A Details for Max-Margin Motivation

The following optimization problem is one form of an N-class max-margin problem, i.e. a multi-class support vector machine [13], on a training set $\{(x_i, y_i)\}_{i=1}^m$:

$$\min_{w_1,\ldots,w_N} \sum_{y=1}^{N} \|w_y\|^2 \text{ s.t. } \forall i \in [m],\ \forall y' \neq y_i,\ w_{y_i}^\mathsf{T} x_i \geq w_{y'}^\mathsf{T} x_i + 1. \tag{6}$$

This is a "hard" version of the problem used as a classification head by MetaOptNet [40], and can be obtained in their framework by taking the penalty parameter $C \to \infty$.

The decision boundaries obtained by small-step-size gradient descent for linear predictors with cross-entropy loss on separable data converge to those obtained by (6), as shown by Theorem 7 in Soudry *et al.* [62], for almost all datasets. Thus, ANIL [51], which uses gradient descent for linear predictors with cross-entropy loss on separable data, will approximately obtain the same solution when using enough steps with appropriately small learning rates.

MetaOptNet uses the homogeneous predictors discussed here. We can handle non-homogeneous linear predictors ($w^\mathsf{T} x + b$ instead of just $w^\mathsf{T} x$) with the standard trick of adding a constant 1 feature to each data point. This solution actually does not quite maximize the margin on the original problem, since it effectively adds b^2 to the objective in (6), but ANIL will find exactly this same solution when using gradient descent on a function with a separate intercept.

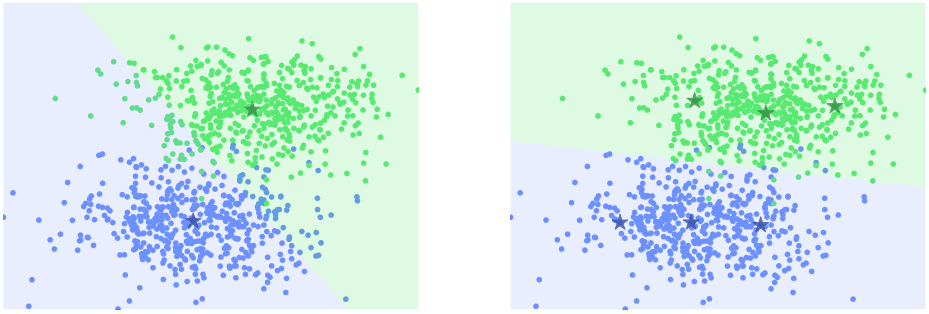
As visualized in Fig. 2 and explained in Sec. 3.3, if the class-conditional data distributions are isotropic Gaussians with the same covariance matrices, it is more advantageous to label the cluster centers than a random point from each cluster (supported by Corollary 1). We provide the proof for Corollary 1 below.

Corollary 1. *Suppose* $Y \sim \text{Uniform}([N])$*, and* $X \mid (Y = y) \sim \mathcal{N}(\mu_y, \sigma^2 I)$*, where the* μ_i *are orthonormal. Then the max-margin separator* (6) *on* $\{(\mu_i, i)\}_{i=1}^N$ *is Bayes-optimal for* $Y \mid (X = x)$*.*

Proof. Combine Proposition 1 and Lemma 1 below. □

The orthonormal assumption keeps the proof tractable; far more analysis would be needed without it. With high-dimensional meta-learned features that are well-aligned to the learning problem, however, it is reasonable to expect that inner products between different classes will be much smaller than the within-class inner products.

This optimality result can break when the clusters do not share a spherical covariance; consider Fig. 4a, where the data is still Gaussian but the shared class-conditional covariance is not spherical. In the one-shot case, max-margin on the separators does not choose the optimal separator. In this case, we could manually select points to choose the correct line. Doing so, however, is quite risky; since we do not know the data labels (or that it is actually Gaussian), we might incorrectly separate the data. Figure 4b shows the same problem in a three-shot setting; here, even though the data is truly generated from a mixture



(a) Trained on N cluster centers **(b)** Trained on $3N$ cluster centers

Fig. 4: Decision boundaries using a multiclass SVM (6) trained on cluster centers (shown by stars), with (a) the one-shot case and (b) the three-shot case.

of two Gaussians, fitting a mixture of six Gaussians gives us an approximate set cover of the data, and the max-margin separator now works well.

In fact, we can expect that (a) as the number of clusters grows, the cluster centers produce a better and better set cover of the dataset; (b) the max-margin separator on a set cover will approximate the max-margin separator on the full dataset, since the support vectors are all nearby.

A.1 Proofs

Proposition 1. *Suppose that* $\{x_i\}_{i=1}^N$ *are orthonormal. Then, the solution to* (6) *with the dataset* $\{(x_y, y)\}_{y=1}^N$ *is given by* $w_y = x_y - \frac{1}{N}\sum_{i=1}^N x_i$*, and hence*

$$\text{for any } x, \quad \arg\max_y w_y^\mathsf{T} x = \arg\min_y \|x - x_y\| \tag{5}$$

Proof. We will be able to analytically solve the KKT conditions for (6) in this case. Rather than using existing analyses of (6), it will be simpler to directly analyze this particular case.

Let $\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \in \mathbb{R}^{Nd}$, where d is the dimension of the x_i and w_y. The objective of our optimization problem is then simply $\|\mathbf{w}\|^2$.

We will next define a matrix A such that the constraints can be written as $A\mathbf{w} + \mathbf{1} \leq \mathbf{0}$, with $A \in \mathbb{R}^{N(N-1)\times Nd}$ and $\leq$ interpreted elementwise. Each constraint is of the form $-w_i^\mathsf{T} x_i + w_j^\mathsf{T} x_i + 1 \leq 0$, where $i \neq j$ are class indices in $[N]$. We can write the corresponding row of A as $(E_j - E_i)x_i$, where $E_i \in \mathbb{R}^{Nd\times d}$ are given by $E_i = \begin{bmatrix} 0_{(i-1)d\times d} \\ I_d \\ 0_{(N-i-1)d\times d} \end{bmatrix}$; these E_i are a block-matrix analogue of standard basis vectors, so that $E_i x_i \in \mathbb{R}^{Nd}$ has x_i in the ith block of d coordinates, and 0 elsewhere. We will order these constraints in A in "row-major" order: recalling that $i \neq j$, this means we have first $i = 1$ $j = 2$, then $i = 1$ $j = 3$,

up to $i = 1$ $j = N$, followed by $i = 2$ $j = 1$, $i = 2$ $j = 3$, and so on. Let $\ell(i, j)$ give the index of the corresponding constraint, so that e.g. $\ell(1, 3) = 2$.

Now, the problem can be written

$$\min_{\mathbf{w}\in\mathbb{R}^{Nd}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ s.t. } A\mathbf{w} + \mathbf{1} \leq \mathbf{0},$$

with the $\frac{1}{2}$ introduced for convenience. The KKT conditions for this problem are

$$\mathbf{w} + A^\mathsf{T}\mu = \mathbf{0} \quad A\mathbf{w} + \mathbf{1} \leq \mathbf{0} \quad \mu \geq \mathbf{0} \quad \mu \odot (A\mathbf{w} + \mathbf{1}) = \mathbf{0},$$

where $\odot$ is elementwise multiplication. From the first condition, $\mathbf{w} = -A^\mathsf{T}\mu$, where $\mu \in \mathbb{R}^{N(N-1)}$ is any vector satisfying

$$\mu \geq \mathbf{0} \quad AA^\mathsf{T}\mu - \mathbf{1} \geq \mathbf{0} \quad \mu \odot (AA^\mathsf{T}\mu - \mathbf{1}) = \mathbf{0}.$$

Since (6) is a strictly convex minimization problem with affine constraints, these conditions are necessary and sufficient for optimality, and the solution $\mathbf{w}$ is unique.

We can reasonably expect, since the x_i are orthonormal, that all constraints should be active, meaning that $AA^\mathsf{T}\mu = \mathbf{1}$. Indeed, choosing $\mu = (AA^\mathsf{T})^{-1}\mathbf{1}$ automatically satisfies the second and third conditions; it only remains to show that this $\mu \geq \mathbf{0}$ in order to show this as an optimal solution to (6).

To do this, we will explicitly characterize AA^T:

$$(AA^\mathsf{T})_{\ell(i,j),\ell(i',j')} = x_i^\mathsf{T}(E_j - E_i)^\mathsf{T}(E_{j'} - E_{i'})x_{i'} = (\delta_{ii'} + \delta_{jj'} - \delta_{ij'} - \delta_{ji'})\, x_i^\mathsf{T}x_{i'},$$

where $\delta_{ij} = \mathbb{1}(i = j)$ is the Kronecker delta, since $E_i^\mathsf{T}E_j = \delta_{ij}I_d$.

Since the x_i are orthonormal, $x_i^\mathsf{T}x_{i'} = \delta_{ii'}$. As we know $i \neq j$ and $i' \neq j'$, this simplifies to

$$(AA^\mathsf{T})_{\ell(i,j),\ell(i',j')} = \delta_{ii'}(1 + \delta_{jj'}).$$

Thus (AA^T) is a block matrix with diagonal blocks of size $(N-1) \times (N-1)$ with values $I_{N-1} + \mathbf{1}_{N-1}\mathbf{1}_{N-1}^\mathsf{T}$, and all off-diagonal blocks zero. Taking $\mu = (AA^\mathsf{T})^{-1}\mathbf{1}_{N(N-1)}$, the zero blocks contribute nothing, so each block of $N-1$ entries of μ is $(I_{N-1} + \mathbf{1}_{N-1}\mathbf{1}_{N-1})^{-1}\mathbf{1}_{N-1}$.

Note that $\mathbf{1}_{N-1}\mathbf{1}_{N-1}^\mathsf{T}$ has one eigenvector $v_1 = \frac{1}{\sqrt{N-1}}\mathbf{1}$ with eigenvalue $\lambda_1 = N - 1$, and the remaining eigenvalues are all zero with eigenvectors satisfying $v_i^\mathsf{T}\mathbf{1} = 0$. Adding I to this matrix simply increases all eigenvalues by one. Thus,

$$\left(I + \mathbf{1}\mathbf{1}^\mathsf{T}\right)^{-1}\mathbf{1} = \frac{1}{N}\left(\frac{1}{\sqrt{N-1}}\mathbf{1}\right)\left(\frac{1}{\sqrt{N-1}}\mathbf{1}\right)^\mathsf{T}\mathbf{1} \tag{7}$$

$$+ \sum_{i=2}^{N-1} v_i \underbrace{v_i^\mathsf{T}\mathbf{1}}_{0} = \frac{1}{N}\underbrace{\frac{\mathbf{1}^\mathsf{T}\mathbf{1}}{N-1}}_{1}\mathbf{1} = \frac{1}{N}\mathbf{1}, \tag{8}$$

and so $\mu = \frac{1}{N}\mathbf{1}_{N(N-1)}$, which is indeed $\geq \mathbf{0}$; thus this is an optimal solution to the problem.

We next reconstruct $\mathbf{w} = -A^\mathsf{T}\mu = -\frac{1}{N}A^\mathsf{T}\mathbf{1}_{N(N-1)}$. Consider the block w_i inside $\mathbf{w}$; its value will be the negative mean of the entries of A with an E_i in them. The $\ell(i,j)$ rows for $j \neq i$ contribute $N-1$ entries of the form $-E_i x_i$. We also have the $\ell(k,i)$ rows, which have one $E_i x_k$ term for each $k \neq i$. Thus

$$w_i = -\frac{1}{N}\left(-(N-1)x_i + \sum_{k\neq i} x_k\right) = -\frac{1}{N}\left(-Nx_i + \sum_{k=1}^{N} x_k\right) = x_i - \bar{x},$$

where $\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$. Thus, for a test point x,

$$\arg\max_i w_i^\mathsf{T} x = \arg\max_i x_i^\mathsf{T} x - \bar{x}^\mathsf{T} x = \arg\max_i x_i^\mathsf{T} x.$$

Because the x_i are orthonormal, this is further equal to

$$\arg\min_i \|x_i\|^2 + \|x\|^2 - 2x_i^\mathsf{T} x = \arg\min_i \|x - x_i\|. \qquad \square$$

Lemma 1. *If $X \mid Y = y \sim \mathcal{N}(\mu_y, \sigma^2 I)$ and $Y \sim \text{Uniform}([N])$, the Bayes-optimal classifier is given by*

$$f^*(x) = \arg\min_y \|x - \mu_y\|.$$

Proof. This well-known fact follows by combining

$$p(Y = y \mid X = x) = \frac{p(X = x \mid Y = y)p(Y = y)}{p(X = x)} \propto p(X = x \mid Y = y)$$

with the definition of the density for X,

$$\arg\max_y \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\|x - \mu_y\|^2\right) = \arg\min_y \|x - \mu_y\|. \qquad \square$$

B Implementation Details for Meta Learning Algorithms

Metric-based We use a meta learning library called learn2learn [5] to implement **ProtoNet** [61]. Following the original paper, we train a model with 30-way and 20-way for 1-Shot and 5-Shot, respectively, for 3,000 iterations. We use a 4 layer convolutional neural network (Conv4) with 64 channel size, and the batch size is set to 100. For optimization, we employ an Adam optimizer with a learning rate of 0.01 without having a learning rate schedule.

Optimization-based We use learn2learn library to implement both **MAML** [17] and **ANIL** [51]. We use Conv4 with 32 channel size for MAML and 64 channel size for ANIL (larger channel size does not perform better for MAML). We train both MAML and ANIL for 60,000 iterations. For optimizer, we employ an Adam optimizer for both with learning rates of 0.003 and 0.001 (adaptation

learning rates of 0.5 and 0.1) for MAML and ANIL, respectively. Batch sizes are set to 32 for both.

For **MetaOptNet** [40], we use the publicly available code provided by the authors of the paper (`https://github.com/kjunelee/MetaOptNet`). We employ the dual formulation of Support Vector Machine (SVM) proposed in MetaOptNet (MetaOptNet-SVM) for experiments with the training shot of 15, and use the default hyperparameter settings. For instance, we use a SGD optimizer with initial learning rate of 0.1 which decays step-wise. We train a model for 60 epochs with a batch size of 8.

Model-based For both Conditional Neural Process (CNP) [20] and Attentive Neural Process (ANP) [35], we use the publicly available code provided by the authors of the paper that addresses regression tasks for computer vision problems [19] (`https://github.com/boschresearch/what-matters-for-meta-learning`).

As the authors provide the model checkpoints for CNP on Distractor dataset and ANP on ShapeNet1D, we utilize them to compare active learning methods in meta-test time. We use 2-Shot for context sets in meta-test time instead of 25-Shot as done in the original work, since 25-Shot is too large to investigate the difference between active learning methods.

Pre-training-based We use the publicly available code provided by the authors of the papers for both **Baseline++** [12] (`https://github.com/wyharveychen/CloserLookFewShot`) and **SimpleShot** [68] (`https://github.com/mileyan/simple_shot`). For both models, we use the features from the pre-trained models on the whole training dataset in inference time. As reported in the public repository for Baseline++, the performance on CUB for 1-Shot and 5-Shot is lower than the numbers reported in the paper by about 1.1% and 2.5%, respectively. Similarly, the reproduced performance of SimpleShot for 1-Shot and 5-Shot is lower by about $4 \sim 5\%$. Note that the numbers correspond for the case of fully stratified random sampling.

C Relationship with Semi-Supervised Few-Shot learning

In this section, we describe the relationship between active meta-learning and semi-supervised few-shot learning [53, 69].

Both semi-supervised few-shot learning and active meta-learning aim to reduce the cost of manual data annotation, but they approach this goal differently. Semi-supervised few-shot learning leverages unlabeled data points without additional annotation, while active meta-learning iteratively adds new labeled data points selected from an unlabeled pool.

Among semi-supervised few-shot learning approaches, pseudo-labeling [31] is particularly closely related to active learning. Both pseudo-labeling and active learning utilize unlabeled data, but their methodologies differ. Active learning

uses uncertainty or diversity to select data for oracle labeling, targeting points whose labels are unknown. In contrast, pseudo-labeling uses a trained model to predict data labels, which can introduce errors if predictions are incorrect. Thus, pseudo-labeling focuses on data points where the model is already confident—precisely the points active learning would not select.

Combining these contrasting methods could be beneficial and interesting. Pseudo-labeling requires a well-trained classifier, which active learning can support by providing a robust labeled dataset.

D Implementation Details for Active Learning Strategies

In this section, we provide detailed description for the implementation of the following active learning methods.

DPP [8] We use DPPy library [23] to implement DPP selection. Gram matrix of the features from the penultimate layer are used as L-ensembles for DPP. We employ k-DPP to select k number of context data points.

Coreset [57] We refer to both original code and code provided by the authors of Typiclust and ProbCover. Since we assume that there is no initial labeled data points, we randomly choose the first data point and then apply the greedy algorithm after that.

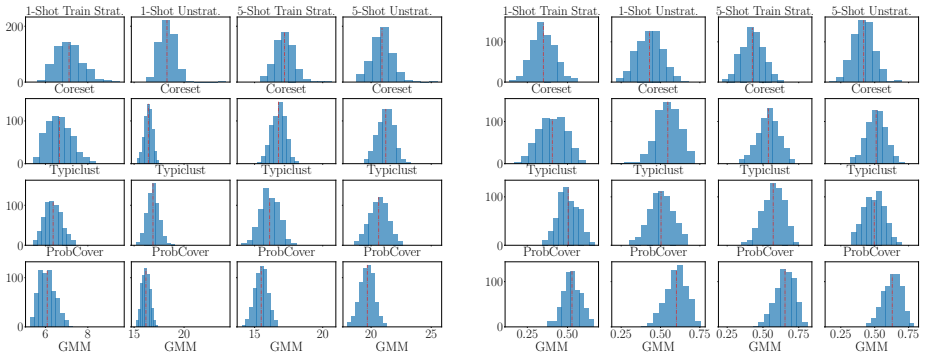
Typiclust [27] We refer to the publicly available code provided by the authors of the paper (`https://github.com/avihu111/TypiClust`). As the maximum number of data points to annotate is 25 (= 5-Way $\times$ 5-Shot), we do not set the maximum number of clusters unlike the original paper. We set the k in k-NN to 20 as with the original work.

ProbCover [70] We use the code provided by the original authors of the paper (it is the same as Typiclust). As we state in Appendix G and Appendix J, we exploit the features from the meta learners instead of self-supervised features to determine the radius parameters of ProbCover. In particular, the radius for each algorithm and dataset combination is determined as shown in Appendix G.

GMM (Ours) We refer to a publicly available implementation for GMM (`https://github.com/ldeecke/gmm-torch`). As previously mentioned, we initialize the cluster centers using k-means. Then, we update the cluster means and covariance matrix (shared by all the clusters) using expectation maximization algorithm for up to 100 iterations. We make the covariance matrix shared between the clusters because we assume that the "influence" of each annotated data point to other data points is roughly the same regardless of data point although the weight of each dimension may be different (if they are the same, it is equivalent to k-means).

E Comparison of quality of selected data points

In this section, we estimate the quality of selected data points from the low budget active learning methods. In Fig. 5, we compare them in the distance and



(a) Distributions of distances **(b)** Distributions of accuracies

Fig. 5: Estimation of goodness of selected data points on MiniImageNet with ANIL using the distribution of (a) the distance between the unlabeled points and closest selected points, and (b) the equality between the true labels of unlabeled points and labels of the closest select points. Red dotted lines show mean values.

Data & Model	Clustering	1-Shot			5-Shot		
		Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
MiniImage. MAML	k-means	56.75 ± 0.20	**33.29 ± 0.26**	37.26 ± 0.18	65.76 ± 0.18	41.61 ± 0.24	**59.17 ± 0.20**
	k-means^{++}	56.12 ± 0.26	32.87 ± 0.32	**38.53 ± 0.21**	65.49 ± 0.21	43.61 ± 0.32	58.63 ± 0.26
	GMM	**58.82 ± 0.24**	**33.34 ± 0.24**	37.68 ± 0.19	**67.18 ± 0.18**	**54.35 ± 0.20**	**59.05 ± 0.20**
FC100 ProtoNet	k-means	**50.20 ± 0.17**	29.69 ± 0.20	**35.03 ± 0.23**	54.07 ± 0.17	41.42 ± 0.23	41.34 ± 0.23
	k-means^{++}	49.91 ± 0.17	27.27 ± 0.22	**34.93 ± 0.27**	**54.72 ± 0.30**	41.61 ± 0.39	42.64 ± 0.39
	GMM	**50.22 ± 0.18**	**34.23 ± 0.23**	**35.03 ± 0.23**	**54.76 ± 0.17**	**46.30 ± 0.21**	**47.03 ± 0.20**

Table 7: Comparison of GMM and k-Means selections on MiniImageNet and FC100 using MAML and ProtoNet.

accuracy as explained in the caption with ANIL [51] on MiniImageNet. Whether a task is 1-Shot or 5-Shot, or train-time stratified or unstratified, we can observe that the metrics for GMM are consistently the best.

F Comparison to k-Means based methods

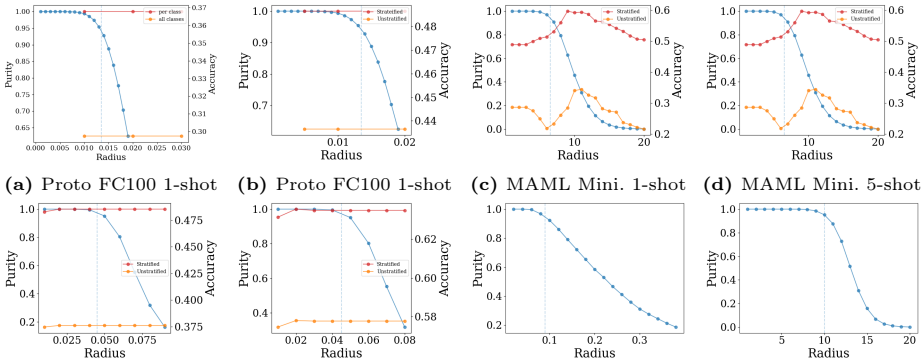
Tab. 7 compares the proposed GMM method to k-means and k-means^{++} since they are closely related.

The performance of GMM, k-means and k-means^{++} are similar in general but for some cases, GMM is significantly better than the others. We conjecture it is because some features are more important than the others, and since GMM takes it into account using Mahalanobis distance (instead of Euclidean distance used in k-means), it selects data points that represents nearby data points better.

G Difficulty of Tuning δ Parameter for ProbCover

In Section 3.2 of Yehuda *et al.* [70], the authors proposed to tune the radius δ based on the purity defined as,

$$\pi(\delta) = P(\{x : B_\delta(x) \text{ is pure}\}) \quad \text{where} \quad B_\delta(x) = \{x' : \|x' - x\|_2 \leq \delta\} \tag{9}$$



(a) Proto FC100 1-shot **(b)** Proto FC100 1-shot **(c)** MAML Mini. 1-shot **(d)** MAML Mini. 5-shot

(e) ANIL Tiered. 1-shot **(f)** ANIL Tiered. 5-shot **(g)** Baseline++ CUB **(h)** SimpleShot Mini.

Fig. 6: Estimation of the optimal radius for ProbCover in meta-learning

Here, a ball $B_\delta(x)$ is "pure" if $f(x') = y$, $\forall x' \in B_\delta(x)$ where y is the label of x. As the radius δ increases, the purity decreases monotonically. They choose the optimal radius δ^* as $\delta^* = \max\{\delta : \pi(\delta) \geq 0.95\}$. More specifically, they first run k-means with k being the number of classes. Then, the purity is measured using the k-means assignment as pseudo-labels.

In their setting (pool-based active learning for image classification), since it is hard to obtain meaningful features from a model trained only a few examples, they use the features from self-supervised learning methods such as SimCLR [11] It is, however, not the case for meta-learning. In meta-test time, the features from the meta learner are usually more meaningful than self-supervised learning features. Hence, we use the mete learner's features to estimate the optimal radius for ProbCover. Following the original paper, we first run k-means and compute the purity in the same way. Since the features can differ by meta learning algorithms and the number of shots, we provide the plots for different algorithms as well as 1 and 5-Shots as shown in Fig. 6 (we select the optimal radius δ based on these plots throughout the experiments). For Fig. 6(a)-(f), we also provide the meta-test performance of stratified and unstratified versions of Random selection to demonstrate that the estimated optimal radius and best radius for meta-test accuracy do not align.

Another difficulty of estimating the optimal radius is that it is hard to set a search space for the radius. As shown in the x-axis of Fig. 6, the reasonable search space varies significantly depending on the meta-learning algorithms and datasets we use. In Yehuda *et al.* , this was less of a problem since they use SimCLR features, which are normalized: the range of the radius is in $[0, 1]$. However, as shown in Appendix J, if we use SimCLR features in meta-test time to actively select context sets, the performance generally drops.

H Additional Experimental Results for Classification

In this section, we provide addtional experimental results for few-shot image classification. In Tab. 8, we compare the active learning strategies for ANIL [51]

$\text{Pick}_\theta^{eval}$	1-Shot			5-Shot		
	Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Random	47.55 ± 0.18	38.19 ± 0.16	34.79 ± 0.15	63.84 ± 0.17	57.92 ± 0.23	57.56 ± 0.18
Entropy	43.89 ± 0.16	32.73 ± 0.16	26.33 ± 0.14	57.56 ± 0.18	40.23 ± 0.18	34.16 ± 0.17
Margin	47.35 ± 0.17	36.01 ± 0.14	30.79 ± 0.14	62.87 ± 0.17	54.89 ± 0.24	56.76 ± 0.17
DPP	49.28 ± 0.17	38.17 ± 0.15	36.52 ± 0.15	63.24 ± 0.19	57.28 ± 0.21	57.23 ± 0.18
Coreset	47.32 ± 0.18	36.97 ± 0.20	40.72 ± 0.14	56.93 ± 0.18	47.68 ± 0.22	52.89 ± 0.17
Typiclust	52.95 ± 0.18	37.21 ± 0.17	34.05 ± 0.14	63.13 ± 0.19	55.84 ± 0.22	56.76 ± 0.17
ProbCover	48.53 ± 0.53	37.61 ± 0.49	34.53 ± 0.43	63.48 ± 0.51	57.77 ± 0.56	57.12 ± 0.58
GMM (Ours)	60.29 ± 0.19	50.92 ± 0.22	42.17 ± 0.17	66.48 ± 0.18	60.12 ± 0.24	60.28 ± 0.17

Table 8: 5-Way K-Shot on TieredImageNet with ANIL, with $\text{Pick}_\theta^{train}$ random.

$\text{Pick}_\theta^{eval}$	1-Shot			5-Shot		
	Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Random	40.41 ± 0.74	31.96 ± 0.56	32.76 ± 0.63	53.11 ± 0.66	47.73 ± 0.70	47.48 ± 0.76
DPP	40.47 ± 0.80	30.33 ± 0.67	33.41 ± 0.66	51.44 ± 0.68	48.21 ± 0.67	47.45 ± 0.68
Coreset	39.20 ± 0.71	27.55 ± 0.66	30.16 ± 0.69	46.80 ± 0.67	24.08 ± 0.65	25.75 ± 0.72
Typiclust	45.20 ± 0.78	26.35 ± 0.47	27.00 ± 0.43	52.39 ± 0.66	23.97 ± 0.42	24.12 ± 0.39
ProbCover	41.93 ± 0.67	26.87 ± 0.62	27.43 ± 0.48	54.36 ± 0.76	37.00 ± 0.69	38.33 ± 0.76
GMM (Ours)	51.16 ± 0.67	40.89 ± 0.74	41.61 ± 0.87	60.48 ± 0.86	52.68 ± 0.70	51.79 ± 0.70

Table 9: 5-Way K-Shot on FC100 with MetaOptNet, with $\text{Pick}_\theta^{train}$ random.

$\text{Pick}_\theta^{eval}$	1-Shot		5-Shot	
	Fully strat.	Train strat.	Fully strat.	Train strat.
Random	45.15 ± 0.73	26.28 ± 0.61	61.22 ± 0.72	51.89 ± 0.73
Entropy	37.08 ± 0.75	21.62 ± 0.37	47.93 ± 0.74	32.74 ± 0.60
Margin	41.53 ± 0.73	24.28 ± 0.51	62.15 ± 0.70	50.90 ± 0.75
DPP	44.52 ± 0.75	26.32 ± 0.58	60.93 ± 0.72	51.79 ± 0.75
Coreset	45.85 ± 0.73	27.04 ± 0.54	56.48 ± 0.72	40.39 ± 0.68
Typiclust	44.53 ± 0.71	22.97 ± 0.42	34.21 ± 0.77	20.04 ± 0.06
ProbCover	49.32 ± 0.71	24.61 ± 0.52	55.60 ± 0.66	32.24 ± 0.67
GMM (Ours)	52.77 ± 0.72	28.17 ± 0.64	62.64 ± 0.71	50.40 ± 0.75

Table 10: 5-Way K-Shot on MiniImageNet with SimpleShot, with $\text{Pick}_\theta^{train}$ random.

on the TieredImageNet dataset. Similarly, Tab. 9 provides the results with MetaOptNet [40] on FC100 dataset. Tab. 10, Tab. 11, and Tab. 12 are for SimpleShot [68], ProtoNet [61], and ANIL [51] on MiniImageNet, respectively. Note that Entropy and Margin selections are not applicable for MetaOptNet-SVM. Regardless of meta-learning algorithm and dataset, GMM significantly outperforms the other active learning methods, and some of them are worse than the Random selection.

I Additional Experimental Details for Regression

Gao *et al.* propose the Distractor and ShapeNet1D datasets to compare meta learning algorithms for vision regression tasks. They evaluate meta learners for

$\text{Pick}_\theta^{eval}$	1-Shot			5-Shot		
	Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Random	47.70 ± 0.20	39.65 ± 0.28	38.72 ± 0.27	64.66 ± 0.18	57.36 ± 0.27	57.42 ± 0.25
Entropy	44.33 ± 0.20	36.35 ± 0.28	34.87 ± 0.27	61.23 ± 0.19	49.83 ± 0.31	48.46 ± 0.32
Margin	47.07 ± 0.20	37.69 ± 0.27	37.84 ± 0.28	63.79 ± 0.18	55.25 ± 0.29	56.15 ± 0.27
DPP	47.90 ± 0.20	39.17 ± 0.28	37.89 ± 0.26	64.36 ± 0.19	57.48 ± 0.26	57.37 ± 0.25
Coreset	47.86 ± 0.20	39.51 ± 0.26	37.79 ± 0.26	55.09 ± 0.20	50.14 ± 0.29	50.27 ± 0.28
Typiclust	59.51 ± 0.17	38.47 ± 0.27	37.57 ± 0.27	61.02 ± 0.19	51.82 ± 0.31	52.02 ± 0.30
ProbCover	48.51 ± 0.20	35.25 ± 0.26	34.50 ± 0.25	43.61 ± 0.19	38.63 ± 0.21	38.24 ± 0.20
GMM (Ours)	64.50 ± 0.16	47.88 ± 0.32	44.71 ± 0.29	67.03 ± 0.19	57.55 ± 0.29	56.44 ± 0.30

Table 11: 5-Way K-Shot on MiniImageNet with ProtoNet, with $\text{Pick}_\theta^{train}$ random.

$\text{Pick}_\theta^{eval}$	1-Shot			5-Shot		
	Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Random	46.59 ± 0.19	36.70 ± 0.19	34.79 ± 0.18	61.35 ± 0.19	55.24 ± 0.20	56.65 ± 0.19
Entropy	44.63 ± 0.20	35.51 ± 0.18	27.35 ± 0.14	55.09 ± 0.19	39.71 ± 0.20	37.45 ± 0.19
Margin	46.58 ± 0.19	36.60 ± 0.19	32.46 ± 0.18	55.62 ± 0.19	40.40 ± 0.20	37.67 ± 0.19
DPP	47.33 ± 0.19	37.45 ± 0.17	37.76 ± 0.18	61.08 ± 0.19	56.18 ± 0.18	57.08 ± 0.18
Coreset	46.40 ± 0.21	38.37 ± 0.17	41.34 ± 0.17	53.74 ± 0.20	47.81 ± 0.20	51.62 ± 0.19
Typiclust	54.44 ± 0.18	36.78 ± 0.17	34.52 ± 0.19	60.87 ± 0.18	52.56 ± 0.20	55.11 ± 0.19
ProbCover	51.56 ± 0.18	27.49 ± 0.15	41.46 ± 0.17	61.68 ± 0.18	53.80 ± 0.20	42.70 ± 0.22
GMM (Ours)	58.50 ± 0.18	48.13 ± 0.20	40.26 ± 0.18	65.14 ± 0.17	59.01 ± 0.20	61.48 ± 0.19

Table 12: 5-Way K-Shot on MiniImageNet with ANIL, with $\text{Pick}_\theta^{train}$ random.

intra-category (IC) and cross-category (CC) inputs where CC corresponds to the cross-domain in few-shot image classification.

Distractor consists of 10 object classes for a training set and 2 novel classes for CC evaluation. Each class contains $1,000$ randomly sampled objects from ShapeNetCoreV2 [10]. 20% of training set is reserved for IC evaluation. In this dataset, each image consists of two objects: the object of interest and a distractor object, which are positioned randomly. The goal is to recognize and locate the object of interest within the image in the presence of a distractor.

ShapeNet1D [19] consists of 27 object classes for a training set and 3 object classes for CC evaluation. Each object class contains 50 images, and 10 images are used for IC evaluation. ShapeNet1D aims to predict the 1D pose, i.e., rotation angle, around the azimuth axis of an object.

To analyze these vision regression tasks, we compare various active learning strategies in the 2-shot setting. We use CNP for Distractor, NP for ShapeNet1D. More details about the models can be found in Appendix B.

J Comparison to Self-Supervised Features

ProbCover and Typiclust use self-supervised features to actively select new data points to annotate, since there are not enough labeled data to train a classifier to output meaningful features. Instead, they utilize the features from SimCLR [11]. To validate if it is better to use the features from a meta learner than SimCLR in

Dataset	Features	1-Shot			5-Shot		
		Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Mini.	MAML	**55.65 ± 0.18**	27.45 ± 0.17	**35.46 ± 0.18**	64.16 ± 0.18	**46.70 ± 0.21**	**57.83 ± 0.21**
	SimCLR	44.84 ± 0.44	**27.59 ± 0.35**	34.80 ± 0.47	**65.95 ± 0.43**	36.03 ± 0.48	57.77 ± 0.47
FC100	ProtoNet	**46.01 ± 0.16**	**30.96 ± 0.19**	**30.61 ± 0.21**	47.54 ± 0.17	**43.61 ± 0.18**	**44.03 ± 0.21**
	SimCLR	36.07 ± 0.44	29.60 ± 0.46	30.13 ± 0.45	**48.59 ± 0.49**	43.29 ± 0.49	43.89 ± 0.59

Table 13: Comparison of MAML and SimCLR features for Typiclust.

Dataset	Features	1-Shot			5-Shot		
		Fully strat.	Train strat.	Unstrat.	Fully strat.	Train strat.	Unstrat.
Mini.	MAML	**52.81 ± 1.16**	21.91 ± 0.24	**36.21 ± 0.18**	**64.70 ± 0.91**	**42.07 ± 0.49**	23.40 ± 0.36
	SimCLR	47.57 ± 0.42	**25.35 ± 0.38**	32.19 ± 0.43	64.33 ± 0.39	36.64 ± 0.58	**26.16 ± 0.43**
FC100	ProtoNet	**48.66 ± 0.16**	**32.86 ± 0.22**	**33.58 ± 0.19**	**51.11 ± 0.17**	**44.20 ± 0.24**	44.40 ± 0.24
	SimCLR	31.40 ± 0.42	29.53 ± 0.42	28.39 ± 0.43	47.11 ± 0.39	39.33 ± 0.54	**45.40 ± 0.52**

Table 14: Comparison of MAML and SimCLR features for ProbCover.

meta-learning, we compare SimCLR features to the features from either MAML or ProtoNet for Typiclust and ProbCover as shown in Tab. 13 and Tab. 14. Here, we use MiniImageNet and FC100 datasets for MAML and ProtoNet, respecitvely as with Tab. 2 and Tab. 1. For both Typiclust and ProbCover, although there are a couple of cases where SimCLR features are better, it is significantly worse than MAML and ProtoNet features in general. It intuitively makes sense because 1) meta learners are trained with large enough data points and 2) it is likely that the information in self-supervised features do not align with that in meta learners.

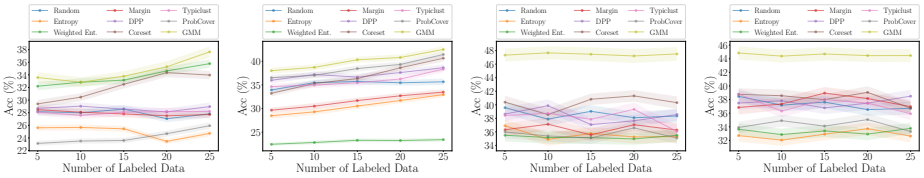
K Sequential Active-Meta Learning

Although iterative sampling is more common in active learning, we have focused on sampling a context set at once because of the following two reasons.

First, even though we iterative label additional samples, the features do not change in most of meta-learning algorithms except for MAML. Even for other optimization-based methods such as ANIL, since the feature extractor is not updated during adaptation on a context set, the features will stay the same for iterative process of active learning. As we demonstrated with ProtoNet in Fig. 7 (c)-(d) (details about experiments are below), although we iteratively add more labeled samples, the performance does not change much as the features do not change. In this case, selecting $N \times K$ samples at once is not different from iterative process while it is cheaper.

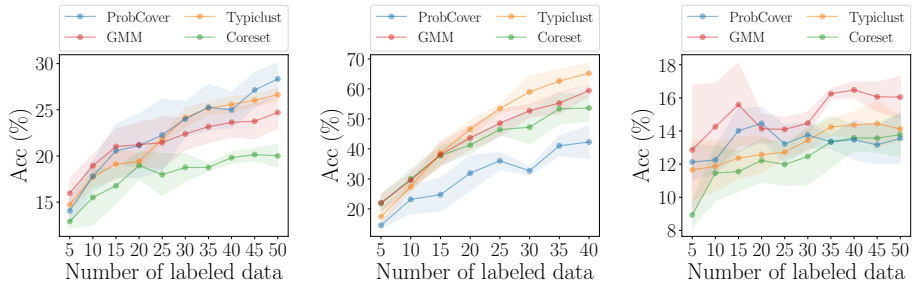
Furthermore, if we iteratively add labeled samples, it will quickly go beyond few-shot regime in meta-learning, which is often not that practical in real world settings. Suppose we have a meta learner trained in 5-way 1-Shot. It is reasonable to add 5 samples per iteration since it is the minimum number to cover all the classes. But only after 5 iterations, it will go few-shot regime where we typically have 25 labeled context samples. It is even less practical for 5-Shot case.

Fig. 7 compare active learning methods for sequential setting where we select 5 context samples at a time until the budget reaches 25 samples. Every



(a) MAML: Train strat. **(b)** MAML: Unstrat. **(c)** Proto: Train strat. **(d)** Proto: Unstrat.

Fig. 7: Test performance of MAML and ProtoNet on MiniImageNet with sequentially actively selected context sets. 5 context samples are selected at each iteration until it reaches 25.



(a) CIFAR10 **(b)** MNIST **(c)** SVHN

Fig. 8: Low-budget active learning methods on image classification with very low budget. Mean and standard error of accuracy for three sets of SimCLR features, three runs per features.

time we select new context samples we may utilize them to maximize new label information. For MAML, we update all the model parameters through adaptation steps. It is, however, not applicable to the other meta-learning methods we use in this work including ProtoNet, since none of the other methods including optimization-based methods such as ANIL, do not update the parameters up to the penultimate layer.

As expected, the test performance of ProtoNet does not change much regardless of active learning methods. But, the test performance of MAML gradually increases as we add more context samples. In sequential active-meta learning, GMM still significantly outperforms other active learning methods.

L Fitting GMM using Expectation Maximization

In this section, we provide details about fitting GMM using the expectation maximization (EM) algorithm. Although it is available in many literature, we add it here for completeness of our method. The log-likelihood objective for a

GMM is given by,

$$\ell(\theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right), \tag{10}$$

where model parameters $\theta = \{(\pi_k, \mu_k, \Sigma_k)\}_{k=1}^{K}$ with $N$and K being the number of samples and mixture components, respectively.

EM algorithm is an iterative algorithm where we alternatively conduct E-step and M-step as follows,

– E-step: we compute the posterior probability ω_{ik} that represents i-th data point belongs to the k-th Gaussian component as,

$$w_{ik} = \frac{\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)} \tag{11}$$

– M-step: we maximize the log-likelihood in terms of the model parameters. Fortunately, for GMM, there are closed form solutions for each parameter.

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} w_{ik}, \quad \mu_k = \frac{\sum_{i=1}^{N} w_{ik} x_i}{\sum_{i=1}^{N} w_{ik}}, \quad \Sigma_k = \frac{\sum_{i=1}^{N} w_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} w_{ik}} \tag{12}$$

We repeat the E-step and M-step until convergence of the log-likelihood or for a fixed number of iteration time. Please note that we use diagonal covariance Σ_k since it is computationally efficient and often fits better in term of log-likelihood.

M Analysis on Low Budget Active Learning Methods

We briefly explain why each active learning method does not perform as well as the proposed GMM method or even Random selection. In this section, we discuss further on the inferiority of low budget active learning methods compared to GMM. We conjecture it may be attributed to its implicit exploration of locally dense regions or inappropriate measure of representativeness. Here we analyzed potential reasons of their failure in very low budget regime.

– Typiclust: after conducting k-means, it selects samples for each cluster j, based on

$$\underset{x \in \text{clust}_j}{\arg\max} \left(\frac{1}{K} \sum_{x_i \in KNN(x)} ||x - x_i||_2 \right)^{-1}$$

where KNN denotes k-nearest neighbors of which size is fixed to 20. This measure seeks for locally dense region by selecting samples that are close to its nearest neighbors.

– ProbCover: it greedily finds the maximally covering samples given a fixed radius. This greedy algorithm provide $(1 - \frac{1}{e})$-approximation for the optimal solution but the gap with the optimal solution can be quite large. Also, the selection of the radius is hard as we discussed in Appendix G. When the radius is small, it tries to find samples that are in locally dense regions.
– DPP: it finds samples of which a kernel matrix (with a pre-defined kernel function) has the maximum determinant, which implicitly finds diverse samples. The determinant of a matrix, however, may not align with selecting maximum covering (or representative) samples. In particular, maximizing the determinant of the kernel matrix may lead to selecting samples far away from other samples.

Compared to these methods, GMM tries to find globally representative samples in non-greedy fashion (using expectation maximization). Also, its measure of covering other samples is in Mahalanobis distance, which intuitively makes more sense than the determinant of a kernel matrix as a measure. The proposed GMM method is also theoretically motivated by the Corollary 1, which says a classifier trained with the selected samples from GMM (cluster means) is a Bayes-optimal classifier under certain conditions.

N Training-Time Active Learning

As mentioned in Sec. 5, we observe that active learning methods do not significantly change the generalization performance of meta learners when applied in meta-train time, which aligns with observations from [46, 58]. To empirically demonstrate it, we apply several active learning methods without stratification in the meta-train time for ProtoNet on MiniImageNet. Fig. 9 shows among Random, DPP and GMM selections, one is not significantly better than another although the Entropy selection is significantly worse than them.

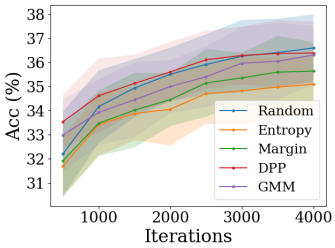


Fig. 9: Comparison of $\text{Pick}_{\theta}^{train}$.