y_t) - \frac{2}{nm}\sum_{i,l}^{n,m} K_{\mathcal{X}}(x_i,y_l)\\ &= \frac{1}{n^2}\mathbf{1}^\intercal \mathbf{K}_{n,n}\mathbf{1} + \frac{1}{m^2}\mathbf{1}^\intercal \mathbf{K}_{m,m}\mathbf{1} - \frac{2}{nm}\mathbf{1}^\intercal \mathbf{K}_{n,m}\mathbf{1}\end{aligned}$$

There are further examples of kernels that are universal, for example by considering characteristic functions of ρ but we will not consider them here.

2 Meetings

2.1 Meeting (Carlo) *<2019-10-22 Tue>*

2.1.1 Work

MMD for noiseless supervised learning We first broke down the actual problem, taking inspiration from what made the MMD bound possible in the MRes dissertation I wrote, we looked at what caused the bound (which only holds in the noiseless case)

$$|\mathcal{E}_P(h) - \mathcal{E}_Q(h)| \leq \mathrm{MMD}_{\mathcal{H}}(P,Q) + \eta_{MMD}$$

where we can control η_{MMD} by making careful choices about the regression and MMD RKHSs and how they relate to each other.

MMD for meta-learning Writing the above out explicitly, we have that for the supervised learning case that

$$|\mathcal{E}_P(h) - \mathcal{E}_Q(h)| = \left| \frac{1}{n_P} \sum_{i=1}^{n_P} \ell(h, z_i) - \frac{1}{n_Q} \sum_{j=1}^{n_Q} \ell(h, z_j) \right|$$

If we now consider our case, we can write this as follows

$$|\mathcal{E}_P(\mathcal{A}) - \mathcal{E}_Q(\mathcal{A})| = \left| \frac{1}{n_P} \sum_{i=1}^{n_P} L(\mathcal{A}, D_i^{tr}, D_i^{val}) - \frac{1}{n_Q} \sum_{j=1}^{n_Q} L(\mathcal{A}, D_j^{tr}, D_j^{val}) \right|$$

and each of the losses $L(\mathcal{A}, D^{tr}, D^{val})$ are defined as follows

$$L(\mathcal{A}, D^{tr}, D^{val}) = \frac{1}{|D^{val}|} \sum_{z \in D^{val}} \ell(\mathcal{A}(D^{tr}), z).$$

We proceed to make the following assumption, $\ell(h, z) = \langle \psi(h), \phi(z) \rangle_{\mathcal{G}}$ which is reminisent of the restriction in [2] on the loss function. This leads to the expression of the meta-loss as

$$L(\mathcal{A}, D^{tr}, D^{val}) = \left\langle \psi(\mathcal{A}(D^{tr})), \frac{1}{|D^{val}|} \sum_{z \in D^{val}} \phi(z) \right\rangle_{\mathcal{G}} = \left\langle \psi(\mathcal{A}(D^{tr})), \mu(D^{val}) \right\rangle_{\mathcal{G}}$$

where we use the feature map ϕ on $\mathcal{Z}$ to some RKHS $\mathcal{G}$ to induce a feature map on *sets* (or more generally distributions) through the mean embedding $\mu(D^{val})$. This leads to a mean embedding of mean embedding (which should be the same as having a *mixture distribution*, which is what we get over the actual supervised learning z's, where the tasks act as mixtures we draw z from. This leads to the following

$$\left| \frac{1}{n_P} \sum_{i=1}^{n_P} \left\langle \psi(\mathcal{A}(D_i^{tr})), \mu(D_i^{val}) \right\rangle - \frac{1}{n_Q} \sum_{j=1}^{n_Q} \left\langle \psi(\mathcal{A}(D_j^{tr})), \mu(D_j^{val}) \right\rangle \right| =$$

This does not seem too good, but maybe we can do something similar to what we did in the MMD proof.

Revisiting MMD proof Let's go through the proof in my dissertation / [9] and see if we can extract assumptions to make this work. Without any assumptions, for arbitrary A, B this holds

$$\begin{aligned} |\mathcal{E}_P(\mathcal{A}) - \mathcal{E}_Q(\mathcal{A})| &= \left| \frac{1}{n_P} \sum_{i=1}^{n_P} L(\mathcal{A}, D_i^{tr}, D_i^{val}) - \frac{1}{n_Q} \sum_{j=1}^{n_Q} L(\mathcal{A}, D_j^{tr}, D_j^{val}) \right| \\ &= |\mathcal{E}_P(\mathcal{A}) - A + A - B + B - \mathcal{E}_Q(\mathcal{A})| \\ &\leq |\mathcal{E}_P(\mathcal{A}) - A| + |B - A| + |\mathcal{E}_Q(\mathcal{A}) - B|. \end{aligned}$$

Note that we can also

2.1.2 Talk

Me and Carlo went over the setting and did some slight changes to notation. In general I have to make sure that I don't overload the characters I use for sets, distributions and so on (easier said than done given the amount of stuff I use).

2.1.3 Notes from meeting

- A good reference for Sobolev Spaces and functional theory: Adams, Sobolev Spaces

2.2 Meeting (Carlo) *<2019-11-04 Mon 17:00>*

2.2.1 Recap

Our setting is that of *meta-learning* where we have a meta distribution μ over distributions with support on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, our data-space. We have a base loss function $\ell : \mathcal{Z} \times \mathcal{H} \to \mathbb{R}_+$ where $\mathcal{H}$ is a hypothesis

class. We sample m iid distributions from μ giving us $(\rho_i)_{i=1}^m \sim \mu^m$ and for each $i \in [\![m]\!]$ we get an iid sample of n datapoints, $\mathcal{T}_i = (z_j)_{j=1}^n \sim \rho_i^n$.

For each dataset $\mathcal{T}_i$ we split this into a train and test set, $\mathcal{T}_i = D_i^{tr} \cup D_i^{val}$ with n_{tr} and n_{val} datapoints respectively. We define the meta-loss for an algorithm $\mathcal{A} : \mathcal{Z}^* \to \mathcal{H}$ as

$$\begin{aligned} L(\mathcal{A}, \mathcal{T}) := L(\mathcal{A}, D^{tr}, D^{val}) &= \frac{1}{|D^{val}|} \sum_{z \in D^{val}} \ell(\mathcal{A}(D^{tr}), z) \\ &= \mathcal{E}_{D^{val}}(\mathcal{A}(D^{tr})) \end{aligned}$$

and we are interested in how to find good meta-algorithms that has low meta-risk

$$\begin{aligned} \mathcal{E}_\mu(\mathcal{A}) &= \mathbb{E}_{\rho \sim \mu}\left[\mathbb{E}_{\mathcal{T} \sim \rho^n}\left[L(\mathcal{A}, \mathcal{T})\right]\right] \\ &= \mathbb{E}_{\rho \sim \mu}\left[\mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathbb{E}_{z \sim \rho}\left[\ell(\mathcal{A}(D^{tr}), z)\right]\right]\right] \\ &= \mathbb{E}_{\rho \sim \mu}\left[\mathbb{E}_{D^{tr} \sim \rho^{n_{tr}}}\left[\mathcal{E}_\rho(\mathcal{A}(D^{tr}))\right]\right]. \end{aligned}$$

We do not have access to μ but only have knowledge of the meta-distribution implicitly through the datasets $\mathcal{T}_i \sim \rho_i^n$.

2.2.2 Work

MMD As we are looking to do active learning, we are interested in bounds inspired by the supervised learning bound for the noiseless case of

$$|\mathcal{E}_P(h) - \mathcal{E}_Q(h)| \leq \mathrm{MMD}_{\mathcal{H}'}(P, Q) + \eta_{MMD}$$

so we consider the same starting point. Let $M = (\mathcal{T}_i)_{i=1}^m$ be the full meta-dataset and let M_t be a subset of these tasks of size t. Now for an arbitrary algorithm $\mathcal{A}$ consider

$$|\mathcal{E}_M(\mathcal{A}) - \mathcal{E}_{M_t}(\mathcal{A})| = \left| \frac{1}{m} \sum_{i=1}^m L(\mathcal{A}, D_i^{tr}, D_i^{val}) - \frac{1}{t} \sum_{j=1}^t L(\mathcal{A}, D_j^{tr}, D_j^{val}) \right|.$$

The bound achieved for MMD in a noiseless setting is dependent on the following assumptions

- ℓ is any loss
- For arbitrary $h \in H \subseteq \mathcal{H}$
- Any train set $S \subseteq P_0$ (here P_0 is the equivalent of M)
- With realizable case of $f \in \mathcal{H}$
- No noise; Distribution ρ determined by $\rho_{\mathcal{X}}$
- For any $H' \subseteq \mathcal{H}$

then

$$\mathcal{E}_{P_0}(h) \leq \mathcal{E}_S(h) + \mathrm{MMD}_{H'}(P_0^x, S^x) + \eta_{MMD}$$

where $\eta_{MMD} = 2 \min_{\tilde{g} \in H'} \max_{h \in H, x \in P_0^x} |\ell(f(x), h(x)) - \tilde{g}(x)|$. We can make this work for our setting pretty easily by considering the following decomposition

$$\begin{aligned} |\mathcal{E}_M(\mathcal{A}) - \mathcal{E}_{M_t}(\mathcal{A})| &= \left| \frac{1}{m} \sum_{i=1}^m L(\mathcal{A}, \mathcal{T}_i) - \frac{1}{t} \sum_{j=1}^t L(\mathcal{A}, \mathcal{T}_i) \right| \\ &\leq \left| \mathcal{E}_M(\mathcal{A}) - \frac{1}{m} \sum_{i=1}^m g(\mathcal{T}_i) \right| + \left| \frac{1}{m} \sum_{i=1}^m g(\mathcal{T}_i) - \frac{1}{t} \sum_{i=1}^t g(\mathcal{T}_j) \right| + \left| \mathcal{E}_{M_t}(\mathcal{A}) - \frac{1}{t} \sum_{i=1}^t g(\mathcal{T}_j) \right| \end{aligned}$$

If we assume that we have some RKHS suitably defined, which we call $\mathcal{G}$ and let $g \in G \subseteq \mathcal{G}$ then since $M_t \subseteq M$ we have that the middle term $\left| \frac{1}{m} \sum_{i=1}^m g(\mathcal{T}_i) - \frac{1}{t} \sum_{i=1}^t g(\mathcal{T}_j) \right| \leq \mathrm{MMD}_G(M, M_t)$ and the other two terms are less than $\sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$ that

$$|\mathcal{E}_M(\mathcal{A}) - \mathcal{E}_{M_t}(\mathcal{A})| \leq \mathrm{MMD}_G(M, M_t) + \inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$$

2.2.3 Notes from meeting

☒ Clean up notation, don't use for example $M, \mathcal{M}$ at the same time

MMD bound Given the bound

$$|\mathcal{E}_M(\mathcal{A}) - \mathcal{E}_{M_t}(\mathcal{A})| \leq \text{MMD}_G(M, M_t) + \inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$$

we can control the MMD term by choosing $\mathcal{G}$ in a proper way. We note that this will have to be an RKHS which maps from $2^{\mathcal{Z}}$ to $\mathbb{R}$. However this is slightly complicated by the second term, $\inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$. We need to choose $\mathcal{G}$ so that this term disappears, this means that we need $\mathcal{G}$ to be expressive enough to make this small (note to self, we assumed that the function was in a ball in order to get rid of this since $G \subseteq \mathcal{G}$ trades of how big the MMD terms get by a multiplicative factor (R, the radius) and how enlargening this R makes the second term grow smaller (we want to find the optimal tradeoff essentially).

Finding a good $\mathcal{G}$ In order to investigate this, we'd have to look at what kind of functions $L(\mathcal{A}, \mathcal{T})$ encode for various different algorithms $\mathcal{A}$ so that we can choose a $\mathcal{G}$ that contain functions approximating $L(\mathcal{A}, \mathcal{T})$ well, due to the term

$$\inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$$

Since this will depend on how we choose $\mathcal{A}$ we will focus on when $\mathcal{A}$ is such that $L(\mathcal{A}, \mathcal{T})$ is a smooth function of $\mathcal{T}$ which means that we need to know how

$$\begin{aligned} L(\mathcal{A}, \mathcal{T}) &= L(\mathcal{A}, D^{tr}, D^{val}) \\ &= \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \ell(\mathcal{A}(D^{tr}), z_i) \\ &= \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \ell(\mathcal{A}(D^{tr})(x_i), y_i) \end{aligned}$$

It's easy to see that L is built from the following parts

- $S(\mathbf{a}) = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} a_i = \frac{1}{n_{val}} \mathbf{1}^T \mathbf{a}$
- $\ell(y', y)$
- $\mathcal{A}(D^{tr})$
- $\mathcal{A}(D^{tr})(x)$

and we can write this in the form of $L = (S \circ \ell)(\mathcal{A}(D^{tr})(\mathbf{X}), \mathbf{Y})$ where

$$\mathcal{A}(D^{tr})(\mathbf{X}) = \begin{bmatrix} \mathcal{A}(D^{tr})(x_1) \\ \vdots \\ \mathcal{A}(D^{tr})(x_{n_{tr}}) \end{bmatrix}$$

$$\ell(\mathcal{A}(D^{tr})(\mathbf{X}), \mathbf{Y}) = \begin{bmatrix} \ell(\mathcal{A}(D^{tr})(x_1), y_1) \\ \vdots \\ \ell(\mathcal{A}(D^{tr})(x_{n_{tr}}), y_{n_{tr}}) \end{bmatrix}$$

The functional class of this will depend on how we choose $\mathcal{A}$. Below is an outline of what goes into a usual meta-algorithm.

Inner loop We choose an algorithm that maps from a train set D^{tr} to a hypothesis space $\mathcal{H}$. This will be done by minimizing the regularised ERM objective, $\mathcal{E}_{D^{tr},\lambda}(h) = \frac{1}{n_{tr}} \sum_{i=1}^{n^{tr}} \ell(h(x_i), y_i) + \lambda \|h\|_{\mathcal{H}}^2$ where we will let the optimisation be (S)GD (just call this SGD from here on) with early stopping.

Outer loop For example MAML [10], we will let this be for now and investigate this later. We would then like to know how the fine tuning factors in to this.

We can choose various architectures, which corresponds to the hypothesis space (given a starting point h_0 when doing SGD)

- MLP parametrised by (n_i, n_o, L, σ) which are in turn the dimensions of the input, output, number of hidden layers and the non-linearity. As long as σ is differentiable this function is differentiable.
- KRR parametrised by K, the kernel. Ditto for this.

KRR Least Squares solution Assume that in the inner loop we are doing KRR, which means the algorithm is the solution to the RERM problem over an RKHS $\mathcal{H}$ with some kernel K that is smooth in both its arguments

$$\mathcal{A}(D^{tr}) = \arg\min_{h \in \mathcal{H}} \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (h(x_i) - y_i)^2 + \lambda \|h\|_{\mathcal{H}}^2$$

which if we write $(\mathbf{K}_x)_i = K(x_i, x)$ gives us the solution

$$\mathcal{A}(D^{tr})(x) = \sum_{i=1}^{n_{tr}} \alpha_i K(x_i, x) = \mathbf{K}_x^T \alpha = \mathbf{K}_x^T (\mathbf{K} + n_{tr} \lambda I_{n_{tr}})^{-1} \mathbf{Y}$$

and it's clear that

- $\alpha(D^{tr}) = (\mathbf{K} + \lambda n_{tr} I)^{-1} \mathbf{Y}$ is smooth as a function of D^{tr}
  - K is smooth in both its arguments which means that $\mathbf{K}$ is smooth as a function of $\mathbf{X}$
  - The matrix $\mathbf{K} + n_{tr}\lambda I$ has smallest eigenvalue bounded away from zero so the inverse is also smooth as a function of $\mathbf{X}$. Note that we fix the size of the dataset which means that n_{tr} doesn't vary.
- $\sum_{i=1}^{n} \alpha_i K(x_i, x) = \mathbf{K}_x^T \alpha$ is smooth as a function of $\mathbf{X}$ and x for fixed α as it's a linear combination of smooth functions.
- Together we have that this is a smooth function of both D^{tr} and x which means that $L(\mathcal{A}, \mathcal{T})$ is a smooth function of $\mathcal{T}$ when $\mathcal{A}$ is KRR.

KRR Least Squares solution (with meta-algorithm from bias) Following [11] we may generalise the KRR to the meta-learning setting by generalising the RERM problem to

$$\mathcal{A}_{\lambda, h_0}(D^{tr}) = \arg\min_{h \in \mathcal{H}} \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (h(x_i) - y_i)^2 + \lambda \|h - h_0\|_{\mathcal{H}}^2$$

with the corresponding solution of (at least when the RKHS is finite-dimensional and we are doing normal LSQ)

$$\mathcal{A}(D^{tr})(x) = \left\langle \sum_{i=1}^{n_{tr}} \alpha_i \phi(x_i), \phi(x) \right\rangle_{\mathcal{H}} = \left\langle \mathbf{w}_{\lambda, h_0}(D^{tr}), \phi(x) \right\rangle_{\mathcal{H}}$$

where

$$\mathbf{w}_{\lambda, h_0}(D^{tr}) = (\phi(\mathbf{X})^T \phi(\mathbf{X}) + n\lambda I)^{-1} (\phi(\mathbf{X})^T \mathbf{Y} + n\lambda h_0) =$$

The analysis done in 2.2.3 still holds, although we can't express the function in its dual form anymore, as long as we are in finite-dimensional Euclidean space it does does not complicate things much. In this case we can also see, that if the change of the hyperparameters $(\lambda, h_0$ changes smoothly as a function of D^{tr} we still have that the space induced by $L(\mathcal{A}_{\lambda, h_0}, \mathcal{T})$ is smooth.

2.3 Meeting (Massi) *<2019-11-08 Fri 11:00>*

2.3.1 Talk

Me and Massi got up to date on what me and Carlo have been working on, debriefing him on the meta active learning setup with MMD. We talked through the points we needed to clarify:

- Isolate the differences in active learning in meta learning to that of active learning from supervised learning.
- Better understand $L_{\mathcal{A}}(\mathcal{T}) \coloneqq L(\mathcal{A}, \mathcal{T})$ as a function of $\mathcal{T}$.

Beside this, he recommended some papers I should look into

- [12] On Lipschitzness of meta-loss (not super relevant)

- [13] Active / Semi-supervised learning (relevant)

A quick note: Massi seemed pessimistic if this was relevant to pursuit, as the active learning bound for SL did not improve upon uniform sampling. **I should bring this up with Carlo and Massi**.

2.3.2 Notes from meeting

Multi-task learning with labeled and unlabeled tasks [13] In semi-supervised learning there are *smoothness conditions* that say that if the marginal distribution of two tasks are similar then the output should also be similar or explicitly called the *Semi-supervised smoothness assumption*:

If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

Looking further into this (and related conditions, see the book [14]) could be very useful for the active learning case. Possibly even making it possible to get conditions for which the active learning bounds using MMD are non-vacuous.

[12]

2.4 Meeting (Carlo) *<2019-11-11 Mon 16:00>*

2.4.1 Recap

MMD bound Given the bound

$$|\mathcal{E}_M(\mathcal{A}) - \mathcal{E}_{M_t}(\mathcal{A})| \leq \text{MMD}_G(M, M_t) + \inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$$

we can control the MMD term by choosing $\mathcal{G}$ in a proper way. We note that this will have to be an RKHS which maps from $2^{\mathcal{Z}}$ to $\mathbb{R}$. However this is slightly complicated by the second term, $\inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$. We need to choose $\mathcal{G}$ so that this term disappears, this means that we need $\mathcal{G}$ to be expressive enough to make this small (note to self, we assumed that the function was in a ball in order to get rid of this since $G \subseteq \mathcal{G}$ trades of how big the MMD terms get by a multiplicative factor (R, the radius) and how enlargening this R makes the second term grow smaller (we want to find the optimal tradeoff essentially).

Finding a good $\mathcal{G}$ In order to investigate this, we'd have to look at what kind of functions $L(\mathcal{A}, \mathcal{T})$ encode for various different algorithms $\mathcal{A}$ so that we can choose a $\mathcal{G}$ that contain functions approximating $L(\mathcal{A}, \mathcal{T})$ well, due to the term

$$\inf_{g \in G} \sup_{\mathcal{T} \in M} |L(\mathcal{A}, \mathcal{T}) - g(\mathcal{T})|$$

Since this will depend on how we choose $\mathcal{A}$ we will focus on when $\mathcal{A}$ is such that $L(\mathcal{A}, \mathcal{T})$ is a smooth function of $\mathcal{T}$ which means that we need to know how

$$\begin{aligned} L(\mathcal{A}, \mathcal{T}) &= L(\mathcal{A}, D^{tr}, D^{val}) \\ &= \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \ell(\mathcal{A}(D^{tr}), z_i) \\ &= \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \ell(\mathcal{A}(D^{tr})(x_i), y_i) \end{aligned}$$

It's easy to see that L is built from the following parts

- $S(\mathbf{a}) = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} a_i = \frac{1}{n_{val}} \mathbf{1}^T \mathbf{a}$
- $\ell(y', y)$
- $\mathcal{A}(D^{tr})$
- $\mathcal{A}(D^{tr})(x)$

and we can write this in the form of $L = (S \circ \ell)(\mathcal{A}(D^{tr})(\mathbf{X}), \mathbf{Y})$ where

$$\mathcal{A}(D^{tr})(\mathbf{X}) = \begin{bmatrix} \mathcal{A}(D^{tr})(x_1) \\ \vdots \\ \mathcal{A}(D^{tr})(x_{n_{tr}}) \end{bmatrix}$$

$$\ell(\mathcal{A}(D^{tr})(\mathbf{X}), \mathbf{Y}) = \begin{bmatrix} \ell(\mathcal{A}(D^{tr})(x_1), y_1) \\ \vdots \\ \ell(\mathcal{A}(D^{tr})(x_{n_{tr}}), y_{n_{tr}}) \end{bmatrix}$$

The functional class of this will depend on how we choose $\mathcal{A}$. Below is an outline of what goes into a usual meta-algorithm.

Inner loop We choose an algorithm that maps from a train set D^{tr} to a hypothesis space $\mathcal{H}$. This will be done by minimizing the regularised ERM objective, $\mathcal{E}_{D^{tr},\lambda}(h) = \frac{1}{n_{tr}} \sum_{i=1}^{n^{tr}} \ell(h(x_i), y_i) + \lambda \|h\|_{\mathcal{H}}^2$ where we will let the optimisation be (S)GD (just call this SGD from here on) with early stopping.

Outer loop For example MAML [10], we will let this be for now and investigate this later. We would then like to know how the fine tuning factors in to this.

We can choose various architectures, which corresponds to the hypothesis space (given a starting point h_0 when doing SGD)

- MLP parametrised by (n_i, n_o, L, σ) which are in turn the dimensions of the input, output, number of hidden layers and the non-linearity. As long as σ is differentiable this function is differentiable.
- KRR parametrised by K, the kernel. Ditto for this.

2.4.2 Work

ERM (KRR) smoothness

KRR Least Squares solution Assume that in the inner loop we are doing KRR, which means the algorithm is the solution to the RERM problem over an RKHS $\mathcal{H}$ with some kernel K that is smooth in both its arguments

$$\mathcal{A}(D^{tr}) = \arg\min_{h \in \mathcal{H}} \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (h(x_i) - y_i)^2 + \lambda \|h\|_{\mathcal{H}}^2$$

which if we write $(\mathbf{K}_x)_i = K(x_i, x)$ gives us the solution

$$\mathcal{A}(D^{tr})(x) = \sum_{i=1}^{n_{tr}} \alpha_i K(x_i, x) = \mathbf{K}_x^T \alpha = \mathbf{K}_x^T (\mathbf{K} + n_{tr} \lambda I_{n_{tr}})^{-1} \mathbf{Y}$$

and it's clear that

- $\alpha(D^{tr}) = (\mathbf{K} + \lambda n_{tr} I)^{-1} \mathbf{Y}$ is smooth as a function of D^{tr}
  - K is smooth in both its arguments which means that $\mathbf{K}$ is smooth as a function of $\mathbf{X}$
  - The matrix $\mathbf{K} + n_{tr}\lambda I$ has smallest eigenvalue bounded away from zero so the inverse is also smooth as a function of $\mathbf{X}$. Note that we fix the size of the dataset which means that n_{tr} doesn't vary.
- $\sum_{i=1}^{n} \alpha_i K(x_i, x) = \mathbf{K}_x^T \alpha$ is smooth as a function of $\mathbf{X}$ and x for fixed α as it's a linear combination of smooth functions.
- Together we have that this is a smooth function of both D^{tr} and x which means that $L(\mathcal{A}, \mathcal{T})$ is a smooth function of $\mathcal{T}$ when $\mathcal{A}$ is KRR.

KRR Least Squares solution (with meta-algorithm from bias) Following [11] we may generalise the KRR to the meta-learning setting by generalising the RERM problem to

$$\mathcal{A}_{\lambda,h_0}(D^{tr}) = \arg\min_{h \in \mathcal{H}} \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (h(x_i) - y_i)^2 + \lambda \|h - h_0\|_{\mathcal{H}}^2$$

with the corresponding solution of (at least when the RKHS is finite-dimensional and we are doing normal LSQ)

$$\mathcal{A}(D^{tr})(x) = \left\langle \sum_{i=1}^{n_{tr}} \alpha_i \phi(x_i), \phi(x) \right\rangle_{\mathcal{H}} = \left\langle \mathbf{w}_{\lambda,h_0}(D^{tr}), \phi(x) \right\rangle_{\mathcal{H}}$$

where

$$\mathbf{w}_{\lambda,h_0}(D^{tr}) = (\phi(\mathbf{X})^T\phi(\mathbf{X}) + n\lambda I)^{-1}(\phi(\mathbf{X})^T\mathbf{Y} + n\lambda h_0)$$

The analysis done in 2.2.3 still holds, although we can't express the function in its dual form anymore, as long as we are in finite-dimensional Euclidean space it does does not complicate things much. In this case we can also see, that if the change of the hyperparameters (λ, h_0 changes smoothly as a function of D^{tr} we still have that the space induced by $L(\mathcal{A}_{\lambda,h_0}, \mathcal{T})$ is smooth.

Early stopping If we instead opt to do gradient descent in the inner loop, such that we perform the following updates when doing KRR on the unpenalised regression problem with the the feature space being finite-dimensional again, we have the following update equation (in primal form)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_{t+1}\frac{1}{n}\phi(\mathbf{X})^T(\mathbf{Y} - \phi(\mathbf{X})\mathbf{w}_t)$$

We simplify this by considering the case of 1-step GD, that is,

$$\mathcal{A}(D^{tr})(x) = \left\langle \mathbf{w}_0 - \gamma_0\frac{1}{n}\phi(\mathbf{X})^T(\mathbf{Y} - \phi(\mathbf{X})\mathbf{w}_0), \phi(x) \right\rangle$$

which is smooth in the train set, which also means that as long as the inner loss is smooth, $L(\mathcal{A}, \mathcal{T})$ is a smooth function of $\mathcal{T}$. Furthermore this is true for any fixed $\mathbf{w}_0$ and it's simple to see that this generalises to any k-step GD. It's unclear what happens when we use a stopping criterion which is not defined explicitly in terms of the number of steps.

2.4.3 Notes from meeting

We worked through some things and through this we decided on the following things to do

2.5 Meeting (Carlo) *<2019-11-19 Tue 14:30>*

2.5.1 Questions

RKHS dense in the space of continuous functions For RKHS's defined over the space $C(\mathbb{R})$ (I think), you can show that these are dense, essentially, but is this true for us as we are looking at spaces $2^{\mathcal{Z}} \to \mathbb{R}$? And is this true uniformly, since we are looking at the worst case scenario

$$\sup_{\mathcal{T}} |L_{\mathcal{A}}(\mathcal{T}) - g|$$

Solution to biased KRR and inversion The solution can be found in the active meta learning document, but I assume that the Hessian is simply p.d., are there simpler condition (for example on $\mathbf{K}, \{\psi_p\}_{p=1}^P$) that makes this hold?

Estimating R Before (active learning dissertation) we assumed that the function we were trying to estimate, f could be found in some ball of radius R, but can we somehow estimate R from the data directly? Or do cross validation?

Kernels on Distributions See what actual kernels we can use, or if these somehow reduce to ordinary kernels.

3 Bibliography

References

[1] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: a review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[2] Carlo Ciliberto, Lorenzo Rosasco, and Alessandro Rudi. A consistent regularization approach for structured prediction. In *Advances in neural information processing systems*, pages 4412–4420, 2016.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[4] Andreas Christmann and Ingo Steinwart. Universal kernels on non-standard input spaces. In *Advances in neural information processing systems*, pages 406–414, 2010.

[5] Bui Thi Thien Trang, Jean-Michel Loubes, Laurent Risser, and Patricia Balaresque. Distribution regression model with a reproducing kernel hilbert space approach. *Communications in Statistics-Theory and Methods*, pages 1–23, 2019.

[6] David Haussler. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California ..., 1999.

[7] Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alexander J Smola. Multi-instance kernels. In *ICML*, volume 2, page 7, 2002.

[8] Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011.

[9] Tom J. Viering, Jesse H. Krijthe, and Marco Loog. Nuclear discrepancy for active learning, 2017.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[11] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems*, pages 10169–10179, 2018.

[12] Rishi Gupta and Tim Roughgarden. A pac approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[13] Anastasia Pentina and Christoph H Lampert. Multi-task learning with labeled and unlabeled tasks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2807–2816. JMLR. org, 2017.

[14] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.