[course site]

#DLUPC

Day 6 Lecture 1

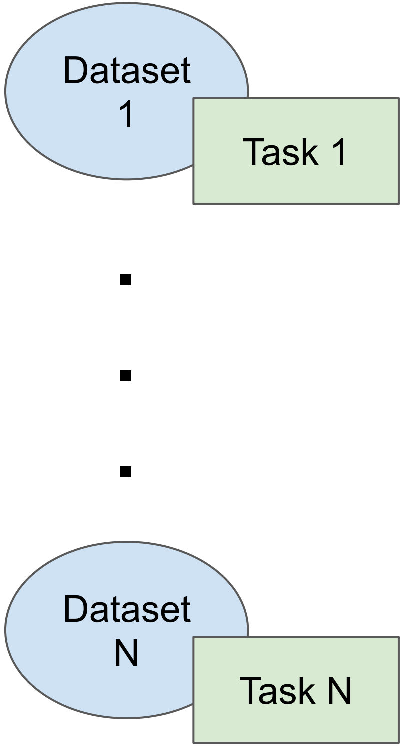
Life-long/incremental Learning

Ramon Morros
ramon.morros@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

‘Classical’ approach to ML

- Isolated, single task learning:
    - Well defined tasks.
    - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
- Data given prior to training
    - Model selection & meta-parameter optimization based on full data set
    - Large number of training data needed
- Batch mode
    - Examples are used at the same time, irrespective of their (temporal) order
- Assumption that data and its underlying structure is static
    - Restricted environment



Challenges

- Data not available priorly, but exemples arrive over time

- Memory resources may be limited
  - LML has to rely on a compact/implicit representation of the already observed signals
  - NN models provide a good implicit representation!

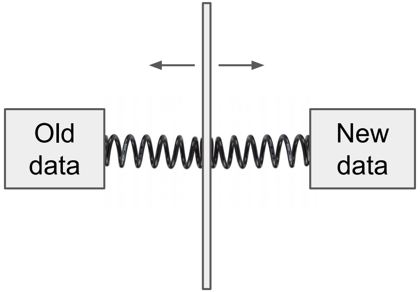
- Adaptive model complexity
  - Impossible to determine model complexity in advance
  - Complexity may be bounded by available resources ➔ intelligent reallocation
  - Meta-parameters such as learning rate or regularization strength can not be determined prior to training ➔ They turn into model parameters!

Challenges

- Concept drift: Changes in data distribution occurs with time
  - For instance, model evolution, changes in appearance, aging, etc.

Source:
https://www.youtube.com/watch?v=HMaWYBlo2Vc

- Stability -plasticity dilemma: When and how to adapt to the current model
  - Quick update enables rapid adaptation, but old information is forgotten
  - Slower adaptation allows to retain old information but the reactivity of the system is decreased
  - Failure to deal with this dilemma may lead to **catastrophic forgetting**



Lifelong Machine Learning (LML)

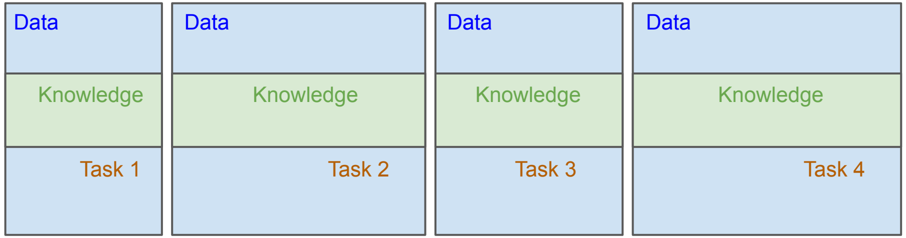
[Silver2013, Gepperth2016, Chen2016b]

Learn, retain, use knowledge over an extended period of time

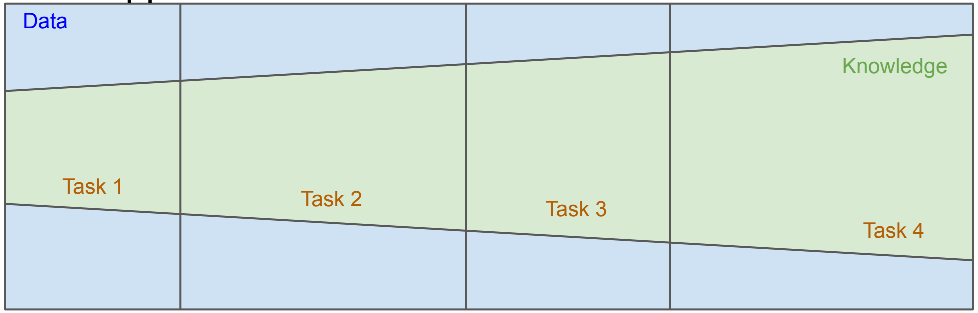
- Data streams, constantly arriving, not static ➜ Incremental learning
- Multiple tasks with multiple learning/mining algorithms
- Retain/accumulate learned knowledge in the past & use it to help future learning
  - Use past knowledge for inductive transfer when learning new tasks
- Mimics human way of learning

Lifelong Machine Learning (LML)

'Classical' approach



LML approach



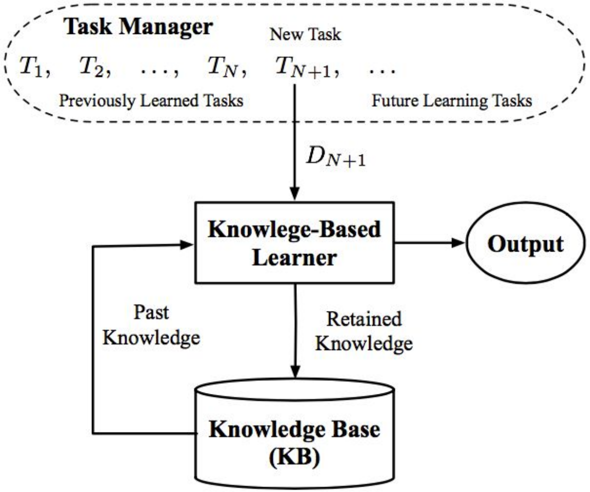


Image from [Chen2016a]

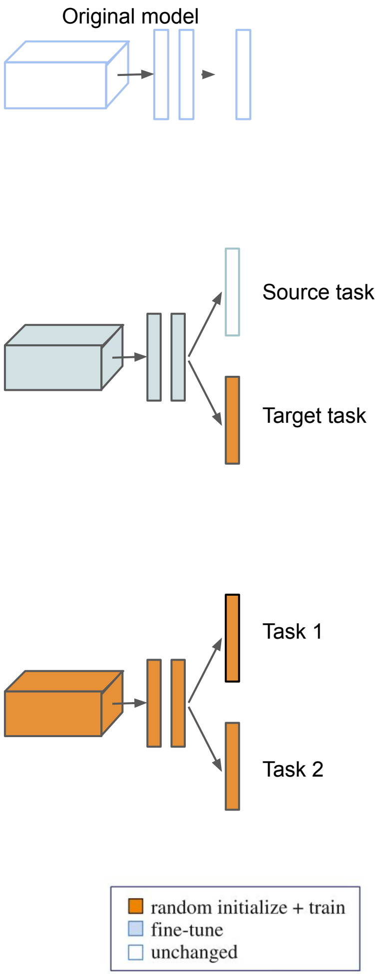
Related learning approaches

Transfer learning (finetuning):

- Data in the source domain helps learning the target domain
- Less data is needed in the target domain
- Tasks must be similar

Multi-task learning:

- Co-learn multiple, related tasks simultaneously
- All tasks have labeled data and are treated equally
- Goal: optimize learning/performance across all tasks through shared knowledge



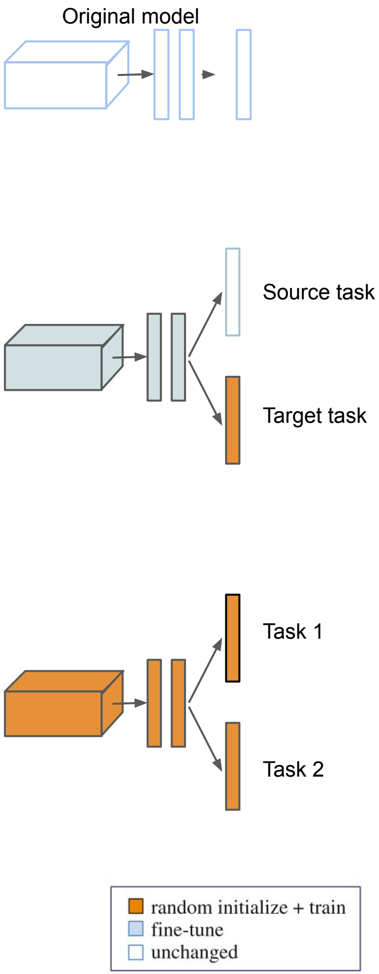
Related learning approaches

Transfer learning (finetuning):

- Unidirectional: source ➔ target
- Not continuous
- No retention/accumulation of knowledge

Multi-task learning:

- Simultaneous learning
- All tasks data is needed for training

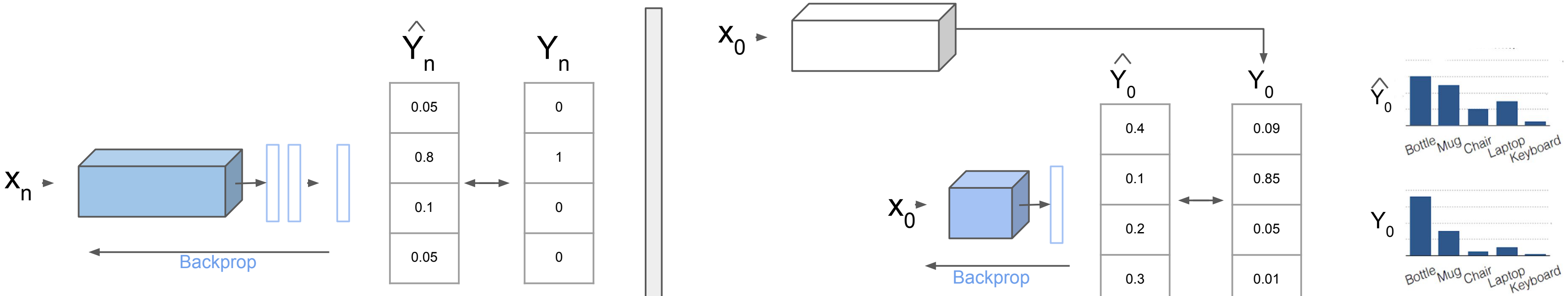


LML Methods

Distillation

Idea: use the class probabilities produced by the large model as “soft targets” for training the small model

- The ratios of probabilities in the soft targets provide information about the learned function
- These ratios carry information about the structure of the data
- Train by replacing the hard labels with the **softmax activations from the original large model**



Multinomial logistic loss

$$\mathcal{L}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

Distillation loss

$$\mathcal{L}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o$$

Hinton, G., Vinyals, O., & Dean, J. “Distilling the Knowledge in a Neural

LWF: Learning without Forgetting [Li2016]

Goal:

Add **new prediction tasks** based on adapting shared parameters **without access to training data for previously learned tasks**

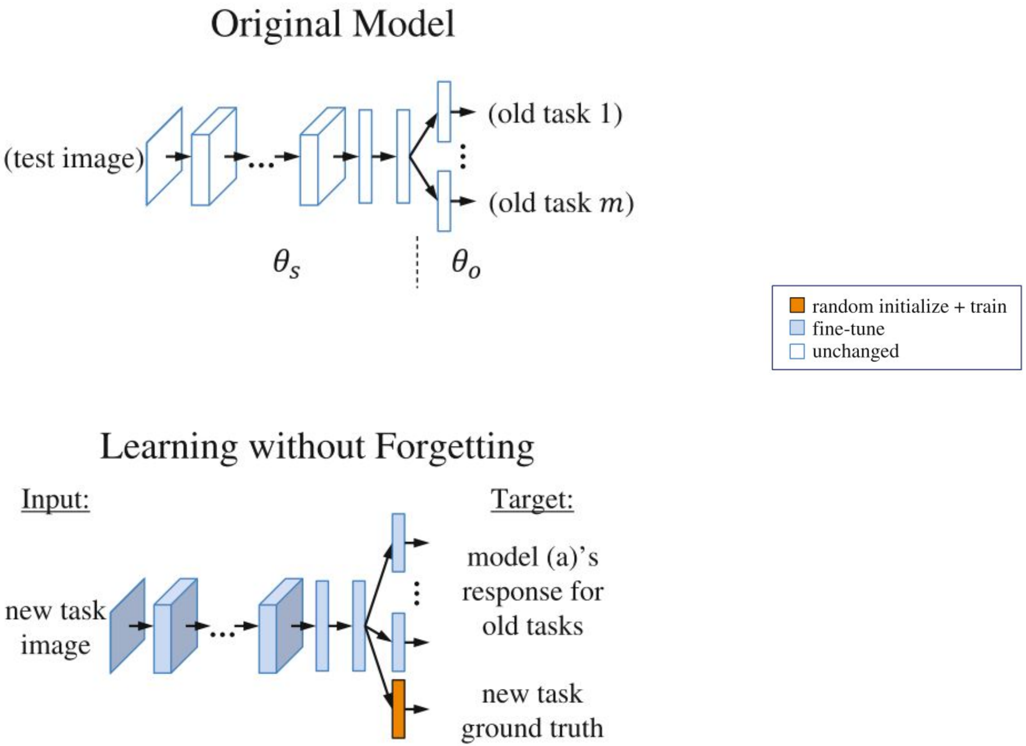
Solution:

Using only examples for the new task, optimize for :

- High accuracy on the new task
- Preservation of responses on existing tasks from the original network (distillation, Hinton2015)
- Storage/complexity does not grow with time. Old samples are not kept

Preserves performance on old task
(even if images in new task provide a poor sampling of old task)

LWF: Learning without Forgetting [Li2016]



LWF: Learning without Forgetting [Li2016]

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

X_n, Y_n: training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \mathrm{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

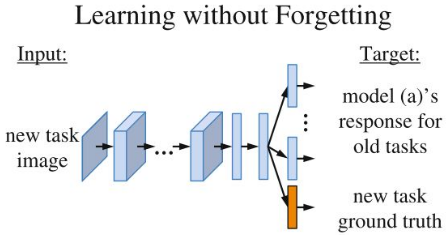
$\theta_n \leftarrow \mathrm{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

Define $\hat{Y}_o \equiv \mathrm{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*

Define $\hat{Y}_n \equiv \mathrm{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*

$$\theta_s^*, \ \theta_o^*, \ \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\operatorname{argmin}} \left(\mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$



Multinomial logistic loss

$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

Weight decay of 0.0005

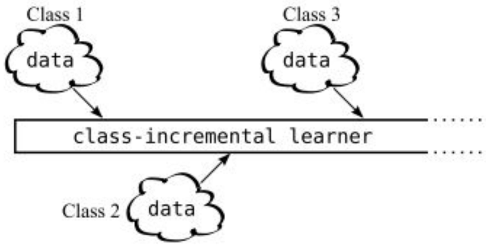
$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o \qquad y'^{(i)}_o = \frac{(y^{(i)}_o)^{1/T}}{\sum_j (y^{(j)}_o)^{1/T}}, \quad \hat{y}'^{(i)}_o = \frac{(\hat{y}^{(i)}_o)^{1/T}}{\sum_j (\hat{y}^{(j)}_o)^{1/T}}.$$

Distillation loss

iCaRL

Goal:

Add new classes based on adapting shared parameters **with restricted access** to training data for previously learned classes.



Solution:

- A subset of training samples (exemplar set) from previous classes is stored.
- Combination of classification loss for new samples and distillation loss for old samples.
- The size of the exemplar set is kept constant. As new classes arrive, some examples from old classes are removed.

iCaRL: Incremental Classifier and Representation learning

Algorithm 2 iCaRL INCREMENTALTRAIN

input $X^s, \dots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets
$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \dots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \dots, s-1$ **do**
  $P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \dots, t$ **do**
  $P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \dots, P_t)$ // new exemplar sets

Algorithm 3 iCaRL UPDATEREPRESENTATION

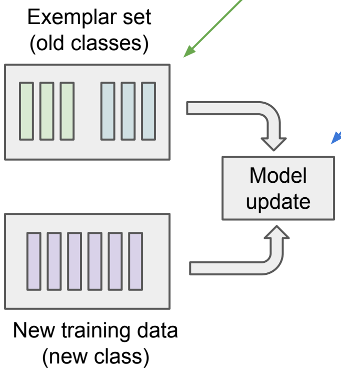
input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
  $q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log(g_y(x_i)) \quad \text{// classification loss}$$
$$+ \sum_{y=1}^{s-1} q_i^y \log(g_y(x_i)) \Big] \quad \text{// distillation loss}$$

[Hinton2015]

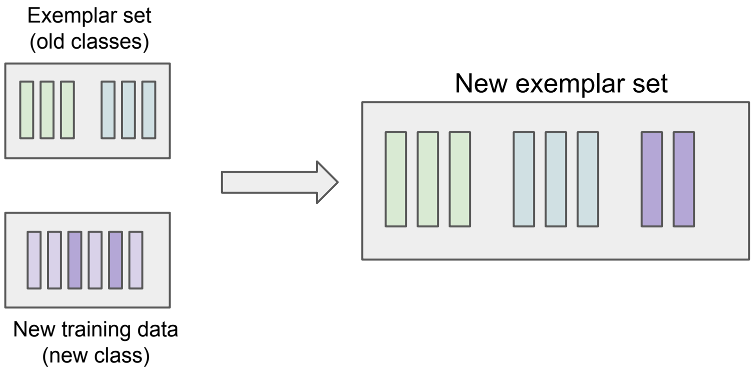


iCaRL: Incremental Classifier and Representation learning

Algorithm 2 iCaRL INCREMENTALTRAIN

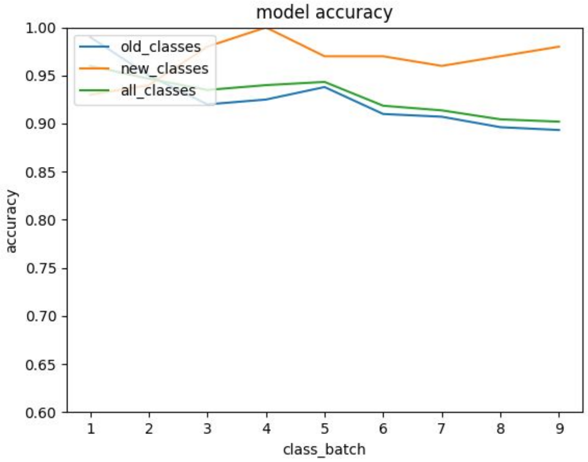
input $X^s, \dots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets

$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \dots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \dots, s-1$ **do**
  $P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \dots, t$ **do**
  $P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \dots, P_t)$ // new exemplar sets

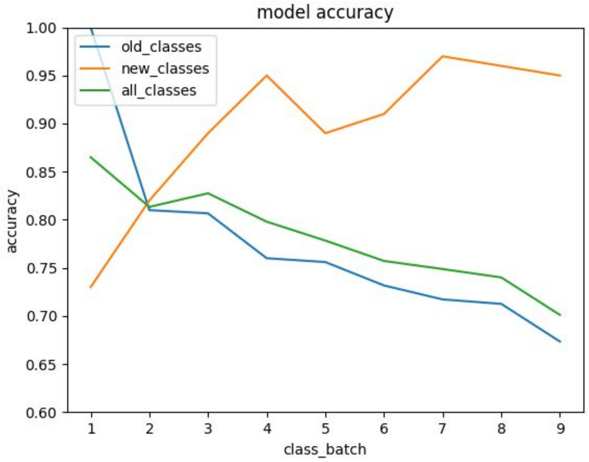


Results on face recognition

- Preliminary results from Eric Presas TFG (co-directed with Elisa Sayrol)



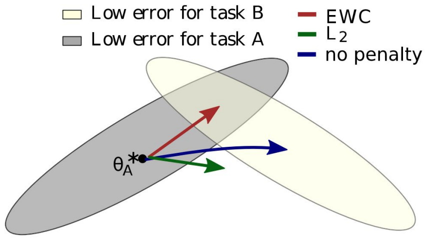
iCaRL



LWF

Elastic Weight Consolidation (EWC)

- Evidence suggests that the mammalian brain may avoid catastrophic forgetting by protecting previously acquired knowledge in neocortical circuits
- Knowledge is durably encoded by rendering a proportion of synapses less plastic (stable over long timescales)
- EWC algorithm slows down learning on certain weights based on how important they are to previously seen tasks
- While learning task B, EWC therefore protects the performance in task A by constraining the parameters to stay in a region of low error for task A centered around θ*
- Constraint implemented as a quadratic penalty. Can be imagined as a spring anchoring the parameters to the previous solution (elastic).
- The stiffness of this spring should not be the same for all parameters; rather, it should be greater for parameters that most affect performance in task A



$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

F: Fisher information matrix
(https://en.wikipedia.org/wiki/Fisher_information#Matrix_form)

Kirkpatrick *et al*. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences*, *114*(13), 2017