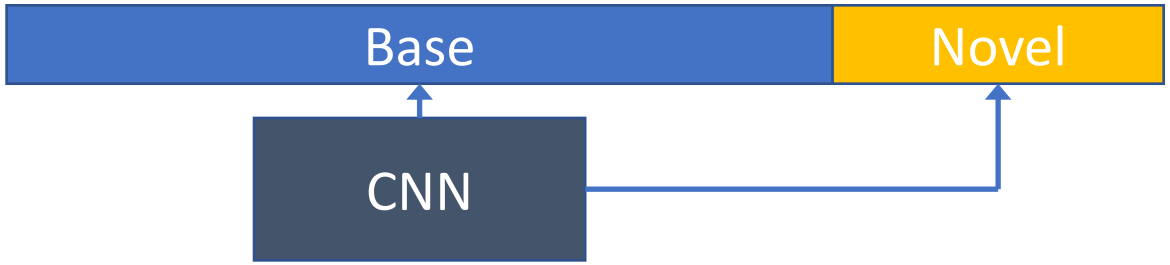
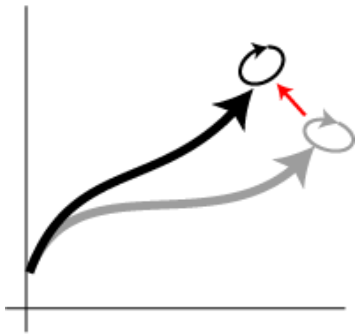
Incremental Few-Shot Learning with Attention Attractor Networks

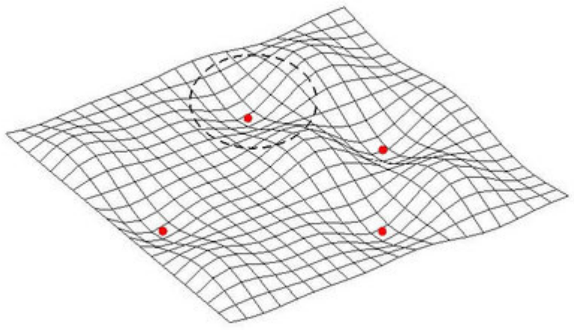
Mengye Ren, Renjie Liao, Ethan Fetaya, Richard S. Zemel

University of Toronto and Vector Institute

- Testing **on only new classes** in “few-shot” is not natural.
- **Incremental few-shot learning**: learn new classes on top of old classes. **No access to the old data**.
- At each test episode, learn a linear classifier **until convergence**.
- Attention over base classes to form **attractor regularizers**.
- At the end of the episode, test on a **query set of both base and novel**.
- Use **recurrent backprop (RBP)** instead of **truncated BPTT** for learning more **stable loss functions**.
- **Learned regularizers** significantly reduce class interference.



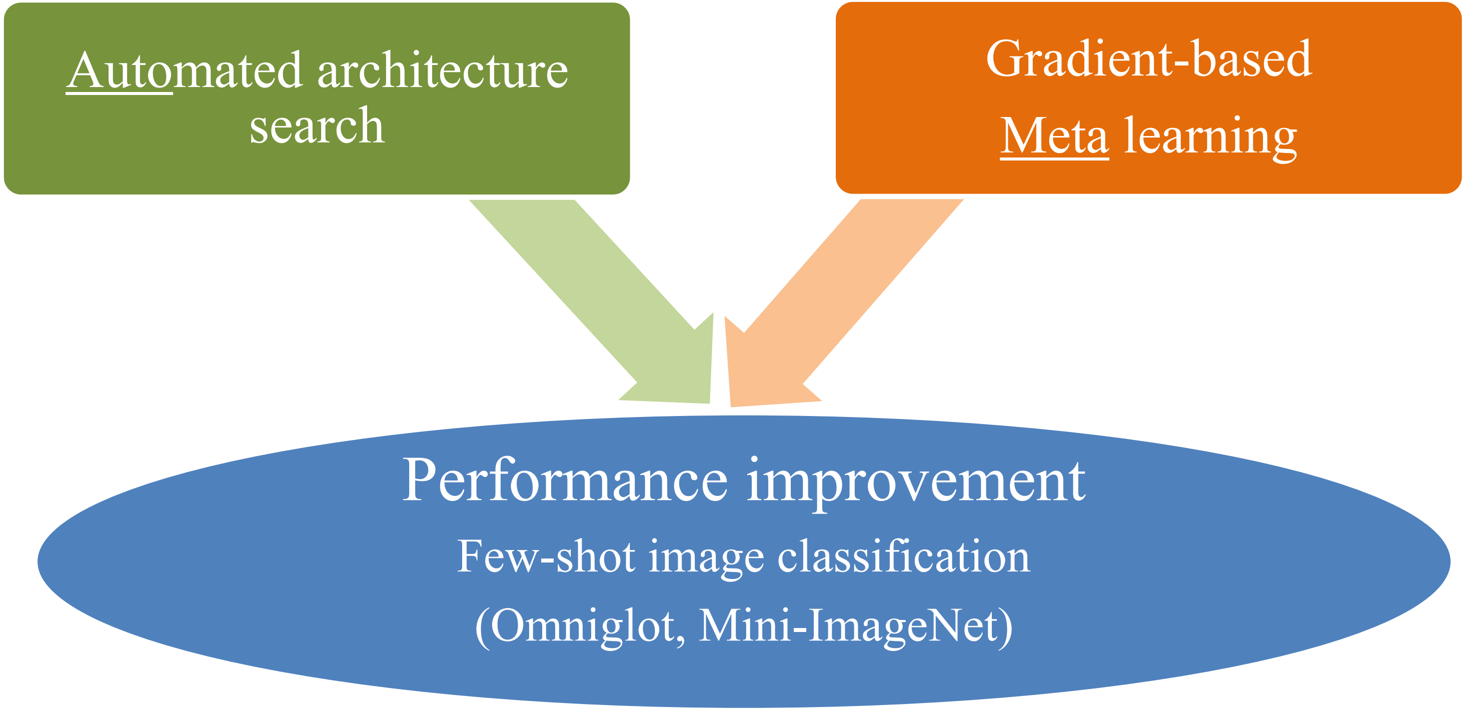




Auto-Meta: Automated Gradient Based Meta Learner Search

Jaehong Kim[1], Sangyeul Lee[1], Sungwan Kim[1], Moonsu Cha[1], Jung Kwon Lee[1], Youngduck Choi[1,2], Yongseok Choi[1], Dong-Yeon Cho[1], and Jiwon Kim[1]

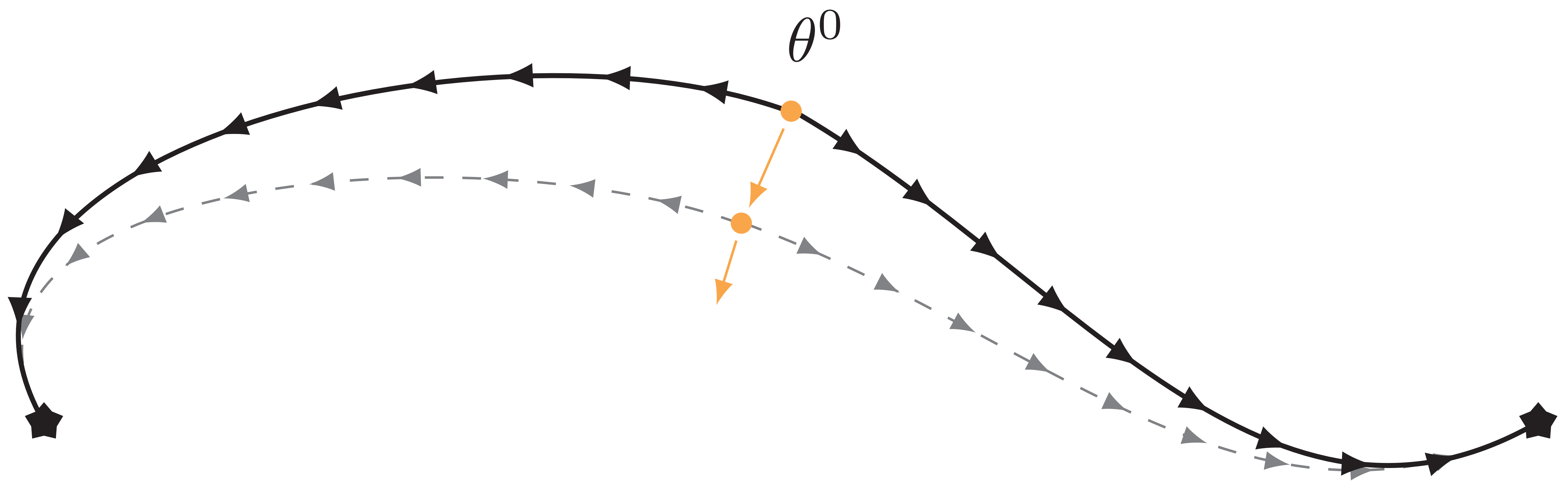
[1] SK T-Brain [2] Yale University



Transferring Knowledge across Learning Processes

Sebastian Flennerhag, Pablo G. Moreno, Neil D. Lawrence, Andreas Damianou

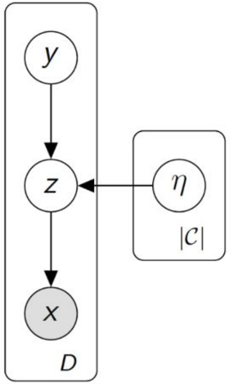
- We propose a framework for meta-learning across task geometries by learning from gradient trajectories
- We present *Leap*, a light-weight meta-learner that scales beyond few-shot learning to tasks requiring millions of gradient steps

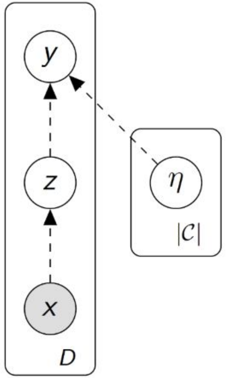


Few-shot Learning For Free by Modelling Global Class Structure

Xuechen Li*, Will Grathwohl*, Eleni Triantafillou*, David Duvenaud, Richard Zemel

- Most approaches to few-shot classification use **episodic training.**
- We advocate for a simpler approach: a generative model over **all classes:** a VAE with a **mixture of Gaussians prior**.
- Few-shot learning is done by **variational inference**.
- Our model solves 3 tasks:
  - Few-shot classification
  - Few-shot generation
  - More realistic: **Few-shot integration**.
- Omniglot experiments:
  - On par with state-of-the-art on few-shot classification.
  - Largely outperform our baseline on few-shot integration.





TAEML: Task-Adaptive Ensemble of Meta-Learners

Workshop on Meta-Learning (MetaLearn2018)

Minseop Park / mike_seop@aitrics.com

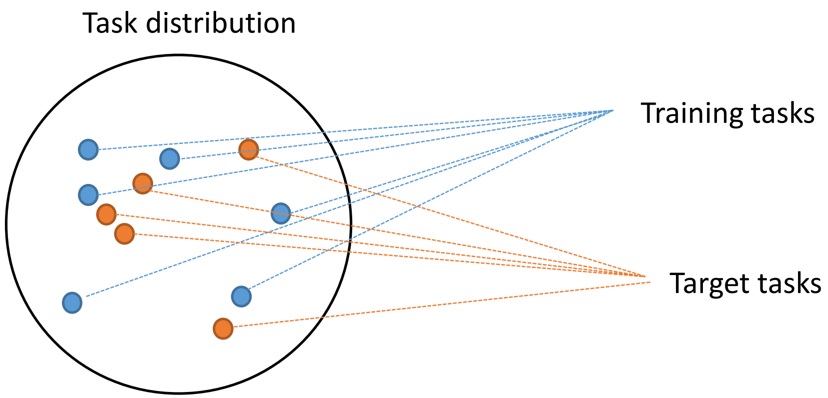


Fig1. Current meta-learning for few-shot classification

Fig2. Solving to few-shot classify the birds: Training all of the tasks won't be efficient

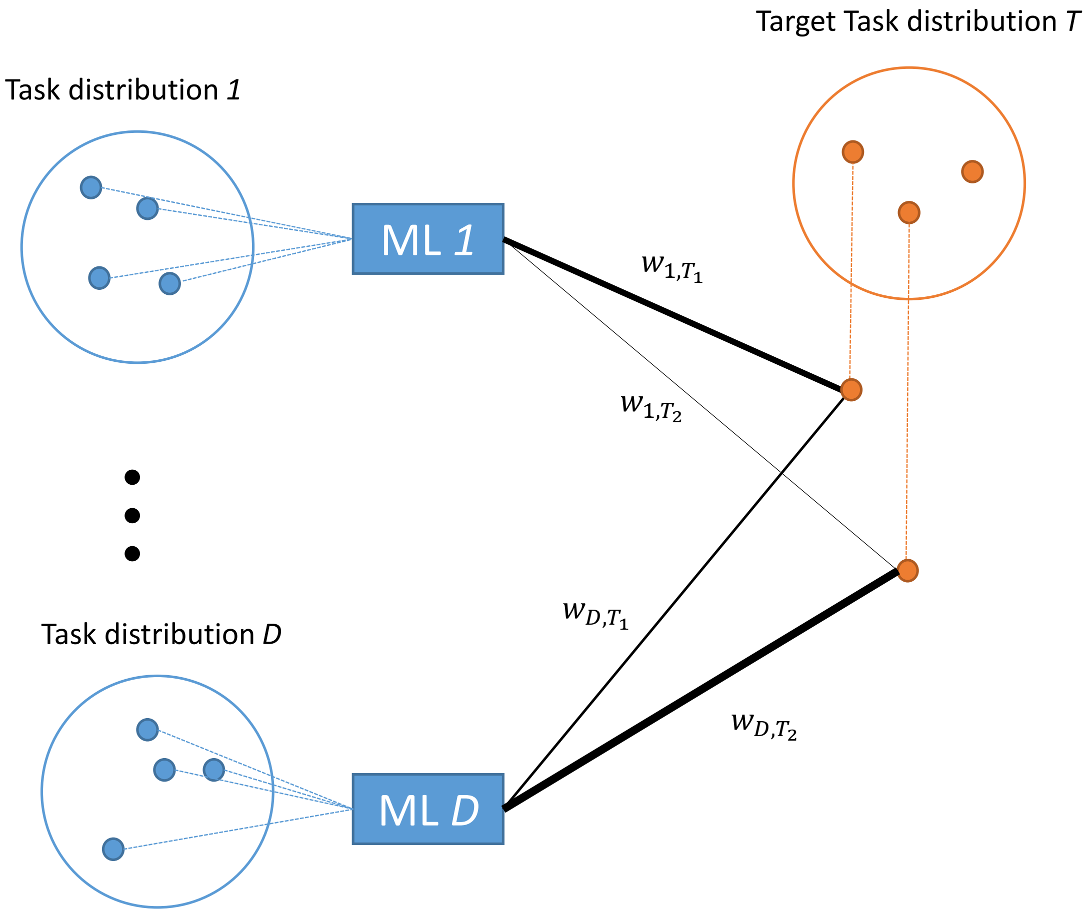


Fig3. Target task adaptive ensemble of pre-trained meta-learners

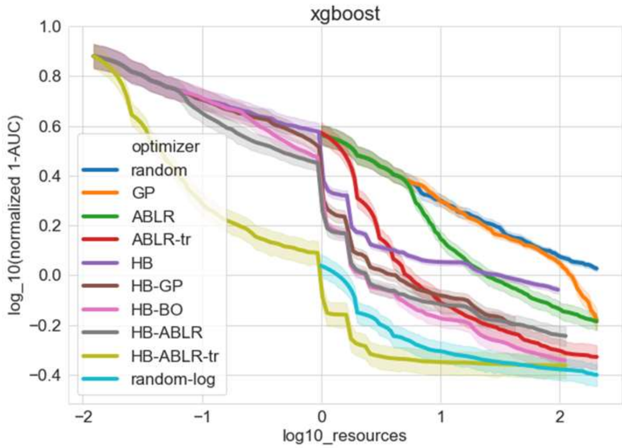
A Simple Transfer-Learning Extension of Hyperband

Lazar Valkov, Rodolphe Jenatton, Fela Winkelmolen, Cédric Archambeau

- Setting: Hyperparameter Optimisation
- Hyperband (HB):
  - Incrementally allocates more resources to the best-performing candidates initially taken from a pool of randomly sampled candidates.
  - Evaluates different number of initial candidates n_i for r_i
- We enhance HB with model-based sampling, using ABLR (Peronne *et al.*)

$$\boldsymbol{\Phi}_{z}(\boldsymbol{X_t}, \boldsymbol{r}_i)$$

$$P(\boldsymbol{y}_t | \, w_t, z, \beta_t) = \mathcal{N}\left(\boldsymbol{\Phi}_{z}(\boldsymbol{X_t}, \boldsymbol{r}_i)\boldsymbol{w_t}, {\beta_t}^{-1} I_{N_t}\right) P(\boldsymbol{w_t} | \alpha_t) = \mathcal{N}(\boldsymbol{0}, \alpha_t^{-1} I_D)$$



- Benefits:
  - Makes use of all data produced by a HB run
  - Can use data from past HB runs to learn better basis function
  - We don’t use heuristics for low number of data points, nor to encourage exploration

Learned optimizers that outperform SGD on wallclock and test loss

Google AI

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, C. Daniel Freeman, Jascha Sohl-Dickstein

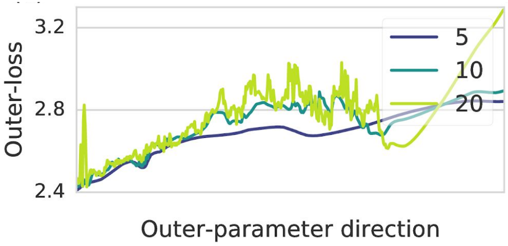
Existing optimizers are **hand designed.** Can we do better with **learning**?

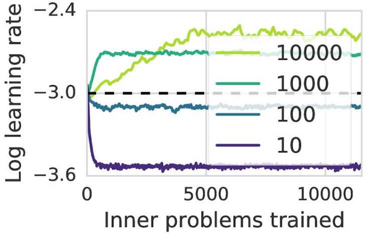
One popular strategy for training such optimizers is to leverage gradients and **truncated backpropagation through time.**

These methods, however, are notoriously **unstable**!

Careful choice of step length is required:

- Long truncations: **exploding gradients**
- Short truncations: **biased gradients**



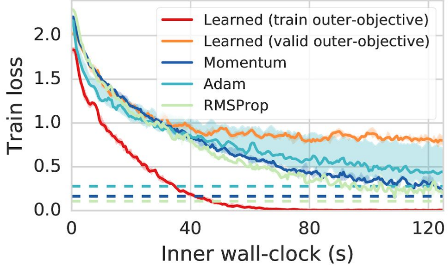


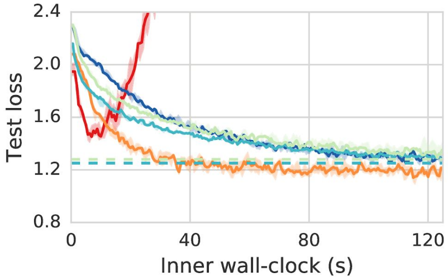
We use **variational optimization** to "smooth" the loss surface by convolving it with a Gaussian.

$$\mathcal{L}(\theta) = \mathbb{E}_{\tilde{\theta}\sim\mathcal{N}(\theta,\sigma^2 I)}\left[L\left(\tilde{\theta}\right)\right]$$

To optimize this objective, we combine **multiple gradient estimators** with difference variances.

We train **simple** MLP-based learned optimizers that are **faster in wallclock time** and **generalize better** than existing hand-designed methods.





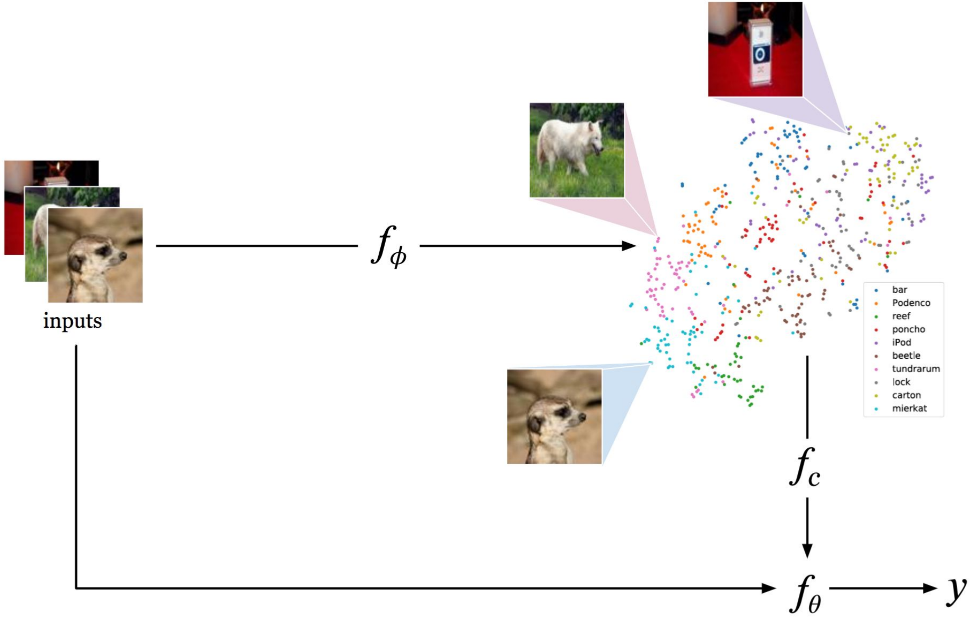
Learning to Learn with Conditional Class Dependencies

Xiang Jiang[1,2], Mohammad Havaei[1], Farshid Varno[1,2], Gabriel Chartrand[1], Nicolas Chapados[1], Stan Matwin[2]
[1] Imagia Inc. [2]Dalhousie University

Integrates two views of the data

The metric space captures class dependencies

Conditional batchnorm helps class separation



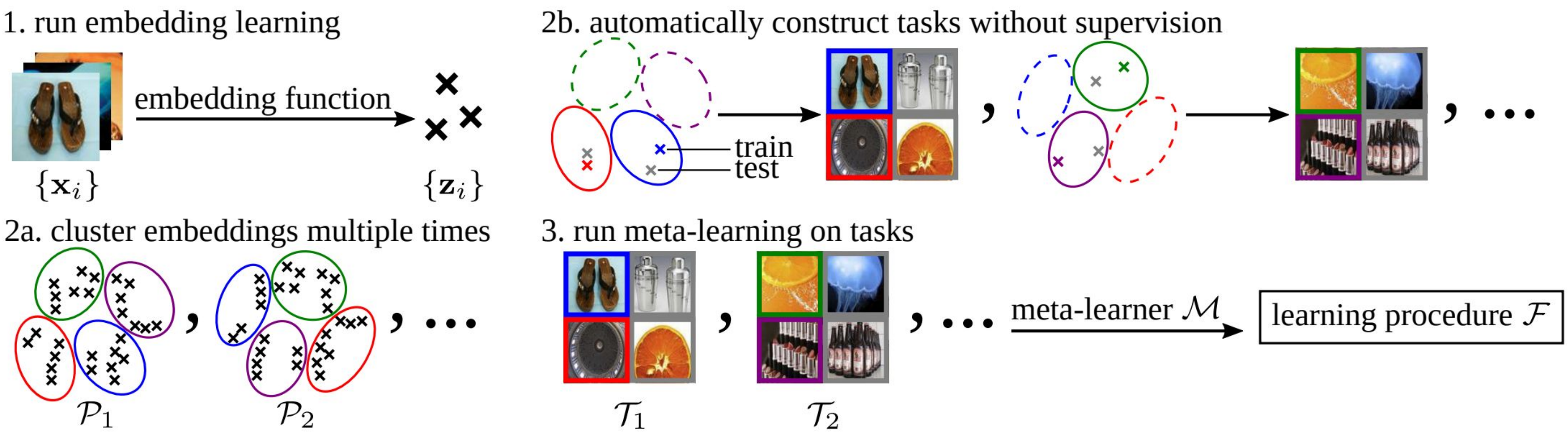
Unsupervised Learning via Meta-Learning

Division of Engineering Science
UNIVERSITY OF TORONTO

Kyle Hsu[1], Sergey Levine[2], Chelsea Finn[2]
[1]University of Toronto [2]UC Berkeley

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

- Unsupervised learning is commonly used as pre-training for downstream learning.
    - We improve upon this by incorporating knowledge about the downstream task type: image classification.
- **Unsupervised meta-learning** via CACTUs: meta-learning over tasks constructed from unlabeled data.



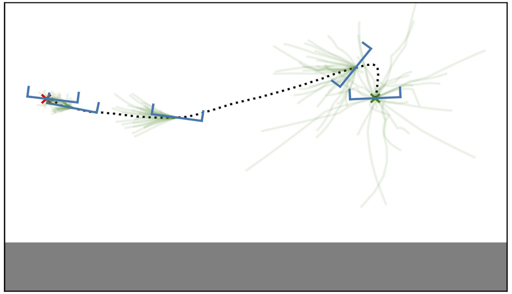
- Results: better than unsupervised learning, worse than supervised meta-learning

CAMeLiD: Control Adaptation via Meta-Learning Dynamics

James Harrison*[,1], Apoorva Sharma*[,1],
Roberto Calandra[2], Marco Pavone[1]

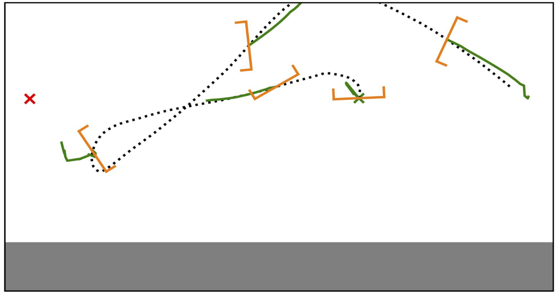
Autonomous Systems Lab

We develop a Bayesian meta-learning model that is capable of **fast, efficient online updates** and is trained for multi-step probabilistic predictions.



CAMeLiD controlling a quadrotor with a random attached mass. By incorporating model uncertainty into control, we successfully stabilize.

Using this model, we build a control algorithm that captures online model uncertainty and **automatically trades off safety and performance**.



Point estimate meta-learning-based control algorithm results in the quadrotor crashing.

[1]Stanford University, [2]UC Berkeley

{jharrison,apoorva,pavone}@stanford.edu, roberto.calandra@berkeley.edu

Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning

Anusha Nagabandi*, Ignasi Clavera*, Simin Liu,
Ron S. Fearing , Pieter Abbeel, Sergey Levine, Chelsea Finn

Goal

Use **recent experiences** to quickly **adapt** to the current situation.

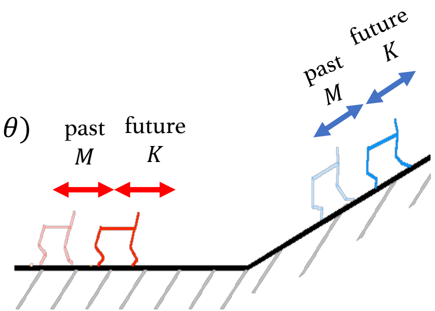
Train time: Learning to Adapt

Meta-learn a dynamics model
Tasks: temporal windows

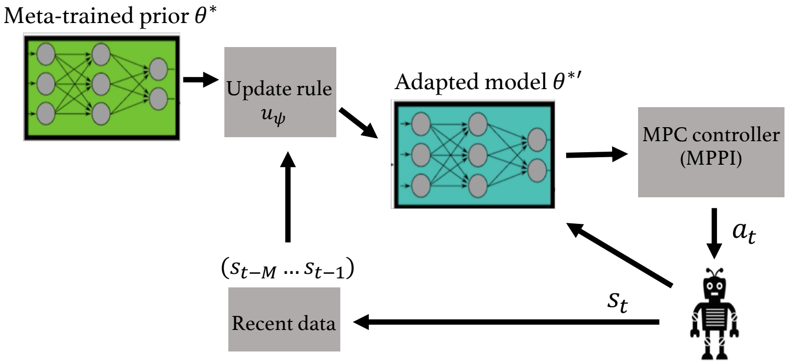
Objective:

$$\min_{\theta,\psi} E[L(D_T^{test},\theta')] \;\; s.t. \;\; \theta' = u_\psi(D_T^{tr},\theta)$$

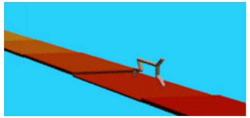
D_T^{test} → Future data
D_T^{tr} → Past data

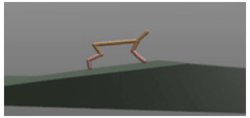


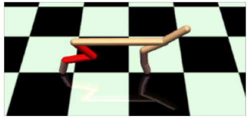
Test time: Meta-Model-Based RL

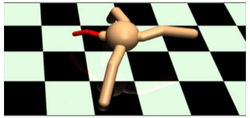


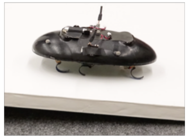
Experiments


Pier


Terrain slopes


Disabled


Crippled


Slope


Pose error

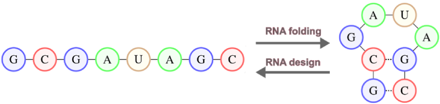

Payload


Missing leg

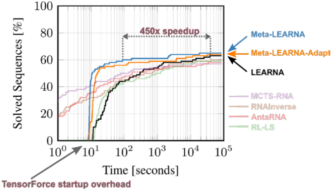
Learning to Design RNA

Frederic Runge* **Danny Stoll*** Stefan Falkner Frank Hutter

UNI FREIBURG



- **Meta-learn** a policy across RNA Design tasks
- **AutoML** for joint optimization of:
  - Policy network architecture
  - RL formulation
  - Training Hyperparameters
- **New state-of-the-art** on three benchmarks

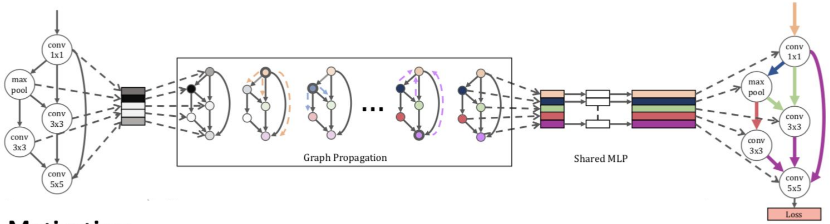


Graph HyperNetworks for Neural Architecture Search

Chris J. Zhang[1,2], Mengye Ren[1,3], Raquel Urtasun[1,3]

[1] Uber Advanced Technologies Group [2] University of Waterloo, [3] University of Toronto

Graph HyperNetworks



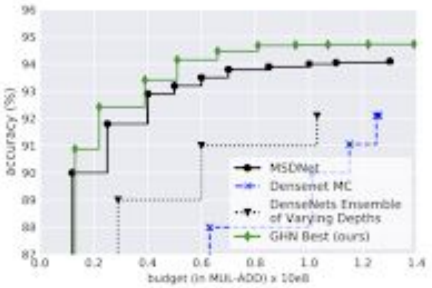
Motivation:

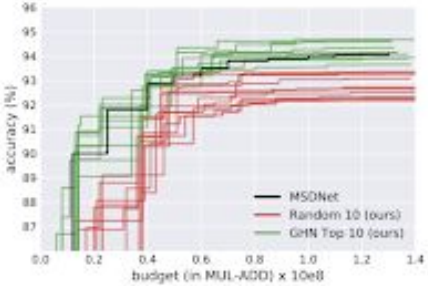
- Neural architecture search is an expensive nested optimization

$$a^* = \arg\min_a \mathcal{L}_{val}(w^*(a), a), \quad w^*(a) = \arg\min_w \mathcal{L}_{train}(w, a)$$

- Instead of using SGD to learn weights, use trained hypernetwork to generate weights
- Graph HyperNetworks (GHN) explicitly model the topology of architectures by learning on a computation graph representation

Anytime Prediction





NAS Benchmarks

CIFAR-10: Comparison with NAS methods which employ random search (top half) and advanced search methods (e.g. RL) (bottom half)

Method	Search Cost (GPU days)	Param $\times 10^6$	Accuracy
SMASHv1 (Brock et al., 2018)	?	4.6	94.5
SMASHv2 (Brock et al., 2018)	3	16.0	96.0
One-Shot Top (F=32) (Bender et al., 2018)	4	2.7 ± 0.3	95.5 ± 0.1
One-Shot Top (F=64) (Bender et al., 2018)	4	10.4 ± 1.0	95.9 ± 0.2
Random (F=32)	-	4.6 ± 0.6	94.6 ± 0.3
GHN Top (F=32)	0.42	5.1 ± 0.6	95.7 ± 0.1
NASNet-A (Zoph et al., 2018)	1800	3.3	97.35
ENAS Cell search (Pham et al., 2018)	0.45	4.6	97.11
DARTS (first order) (Liu et al., 2018b)	1.5	2.9	97.06
DARTS (second order) (Liu et al., 2018b)	4	3.4	97.17 ± 0.06
GHN Top-Best, 1K (F=32)	0.84	5.7	97.16 ± 0.07

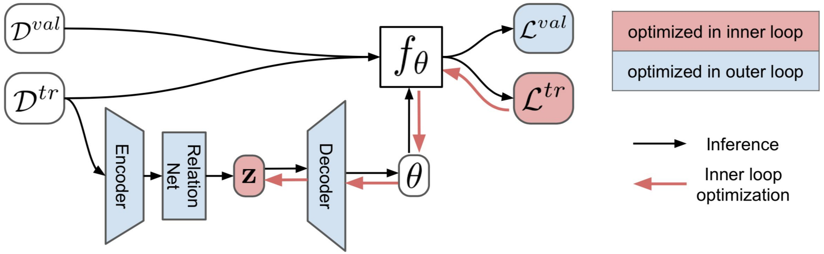
ImageNet Mobile: Comparison with NAS methods which employ advanced search methods (e.g. RL)

Method	Search Cost (GPU days)	Param $\times 10^6$	FLOPs $\times 10^6$	Accuracy Top 1	Accuracy Top 5
NASNet-A (Zoph et al., 2018)	1800	5.3	564	74.0	91.6
NASNet-C (Zoph et al., 2018)	1800	4.9	558	72.5	91.0
AmoebaNet-A (Real et al., 2018)	3150	5.1	555	74.5	92.0
AmoebaNet-C (Real et al., 2018)	3150	6.4	570	75.7	92.4
PNAS (Liu et al., 2018a)	225	5.1	588	74.2	91.9
DARTS (second order) (Liu et al., 2018b)	4	4.9	595	73.1	91.0
GHN Top-Best, 1K	0.84	6.1	569	73.0	91.3

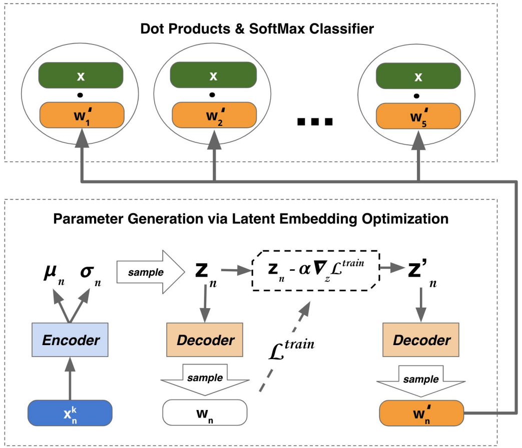
Meta-Learning with Latent Embedding Optimization (LEO)

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, Raia Hadsell

We learn a data-dependent latent generative representation of model parameters, and perform gradient-based meta-learning in this low dimensional latent space.



The resulting approach, Latent Embedding Optimization (LEO), decouples the gradient-based adaptation procedure from the underlying high-dimensional space of model parameters.



LEO is *state-of-the-art* on both *miniImageNet* and *tieredImageNet* 5-way 1-shot and 5-shot classification tasks.

DeepMind

We are in the process of open-sourcing our embeddings and code!

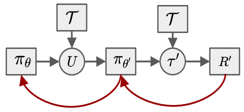
Proximal Meta-Policy Optimization: ProMP

Jonas Rothfuss*, Dennis Lee*, Ignasi Clavera*,
Tamim Asfour, and Pieter Abbeel

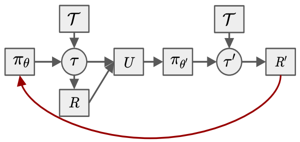
Goal

1. Analyze **credit assignment** in meta-reinforcement learning
2. Develop a **new objective** that trains for the pre-update sampling distribution

Credit Assignent Sampling Distribution



MAML



E-MAML

Low Variance Curvature Estimator (LVC)

$$J^{\mathrm{LVC}}(\boldsymbol{\tau}) = \sum_{t=0}^{H-1} \frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\perp(\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t))} \left(\sum_{t'=t}^{H-1} r(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'})\right) \qquad \boldsymbol{\tau} \sim P_{\mathcal{T}}(\boldsymbol{\tau})$$

- Meta-gradient with **low variance**
- **Unbiased** closed to local optima

Proximal Meta-Policy Optimization: ProMP

ProMP Objective:

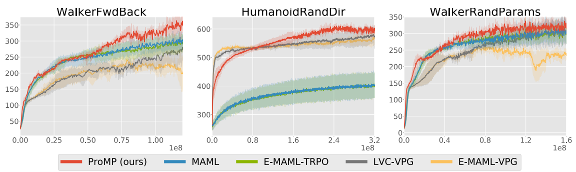
$$J_{\mathcal{T}}^{\mathrm{ProMP}}(\theta) = J_{\mathcal{T}}^{\mathrm{CLIP}}(\theta') - \eta \bar{\mathcal{D}}_{KL}(\pi_{\theta_o}, \pi_\theta) \quad \text{s.t.} \quad \theta' = \theta + \alpha \nabla_\theta J_{\mathcal{T}}^{LR}(\theta)$$

Incoporates the benefits of:

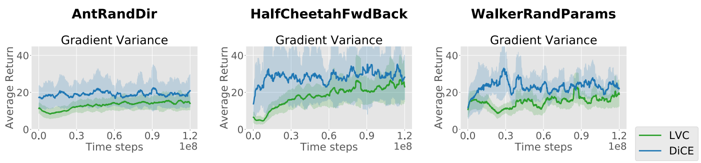
- Proximal Policy Optimization
- LVC Estimator

Experiments

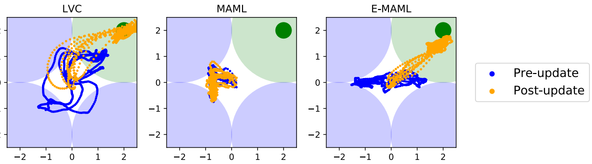
Performance Comparison



Variance Comparison



Exploration – Exploitation



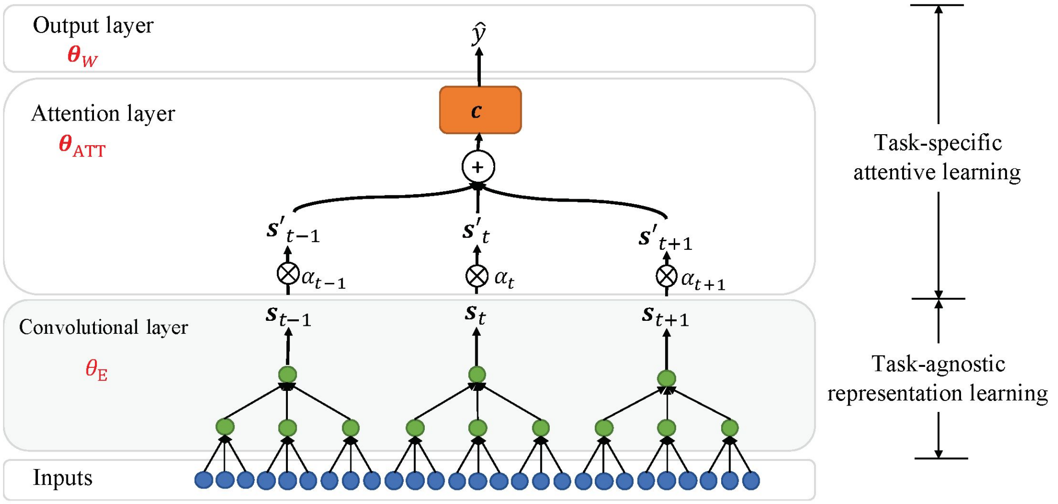
Attentive Task-Agnostic Meta-Learning for Few-Shot Text Classification

Xiang Jiang[1,2], Mohammad Havaei[1], Gabriel Chartrand[1], Hassan Chouaib[1], Thomas Vincent[1], Andrew Jesson,[1] Nicolas Chapados[1], Stan Matwin[2]
[1] Imagia Inc. [2]Dalhousie University

Task-agnostic representation learning

Task-specific attentive adaptation

Attention decouples the representation learning



Variadic Meta-Learning by Bayesian Nonparametric Deep Embedding

Kelsey Allen, Hanul Shin*, Evan Shelhamer*, Josh Tenenbaum

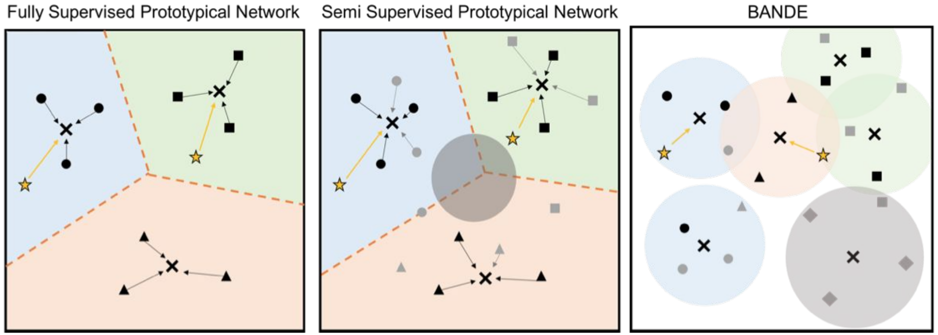
?

few-shot learning
small-scale

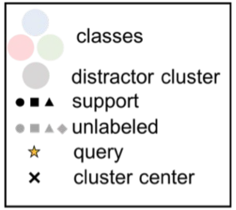
supervised learning
large-scale

variadic meta-learning

any-shot, any-way generalization
between meta-train and meta-test
with mixed supervision



BANDE clusters labeled and unlabeled data into *multi-modal prototypes* that represent each class by a set of clusters instead of only one



experiments:

- from **5-way to 1692-way** and
  from **1-shot to unsupervised** on Omniglot
- from **1-shot to 50-shot** on mini-ImageNet
- from **2-shot to 5000-shot** on CIFAR-10

with comparison of prototypes, MAML, graph nets, and good old supervised learning

multi-modal prototypes
for alphabet and character recognition

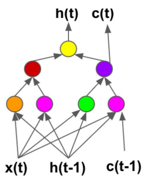
Training	Testing	Proto. Nets	BANDE
Alphabet	Alphabet	64.9±0.2	**91.2**±0.1
Alphabet	Chars (20-way)	85.7±0.2	**95.3**±0.2
Chars	Chars (20-way)	**94.9**±0.2	**95.1**±0.1

From Nodes to Networks: Evolving Recurrent Neural Networks

Aditya Rawal* , Risto Miikkulainen*

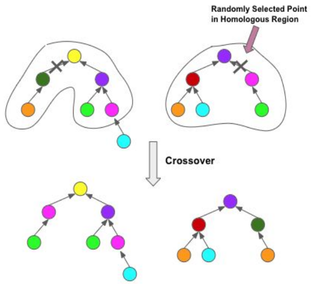
aditya.rawal@uber.com, risto@cs.utexas.edu

* Work done at Sentient Technologies



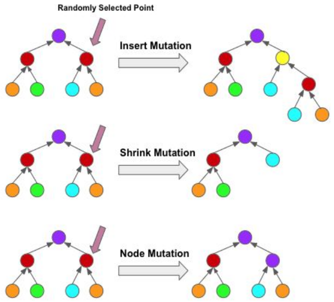
Recurrent Cell as Tree

Evolve

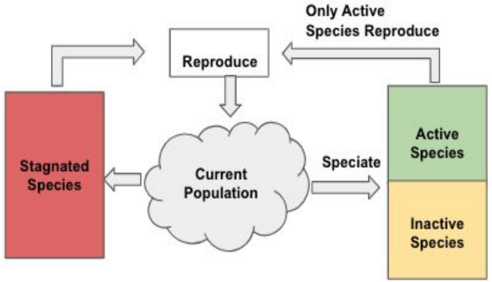


$$\delta_T(T_i,T_j)=\beta\frac{N_{i,j}-2n_{s_{i,j}}}{N_{i,j}-2}+(1-\beta)\frac{D_{i,j}-2d_{s_{i,j}}}{D_{i,j}-2}$$ (Tujillo et al., 2015)

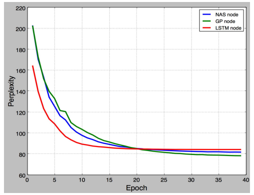
Crossover

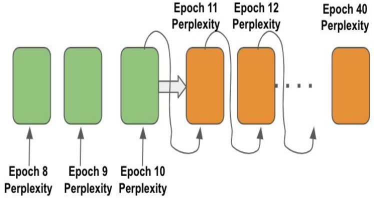


Mutation

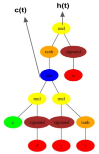


Encourage Search for Novel Cells

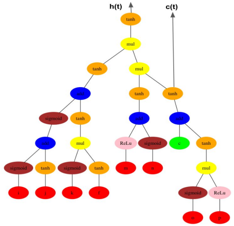




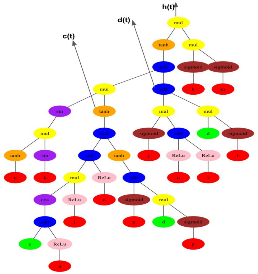
Meta-LSTM: Seq2Seq model to predict learning curve. Speeds-up search by **4X.**



LSTM



NAS Cell



Evolved Cell

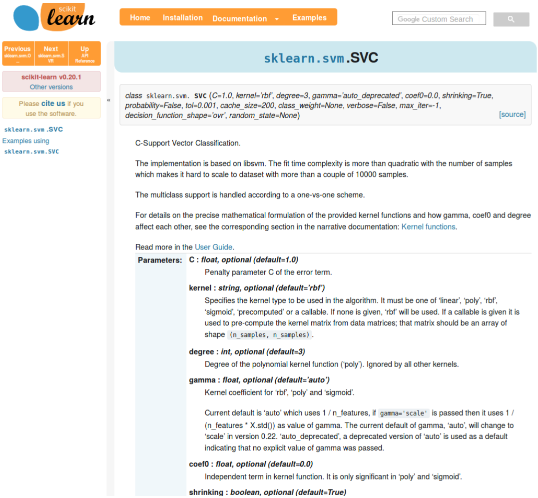
Language Modeling

Transfer to Music

Music

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren


scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous sklearn.svm.O... | Next sklearn.svm.S VR | Up API Reference

scikit-learn v0.20.1
Other versions

Please **cite us** if you use the software.

sklearn.svm.SVC
Examples using sklearn.svm.SVC

sklearn.svm.SVC

class `sklearn.svm.` **SVC** *(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)* [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters:

C : *float, optional (default=1.0)*
Penalty parameter C of the error term.

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *float, optional (default='auto')*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.std()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

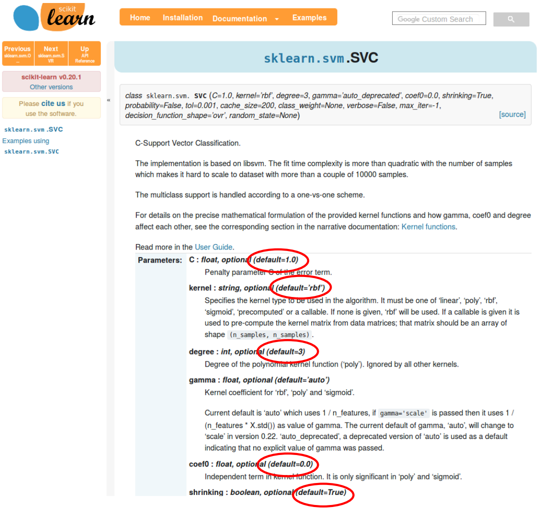
coef0 : *float, optional (default=0.0)*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *boolean, optional (default=True)*

- Defaults commonly used in Machine Learning research and practise

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren



scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous sklearn.svm.O... Next sklearn.svm.S VR Up API Reference

scikit-learn v0.20.1
Other versions

Please **cite us** if you use the software.

sklearn.svm.SVC
Examples using sklearn.svm.SVC

sklearn.svm.SVC

class sklearn.svm. **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters: **C : *float, optional (default=1.0)***
Penalty parameter C of the error term.

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *float, optional (default='auto')*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.std()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

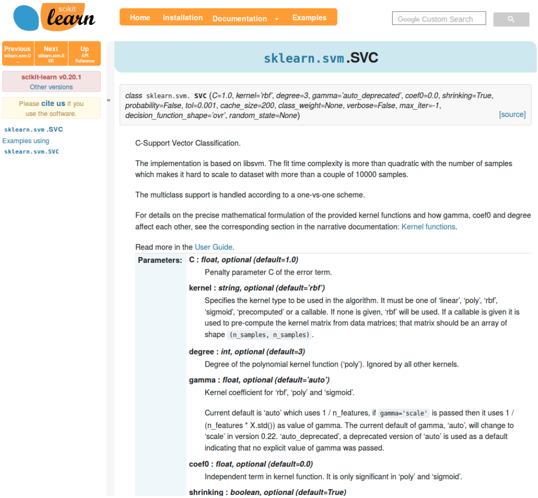
coef0 : *float, optional (default=0.0)*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *boolean, optional (default=True)*

- Defaults commonly used in Machine Learning research and practise
- Example: SVM(C=1.0, γ=0.0125, kernel=RBF)

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren



Home Installation Documentation Examples

Google Custom Search

Previous sklearn.svm.O... Next sklearn.svm.S VR Up API Reference

scikit-learn v0.20.1
Other versions

Please **cite us** if you use the software.

sklearn.svm.SVC
Examples using sklearn.svm.SVC

sklearn.svm.SVC

class sklearn.svm. **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters: **C** : ***float, optional (default=1.0)***

Penalty parameter C of the error term.

kernel : *string, optional (default='rbf')*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *float, optional (default='auto')*

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.std()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

coef0 : *float, optional (default=0.0)*

Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *boolean, optional (default=True)*

- Defaults commonly used in Machine Learning research and practise
- Example: SVM(C=1.0, γ=0.0125, kernel=RBF)
- Goal: Defaults based on meta-feature
- Example: SVM(C=85, γ=0.2 / num. features, kernel=RBF)
- Classical form of meta-learning
- Question: How to find good symbolic defaults?
- Answer: Let's discuss this at our poster!

Columbia Engineering
The Fu Foundation School of Engineering and Applied Science