1 Gaussian Task

Consider a supervised learning problem where n samples $S = \{X_i, Y_i\}_{i=1}^n$ are drawn from $P(\mathcal{X}, \mathcal{Y})$. We consider a family of classification problems

$$P(x, y) = \delta(y = 1)X_1 + \delta(y = -1)X_{-1}$$

where $X_1 \sim \mathcal{N}(\mu + \Delta, 1)$ and $X_{-1} \sim \mathcal{N}(-\mu + \Delta, 1)$.

We have n samples from a task with $\mu = 1$, $\Delta = 0$; We aim to generalize on this task. In addition, we have access to m samples from a task with $\mu = 1$, $\Delta > 0$. Let P_t and P_{ood} denote the two tasks respectively.

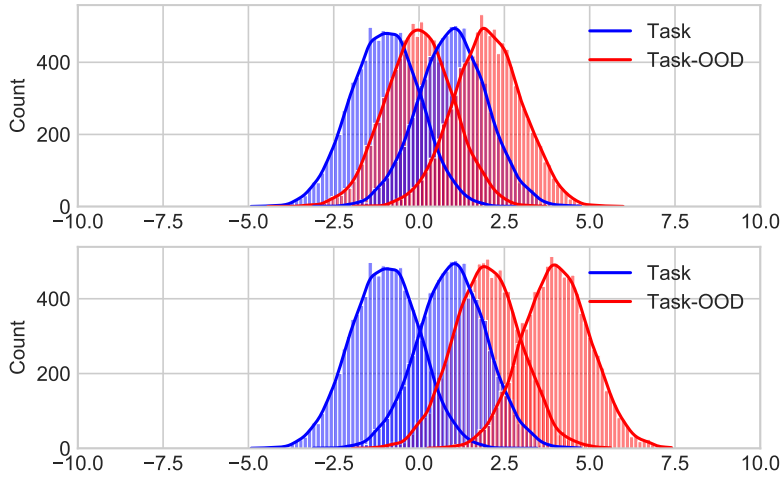

Figure 1: The two figures correspond to two different settings for the "Gaussian OOD data experiment" **(top)** Two tasks with very small Δ, which should make transfer easier **(bottom)** Two tasks with larger Δ where the transfer is expected to be smaller.

We are interested in understanding the behavior of learning from multiple tasks algorithms when m, n and Δ are varied.

2 Analysis of Algorithms that use OOD Data

We consider the linear hypothesis space H that splits the data into two classes using a single point. Formally, $f \in H$ is a hypothesis such that

$$f(x) = \begin{cases} y = 1 & x > h \\ y = -1 & x \leq h \end{cases}.$$

Every $f \in H$ corresponds to a scalar $h \in \mathbb{R}$. We will henceforth use f and h interchangeably and overload notation for h.

An OOD algorithm $A : (\mathcal{X}_t \times \mathcal{Y}_t)^n \times (\mathcal{X}_{\text{ood}} \times \mathcal{Y}_{\text{ood}})^m \mapsto \mathcal{H}$, maps n samples from the target task and m samples from an out-of-distribution task to a hypothesis that aims to minimize the population risk

$$e_t(h) = \mathbb{E}_{(x,y)\sim P_t} \left[\mathbf{1}[h(x) \neq y] \right].$$

Different draws of the training set S yield different hypotheses. Hence, we average over all possible draws of the training set to arrive at the objective

$$\mathcal{E}(A) = \mathbb{E}_{S\sim P^n} \left[e_t(A(S)) \right].$$

2.1 Computing $\mathcal{E}$

Let ϕ denote the density of the standard normal and let Φ denote the CDF of the standard normal. The population risk of hypothesis h is denoted by

$$e_t(h) = \frac{1}{2} \left((1 - \Phi(h + \mu) + \Phi(h - \mu) \right).$$

Assume that $\bar{h} = A(S)$ is a normally distributed random variable. We are interested in minimizing

$$\begin{aligned}
\mathbb{E}_{\bar{h}}[e_t(\bar{h})] &= \mathbb{E}_{\bar{h}} \left[\frac{1}{2} \left((1 - \Phi(h + \mu) + \Phi(h - \mu)) \right) \right] \\
&= \int_{-\infty}^{\infty} \left[\frac{1}{2} \left((1 - \Phi(h + \mu) + \Phi(h - \mu)) \right) \right] \phi \left(\frac{h - \bar{\mu}}{\bar{\sigma}} \right) \mathrm{d}h \\
&= \int_{-\infty}^{\infty} \frac{1}{2} \left[(1 - \Phi(y\bar{\sigma} + \bar{\mu} + \mu) + \Phi(y\bar{\sigma} + \bar{\mu} - \mu)) \right] \phi(y) \, \mathrm{d}y \\
&= \frac{1}{2} - \frac{1}{2} \int_{-\infty}^{\infty} \Phi(y\bar{\sigma} + \bar{\mu} + \mu)\phi(y)\mathrm{d}y + \frac{1}{2} \int_{-\infty}^{\infty} \Phi(y\bar{\sigma} + \bar{\mu} - \mu)\phi(y)\mathrm{d}y
\end{aligned}$$

We use the identity

$$\int_{\infty}^{\infty} \Phi \left(\frac{x - a}{b} \right) \phi(x) dx = \Phi \left(\frac{-a}{\sqrt{1 + b^2}} \right).$$

We can re-write the identity into a more convenient expression

$$\int_{\infty}^{\infty} \Phi(cx + d)\phi(x) dx = \int_{\infty}^{\infty} \Phi \left(\frac{x + d/c}{1/c} \right) \phi(x) dx = \Phi \left(\frac{d}{\sqrt{1 + c^2}} \right).$$

Using the above

$$\begin{aligned}
\mathbb{E}_{\bar{h}}[e_t(\bar{h})] &= \frac{1}{2} - \frac{1}{2} \int_{-\infty}^{\infty} \Phi(y\bar{\sigma} + \bar{\mu} + \mu)\phi(y)\mathrm{d}y + \frac{1}{2} \int_{-\infty}^{\infty} \Phi(y\bar{\sigma} + \bar{\mu} - \mu)\phi(y)\mathrm{d}y \\
&= \frac{1}{2} - \frac{1}{2}\Phi \left(\frac{\bar{\mu} + \mu}{\sqrt{1 + \bar{\sigma}^2}} \right) + \frac{1}{2}\Phi \left(\frac{\bar{\mu} - \mu}{\sqrt{1 + \bar{\sigma}^2}} \right) \qquad (1)
\end{aligned}$$

We optimize the objective using brute-force search. The variables to optimize are $\bar{\mu}$ and $\bar{\sigma}$.

2.2 Single-head Classifier: LDA

LDA finds a linear sub-space that maximizes intra-class variance while minimizing inter-class variance. It shares similarities to PCA, and can be understood to be its "label-aware" counterpart. This blog post is a good resource on the topic.

LDA can also be used as a classifier although this is less common. Consider a binary classification problem and assume $P(y=1|x)$ and $P(y=-1|x)$ are both Gaussians with identical variances (Σ). The classifier is given by:

$$f(x) = \mathbf{1}(w \cdot x > c)$$

where

$$w = \Sigma^{-1}(\bar{\mu}_1 - \bar{\mu}_{-1}) \qquad \text{and} \qquad c = w \cdot \frac{\bar{\mu}_1 + \bar{\mu}_{-1}}{2}$$

For the 1-dimensional case, the above classifier reduces to

$$f(x) = \mathbf{1}\left(\frac{\bar{\mu}_1 + \bar{\mu}_{-1}}{2}\right)$$

Consider tasks P_t and P_{ood} and samples from two tasks S_t and S_{ood}. We "mix" the two datasets to get

$$S = \alpha S_t + (1-\alpha) S_{ood}.$$

This affects the calculation of class means, which are random variables distributed as follows:

$$\bar{\mu}_{+1} \sim \mathcal{N}\left(\frac{+\alpha n\mu + (1-\alpha)m(+\mu+\Delta)}{\alpha n + (1-\alpha)m}, \sqrt{\frac{2((1-\alpha)^2 m + \alpha^2 n)}{(\alpha n + (1-\alpha)m)^2}}\right),$$

$$\bar{\mu}_{-1} \sim \mathcal{N}\left(\frac{-\alpha n\mu + (1-\alpha)m(-\mu+\Delta)}{\alpha n + (1-\alpha)m}, \sqrt{\frac{2((1-\alpha)^2 m + \alpha^2 n)}{(\alpha n + (1-\alpha)m)^2}}\right).$$

Hence,

$$\bar{h} = \frac{\bar{\mu}_1 + \bar{\mu}_{-1}}{2} \sim \mathcal{N}\left(\frac{(1-\alpha)m\Delta}{\alpha n + (1-\alpha)m}, \frac{\sqrt{(1-\alpha)^2 m + \alpha^2 n}}{(\alpha n + (1-\alpha)m)}\right)$$

or in other words $\bar{\mu} = \frac{(1-\alpha)m\Delta}{\alpha n + (1-\alpha)m}$ and $\bar{\sigma}^2 = \frac{(1-\alpha)^2 m + \alpha^2 n}{(\alpha n + (1-\alpha)m)^2}$.

fig. 2 includes results on this algorithm.

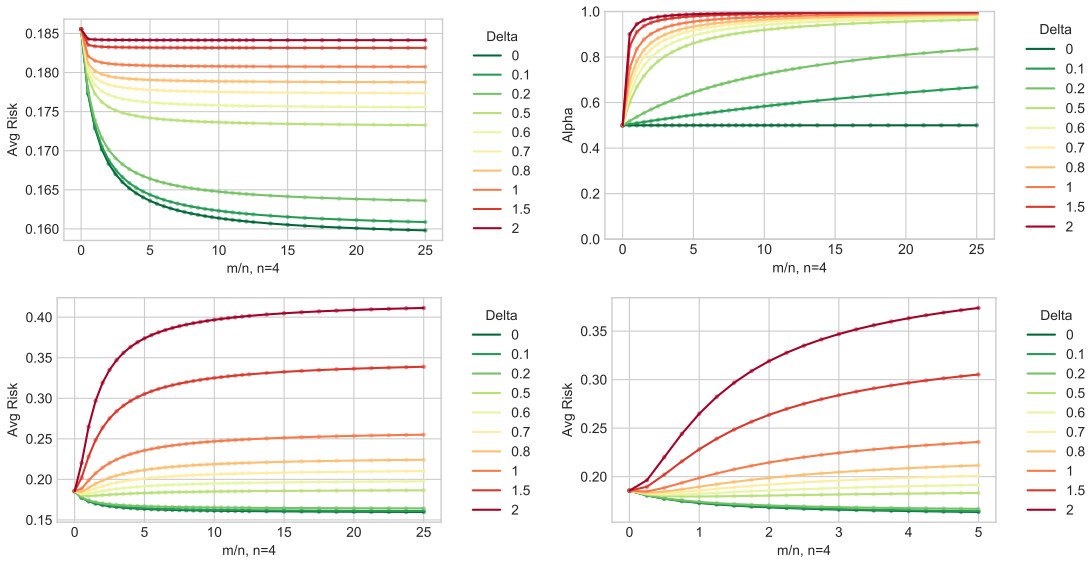

Figure 2: **(top-left)** Risk under optimal alpha and **(top-right)** The value of optimal alpha under the LDA model applied to the target and OOD tasks. Population risk when $\alpha = 0.5$ **(bot-right)** for larger range of m **(bot-right)** and smaller range of m emulating Ashwin's setup.

Note: The optimal value of α is obtained by minimizing $E[e_t(\bar{h})]$. An analytic expression of α is difficult to compute with both $\bar{\mu}$ and $\bar{\sigma}$ depending on α. Hence we use brute-force search to find the solution. We expect eq. (1) to be a smooth function so heuristic random search algorithms will be effective.

2.3 Interpolating the Hypotheses

Consider an algorithm which estimates h_t using n samples using the equation

$$h_t = \frac{\hat{\mathbb{E}}(X|y=1) + \hat{\mathbb{E}}(X|y=-1)}{2} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

In other words, we estimate the mean of Gaussians corresponding to each class and consider the bisector of the two means to be the hypothesis.

Given m samples from P_t and n samples from P_{ood}, we combine data from both datasets to obtain a hypothesis

$$\bar{h} = \alpha h_t + (1-\alpha) h_{\text{ood}}$$

where α is a function of n, m and Δ. Unlike the previous section, α is a factor that mixes the two hypotheses, as opposed to the two datasets.

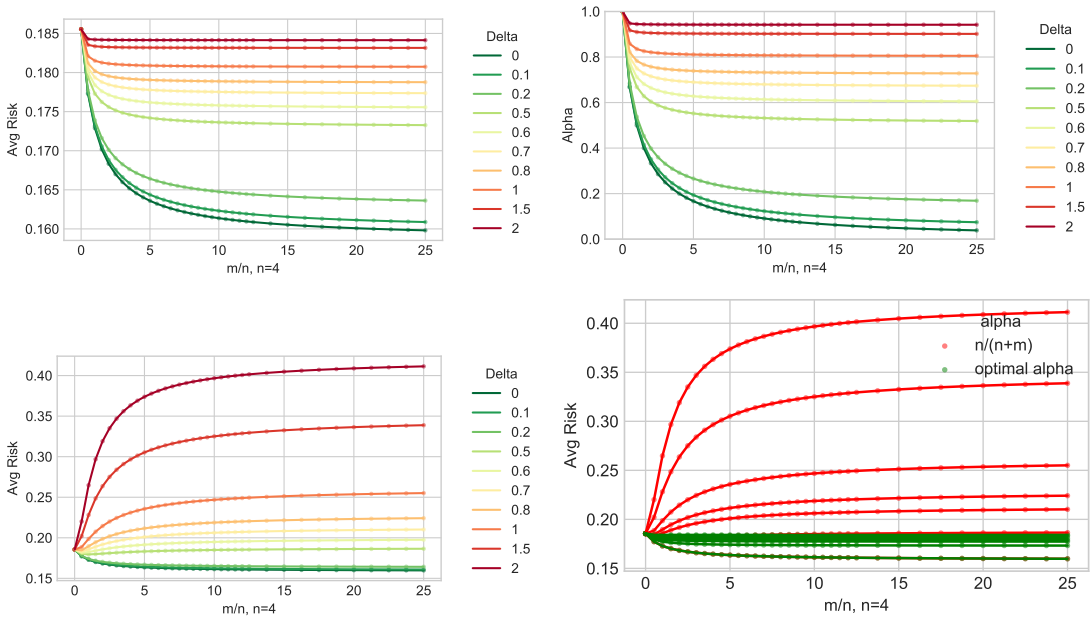

Figure 3: **top-right**: Large the value of Δ, smaller the value of α **top-left**: In all cases, extra data does not hurt if we choose the right value of α. The loss never becomes worse with more samples. **(bot-right)** Loss plotted for $\alpha = n/(n+m)$. **bot-right**: The losses plotted on the same scale when α is tuned/fixed.

Assume that S has $\frac{n}{2}$ samples from each class in both P_t and P_{ood}. h_t is an average of n Gaussian random variables from P_t and is distributed as

$$h_t \sim \mathcal{N}\left(0, \sqrt{\frac{1}{n}}\right).$$

Similarly

$$h_{\text{ood}} \sim \mathcal{N}\left(\Delta, \sqrt{\frac{1}{m}}\right),$$

since h_{ood} is the summation of $\frac{m}{2}$ random variables with law $\mathcal{N}(\mu + \Delta, 1)$ and $\frac{m}{2}$ random variables with law $\mathcal{N}(-\mu + \Delta, 1)$. The resultant hypothesis is also normally distributed over draws of the samples i.e., $\bar{h} \sim \mathcal{N}(\bar{\mu}, \bar{\sigma})$ where

$$\bar{\mu} = (1-\alpha)\Delta \qquad \text{and} \qquad \bar{\sigma}^2 = \frac{\alpha^2}{n} + \frac{(1-\alpha)^2}{m}$$

fig. 3 includes results on this algorithm.

3 Multi-head

The multi-head model assumes a weaker condition on the tasks. Instead of using the same classifier, each task has its own task-specific classification layer. The hypothesis for task i is denoted by

$$h_i = g_i \circ f$$

where $f : \mathbb{R}^p \mapsto \mathbb{R}^k$ is a shared feature generator.

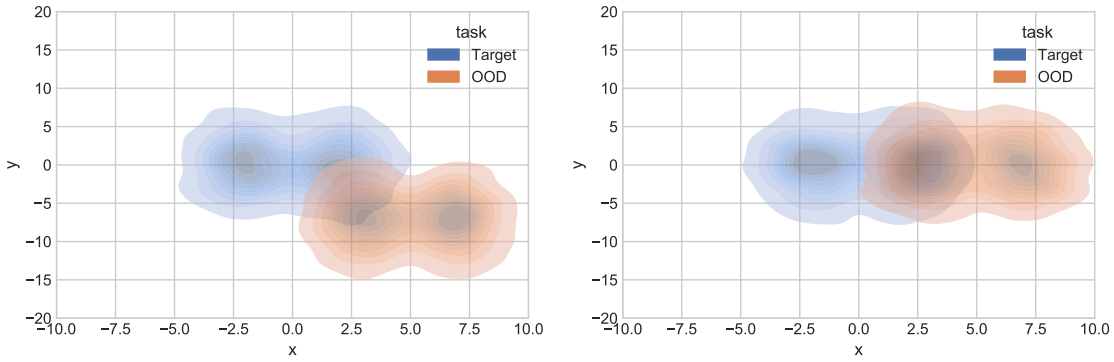

Figure 4: Examples of two tasks which are translations of each other. Both tasks have the same ideal 1-dimensional embedding so benefit from being trained together in the multi-head model.

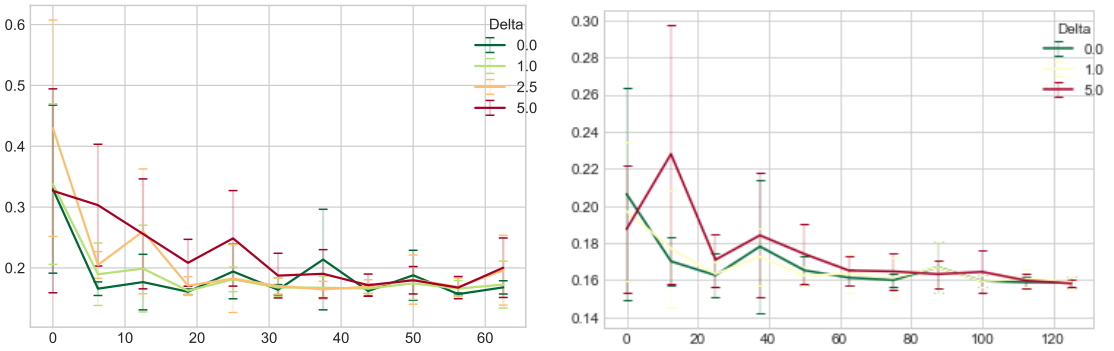

Figure 5: We consider (left) n=2 and (right) n=8. In both cases, the Multi-head model benefits from more OOD data.

As a simple example, we consider target and OOD tasks formed from the two-dimensional Gaussians. The construction is similar to 1D-Gaussian tasks described earlier. We consider a target task with the class means of $(1, 0)$ and $(-1, 0)$. The variance of both classes is

$$\Sigma = \begin{bmatrix} 1.0 & 0 \\ 0 & 10.0 \end{bmatrix}$$

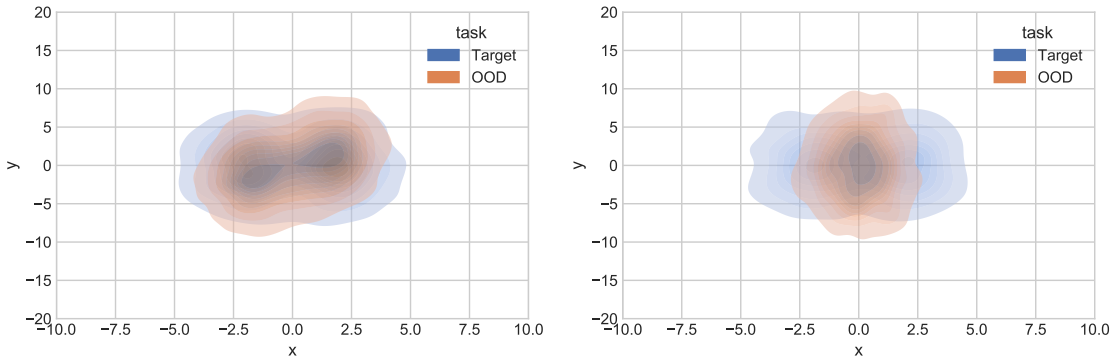

Figure 6: Examples of two tasks which are rotations of each other. For the figure of the right, there exists no one-dimensional embedding that can optimally solve both tasks

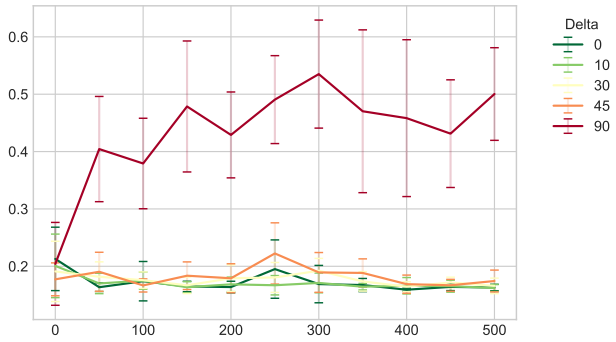

Figure 7: We consider rotated tasks and note that, the performance degrades as we rotate to 90 degrees. We believe that the degradation is not gradual since $\sigma_{yy} = 10 \times \sigma_{xx}$ (see right fig. 6.

We consider the following neural network architecture for this task

$$\text{input} \to \text{FC}(2, 100) \to \text{FC}(100, 1) \to \text{FC}(1, 1)_i$$

The last layer is the task-specific layer. This architectures attempts to learn a 1-dimensional representation that can separate the labels of both tasks.

We use 100 neurons in an intermediate layer since the neural network initializes poorly without this layer. A large number of randomly initializing neurons results in higher probability that all the output logits are 0.0.

Baxter's model is a generalization of this architecture; It attempts to find a sub-space $\mathbb{R}^k$ that separates the labels of all tasks. In essence, it maximizes inter-class variance and minimize intra-class variance averaged over all tasks.

For the Gaussian case, we expect this model to work for a family of tasks which are translations

of each other. For all such tasks, the same 1D embedding (line parallel to x-axis) is the optimal embedding and hence, data from OOD tasks are beneficial. We observe this in fig. 5.

However, this benefit does not extend to rotations. When the OOD task is a rotated version of the original task, then there exists no clear 1-dimensional embedding that works well for both tasks.

3.1 Single-head vs Multi-head

We can understand augmentations as different tasks trained using a single-head model. In general, single-head works with a more stringent notion of relatedness. The tasks need to have distributional overlap in order to guarantee any success. At the same time, the single-head model yields greater benefits when tasks are closely related.

Multi-head enforces fewer constraints and allows tasks to be arbitrarily far apart in k dimensions as long as the k dimensions preserve the structure of each task.