DeepMind

Functional Regularisation for Continual Learning with Gaussian Processes

Michalis Titsias

What is Continual Learning?

Continual Learning (CL):

- Life-long learning from data and tasks
- Imagine an agent that keeps learning in an online fashion
- Tasks are coming sequentially
- Systems are based on **deep neural networks**

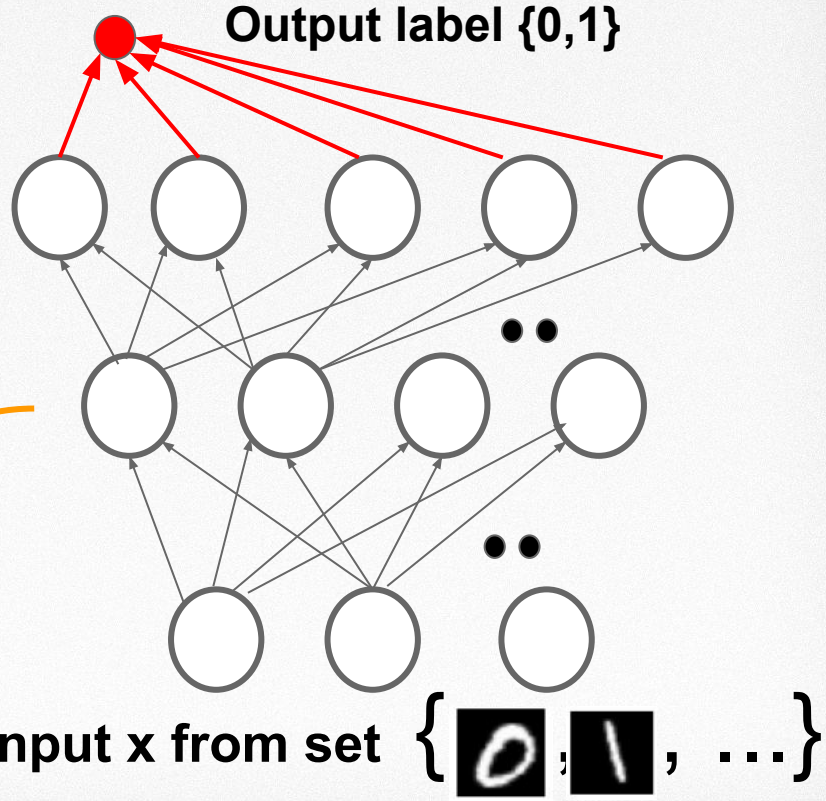
What is Continual Learning?

First task:

Classify instances of digits 0,1

- Binary classification problem
- Input is an image
- Output is binary label {0,1}

Neural network that maps input to output



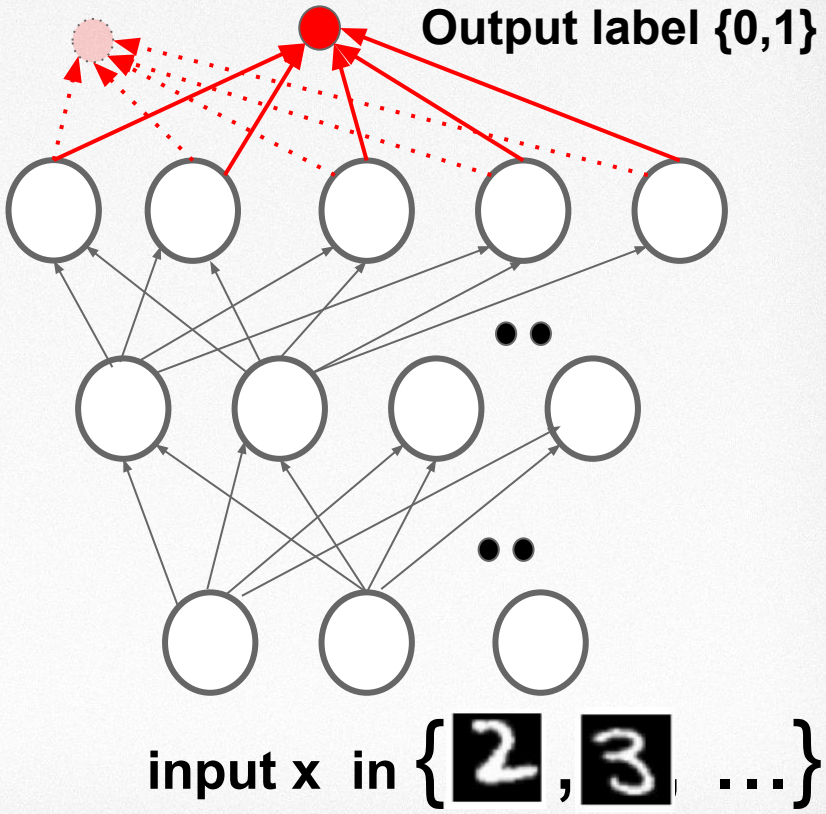
What is Continual Learning?

Second task:

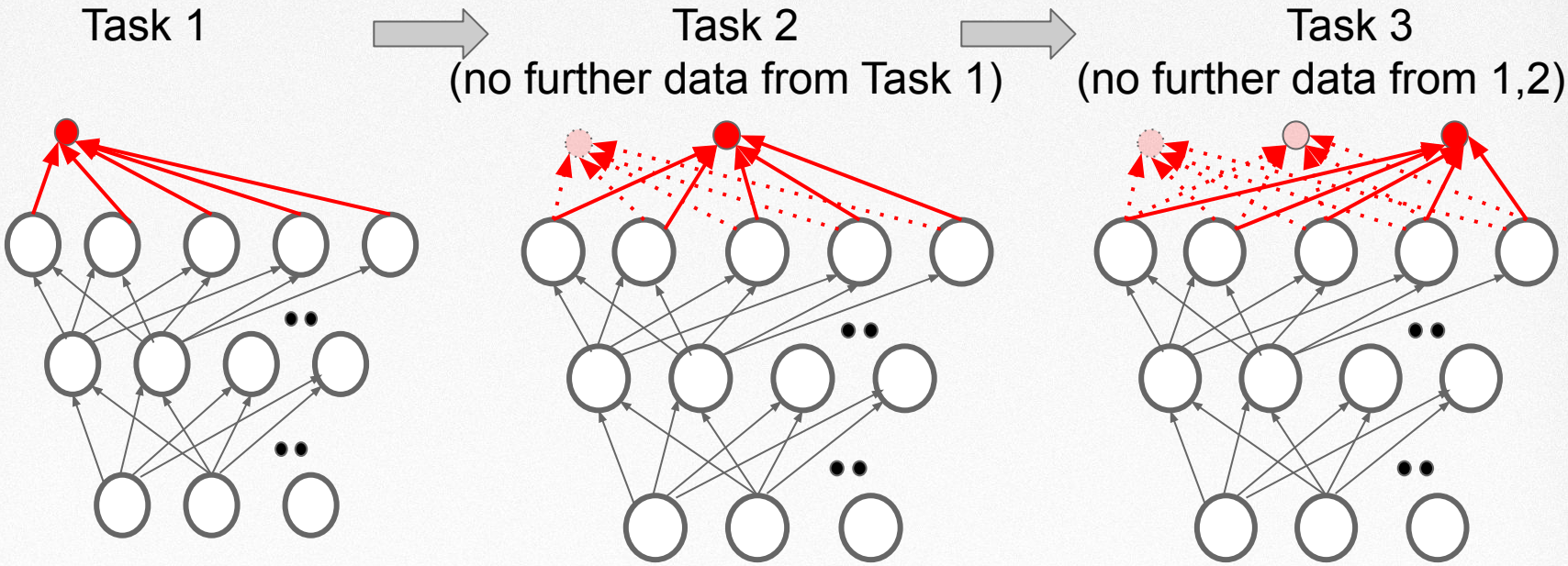
Classify instances of digits 2,3

- Again binary classification problem

- **No further training data from task 1 (training must continue without forgetting task 1)**



What is Continual Learning?



As we learn new tasks, the NN might forget previous tasks

- **Catastrophic forgetting**

What is Continual Learning?

Challenges of Continual Learning:

- Avoid **catastrophic forgetting**
- **Scalability over tasks**: Don’t restrict the capacity and allow the network to learn many tasks
- Lean **without task boundaries**: Detect when data are coming from a new task
- **Transfer learning**: Learn faster new tasks

Outline

- Problem Setup
- Continual Learning with Experience Replay
- From Bayesian NNs to Gaussian Processes
- Functional Regularised Continual Learning
- Automatic detection of task changepoints
- Experiments
- Conclusion

Problem Setup

Tasks/data are encountered sequentially $(X_1, \mathbf{y}_1), (X_2, \mathbf{y}_2), \ldots$

- Input set of data : X_i
- Corresponding outputs/labels : $\mathbf{y}_i$

Learn based on a deep NN: final hidden layer provides the features $\phi(x; \theta) \in R^K$

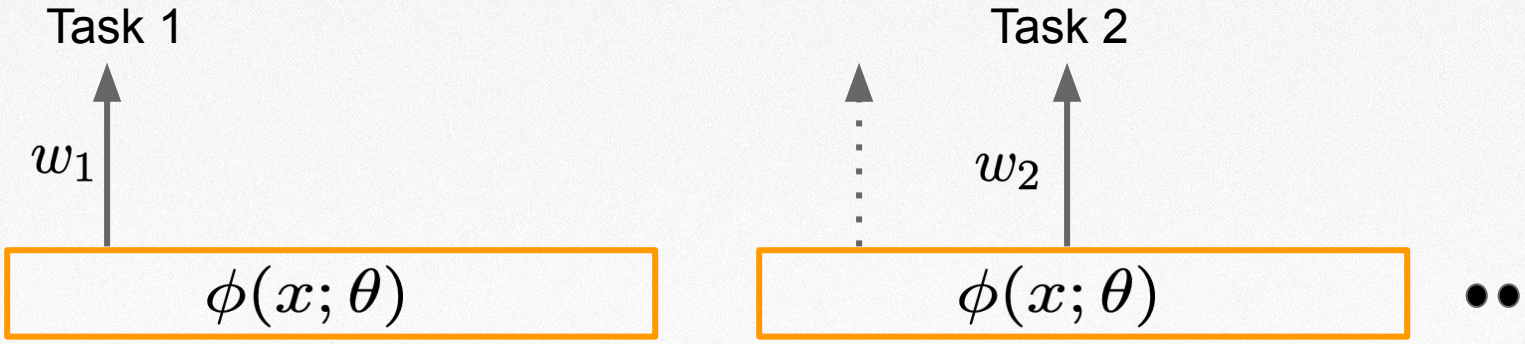
This representation is shared across tasks: θ is a shared parameter

To solve specific task *i* construct an output layer/function

$$f_i(x; w_i) = w_i^\top \phi(x; \theta)$$

where w_i vector of task specific weights

Problem Setup



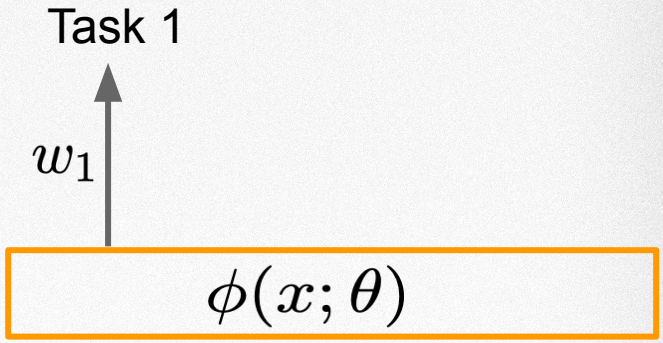
Each task adds a new head/output to the shared feature vector

We need to keep learning continually the feature parameters θ without forgetting previous tasks

Continual Learning with Experience Replay

Learning the first task is just a standard ML problem

We have a loss (e.g. negative log-likelihood)



$$\ell_1(\mathbf{y}_1, X_1; w_1, \theta) = -\log p(\mathbf{y}_1 | X_1, w_1, \theta)$$

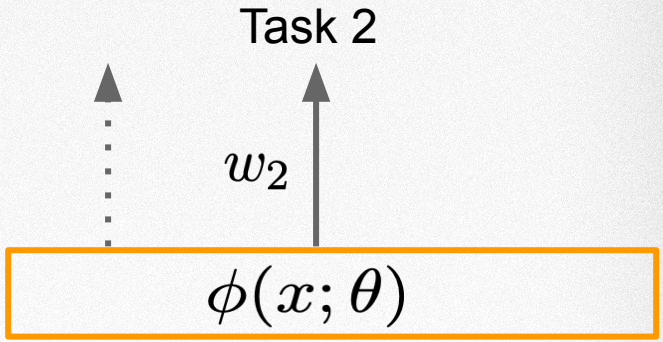
Minimise the loss by applying stochastic gradient descent (SGD) using mini-batches

Continual Learning with Experience Replay

Data from 2nd task arrive (and will not be getting data from 1st task anymore!)

We need to add regularisation to not forget 1st task

We store a small subset of data from the 1st task to replay them



Regularised loss:

$$\ell_2(\mathbf{y}_2, X_2; w_2, \theta) + \lambda \ell_1(\widetilde{\mathbf{y}}_1, \widetilde{X}_1; w_1, \theta)$$

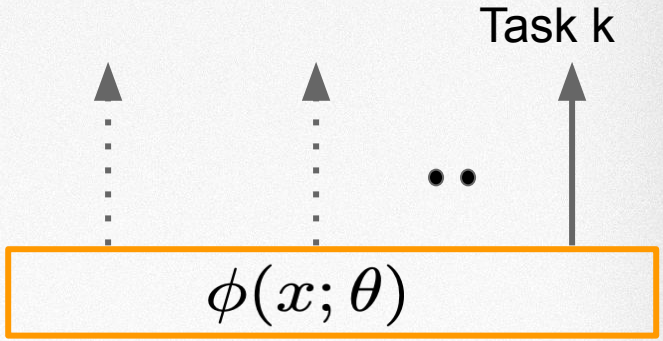
Where the small subset of data for replaying is: $(\widetilde{X}_1, \widetilde{\mathbf{y}}_1)$

Continual Learning with Experience Replay

When learning the *k*-th task the loss is

$$\ell_k(\mathbf{y}_k, X_k; w_k, \theta) + \lambda \sum_{i=1}^{k-1} \ell_i(\widetilde{\mathbf{y}}_i, \widetilde{X}_i; w_i, \theta)$$

Where the regularisation is a sum of replays from all previous tasks

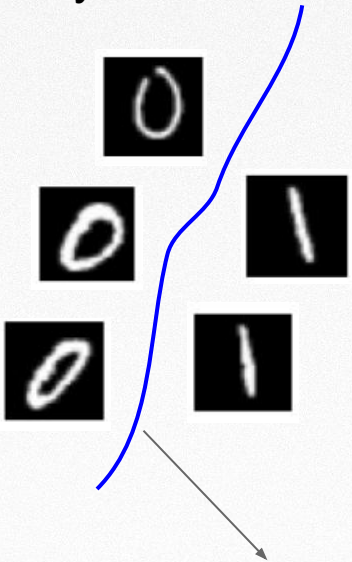


Experience replay is quite efficient. But still limited:

- it does not take into account uncertainty
- uncertainty can be really important for better regularisation

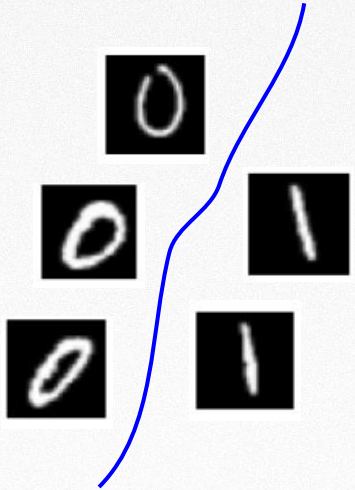
(Example of why uncertainty matters)

Suppose we learned to classify 0s from 1s



Decision boundary

What the system will do if a new point arrives from a different distribution ?

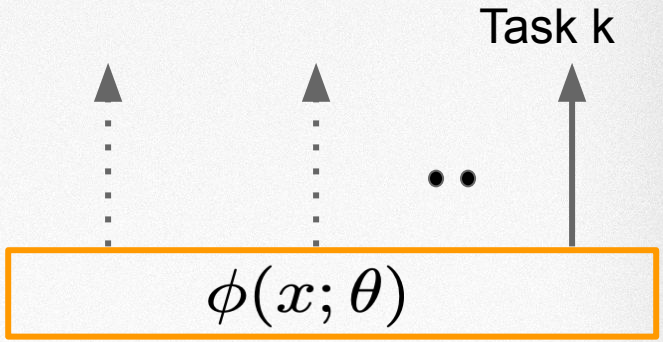


New data point

Continual Learning with Experience Replay

When learning the *k*-th task the loss is

$$\ell_k(\mathbf{y}_k, X_k; w_k, \theta) + \lambda \sum_{i=1}^{k-1} \ell_i(\widetilde{\mathbf{y}}_i, \widetilde{X}_i; w_i, \theta)$$



We would like to improve experience replay methods by adding uncertainty

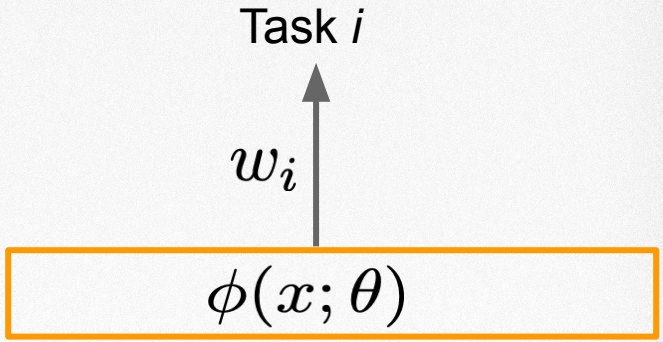
How? Use Bayesian inference:

- In particular concepts from Gaussian processes (GPs)
- (we will cover first some basics about Bayesian NNs and GPs and then return to continual learning)

From Bayesian NNs to Gaussian Processes

To introduce uncertainty we apply Bayesian inference over task-specific weights

$$f_i(x; w_i) = w_i^\top \phi(x; \theta)$$



Gaussian prior over w_i :

$$p(w_i) = \mathcal{N}(w_i|0, \sigma_w^2 I)$$

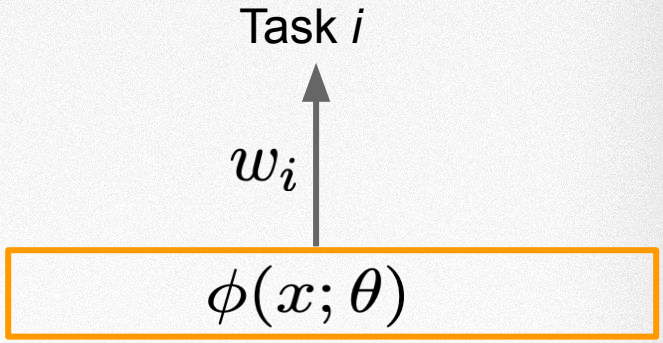
Where for the task *i* the weights are different draws from the same prior

From Bayesian NNs to Gaussian Processes

Bayesian learning and prediction

Observe data : $X_i, \mathbf{y}_i$

Train the feature parameters θ by maximising the marginal likelihood



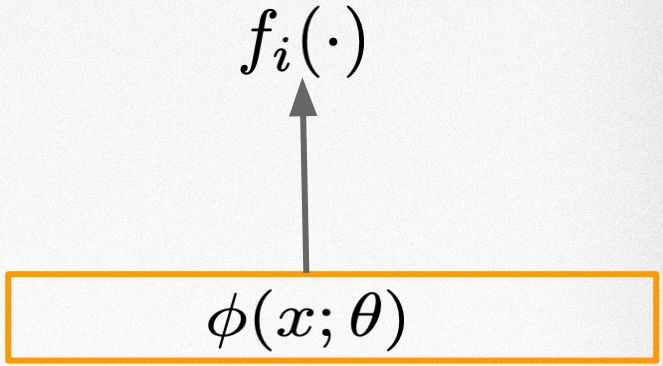
$$p(\mathbf{y}_i|X_i,\theta) = \int p(\mathbf{y}_i|X_i,w_i,\theta)p(w_i)dw_i$$

Compute posterior over the weights: $p(w_i|\mathbf{y}_i,X_i,\theta) = \dfrac{p(\mathbf{y}_i|X_i,w_i,\theta)p(w_i)}{p(\mathbf{y}_i|X_i,\theta)}$

Then do prediction on test points: $p(y_*|x_*,\theta) = \int p(y_*|x_*,w_i,\theta)p(w_i|\mathbf{y}_i,X_i,\theta)dw_i$

From Bayesian NNs to Gaussian Processes

$$f_i(x) = w_i^\top \phi(x;\theta)$$



Since the weights are Gaussian, the above is just linear combination of Gaussian variables

So the direct output function values $f_i(x)$ are also Gaussian
Their mean and covariance is

$$E[f_i(x)] = 0, \quad Cov(f_i(x), f_i(x')) = k(x,x') = \sigma_w^2 \phi(x;\theta)^\top \phi(x';\theta)$$

This means that the full function (consisting of all infinite many points) is a Gaussian process:

$$f_i(\cdot) \sim \mathcal{GP}(0, k(x,x'))$$

From Bayesian NNs to Gaussian Processes

The Gaussian process or function space representation leads to non-parametric inference

We have as many parameters as training data

$$[\mathbf{f}_i]_j = f_i(x_j), \quad x_j \in X_i$$

Prior $p_\theta(\mathbf{f}_i) = \mathcal{N}(\mathbf{f}_i|\mathbf{0}, K_i), \quad [K_i]_{j,k} = \sigma_w^2 \phi(x_{i,j};\theta)^\top \phi(x_{i,k};\theta)$
(**a big Gaussian of the size of the training set!**)

Posterior distribution : $\underbrace{p(\mathbf{f}_i|\mathbf{y}_i, X_i)}_{\text{Posterior}} \propto \underbrace{p(\mathbf{y}_i|\mathbf{f}_i)}_{\text{Likelihood}} \times \underbrace{p_\theta(\mathbf{f}_i)}_{\text{Prior}}$

From Bayesian NNs to Gaussian Processes

Function and weight space inference are equivalent from theoretical point of view (computationally can be very different though!)

E.g. the marginal likelihood is the same

$$p(\mathbf{y}_i|X_i,\theta) = \int p(\mathbf{y}_i|X_i,w_i,\theta)p(w_i)dw_i = \int p(\mathbf{y}_i|\mathbf{f}_i)p(\mathbf{f}_i|\theta)d\mathbf{f}_i$$

The second integral is simply obtained by reparametrising (i.e.changing variables) in the first one, so that the new variables are

$$f_i(x_j) = w_i^\top \phi(x_j;\theta), \quad x_j \in X_i$$

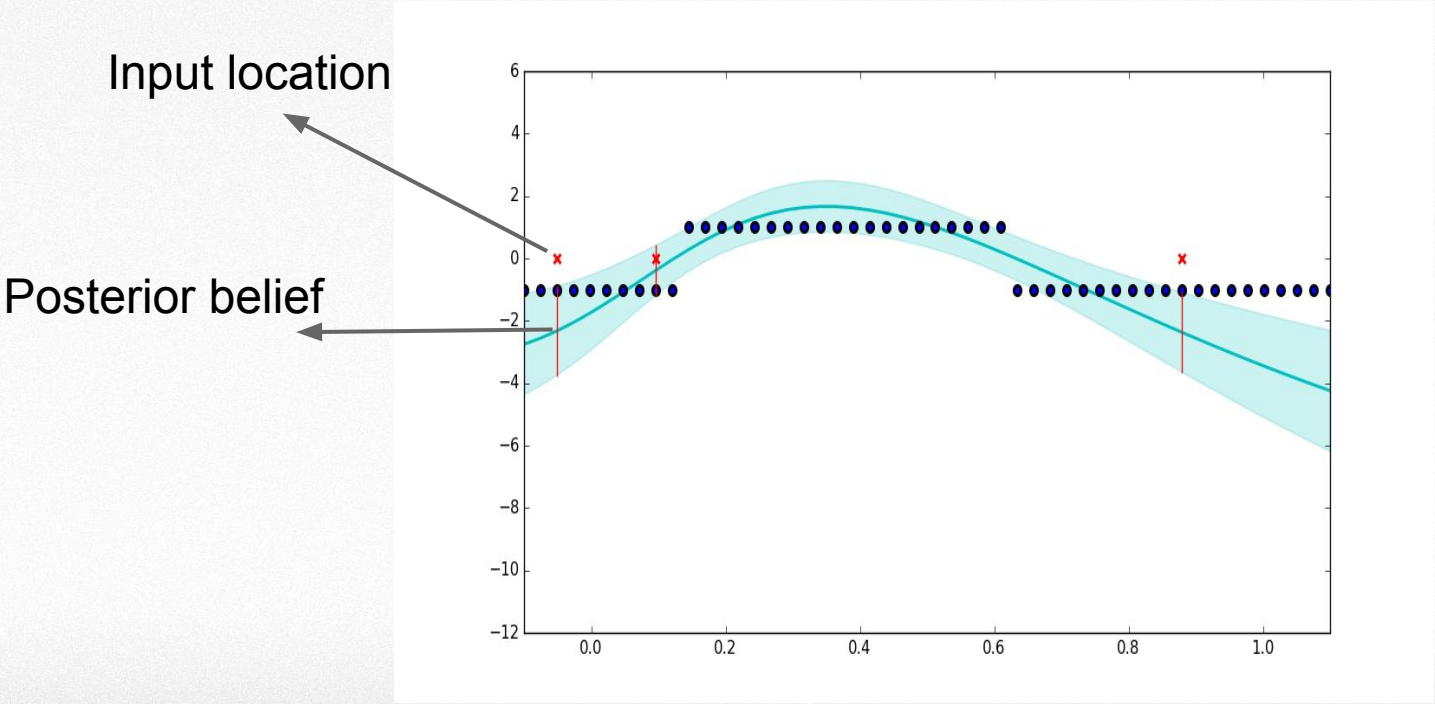
From Bayesian NNs to Gaussian Processes

However, the function view has some advantages for continual learning

- Learning a task corresponds to learning the shape of a function
- Thus, to avoid forgetting the task we could remember the values of that function in some informative input locations (that best describe the shape)
- Even better we could remember a posterior belief/distribution on those function values

From Bayesian NNs to Gaussian Processes

Remembering a function at informative locations in binary classification example



Functional Regularisation for Continual Learning

Algorithm (makes experience replay methods more Bayesian):

1. Learn each task using Bayesian inference

2. From the functional view obtain uncertainties/posterior distribution over function values on the training data

3. Select informative function input locations and store them for replay

4. Regularise continual learning by replaying full posterior distributions on the selected inputs (and not input-label pairs)

Functional Regularisation for Continual Learning

Learning the first task (and updating parameters θ) requires maximising the log marginal likelihood

$$\log p(\mathbf{y}_1|X_1,\theta) = \log \int p(\mathbf{y}_1|X_1,w_1,\theta)p(w_1)dw_1$$

But, this is intractable and instead we maximise the evidence lower bound (ELBO)

$$\log p(\mathbf{y}_1|X_1,\theta) = \log \int \frac{q(w_1)}{q(w_1)} p(\mathbf{y}_1|X_1,w_1,\theta)p(w_1)dw_1$$

$$\log p(\mathbf{y}_1|X_1,\theta) \geq F_1(q,\theta) = \int q(w_1) \log \frac{p(\mathbf{y}_1|X_1,w_1,\theta)p(w_1)}{q(w_1)} dw_1$$

Functional Regularisation for Continual Learning

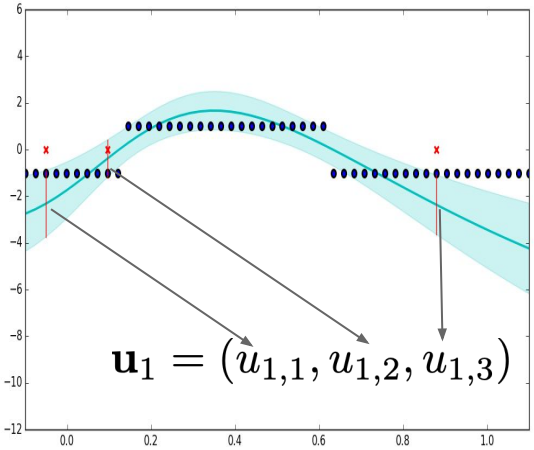
$$F_1(q,\theta) = \int q(w_1) \log \frac{p(\mathbf{y}_1|X_1, w_1, \theta)p(w_1)}{q(w_1)} dw_1$$

Where we maximize this over θ and the parameters of Gaussian variational distribution

$$q(w_1) = \mathcal{N}(w_1|\mu_{w_1}, \Sigma_{w_1})$$

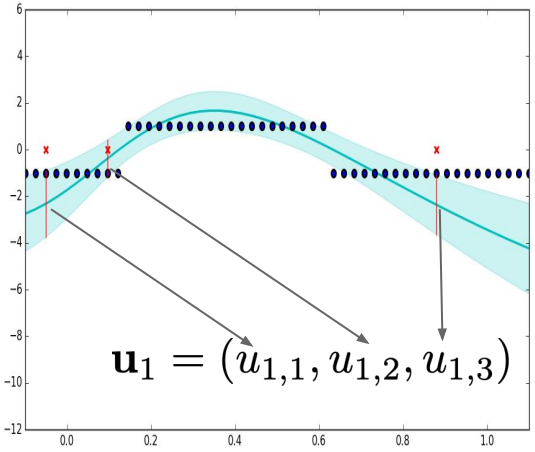
Given this posterior over weights we can express the posterior for any set of function values

$$q(\mathbf{u}_1) = \mathcal{N}(\mathbf{u}_1|\mu_{u_1}, \Sigma_{u_1})$$



Functional Regularisation for Continual Learning

$$q(\mathbf{u}_1) = \mathcal{N}(\mathbf{u}_1 | \mu_{u_1}, \Sigma_{u_1})$$



We also have to select the input locations.
We use

- either random selection
- or specialised criteria from the Gaussian process literature

Then we store the (small) set of selected inputs and the posterior $(\mu_{u_1}, \Sigma_{u_1})$

- And use them to avoid forgetting the task/function that we learned

Functional Regularisation for Continual Learning

Finally when we learn the k-th task we have an objective of the form

$$F_k(q_k, \theta) - \sum_{i=1}^{k-1} \mathrm{KL}(q(\mathbf{u}_i)||p(\mathbf{u}_i; \theta))$$

Where each $\mathrm{KL}(q(\mathbf{u}_i)||p(\mathbf{u}_i; \theta))$ is the Kullback Leibler divergence of the GP prior that depends on θ to remain consistent with the fixed posterior belief of the *i*-th previous task

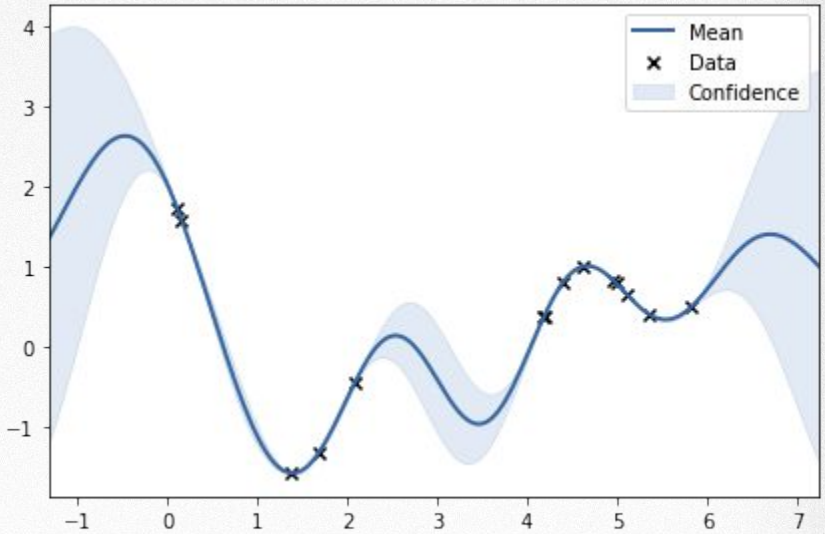
Compare the above with the non-Bayesian experience replay loss

$$\ell_k(\mathbf{y}_k, X_k; w_k, \theta) + \lambda \sum_{i=1}^{k-1} \ell_i(\widetilde{\mathbf{y}}_i, \widetilde{X}_i; w_i, \theta)$$

Automatic detection of task changepoints

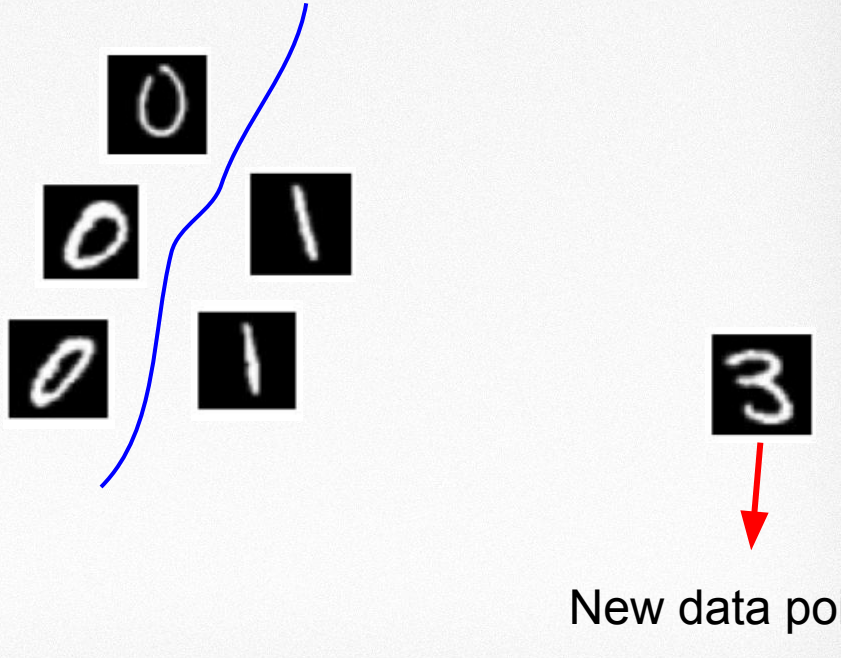
An interesting property of Bayesian posterior over function values (i.e. the functional or GP view) is that you obtain reduced uncertainty close to the training data

But far away from the training data your uncertainty Is high and close to the prior uncertainty



Automatic detection of task changepoints

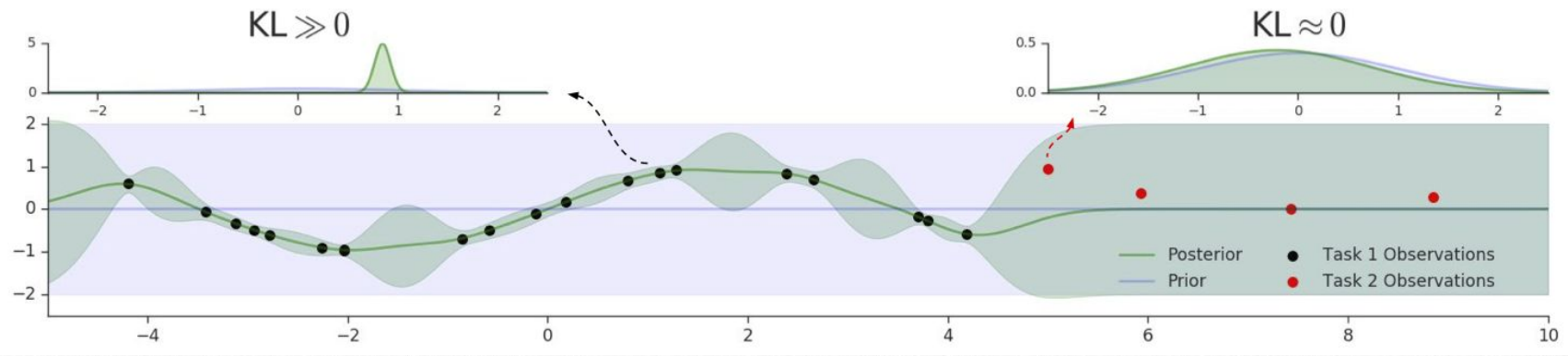
What the system will do if a new point arrives from a different distribution ?



A good Bayesian model should know that is best to not classify the new data point at all!

- Too much uncertainty there..

Automatic detection of task changepoints



The increased uncertainty is useful to detect out-of-distribution data, such as when suddenly there is a changepoint and the new data are coming from a new task/distribution

Experiments

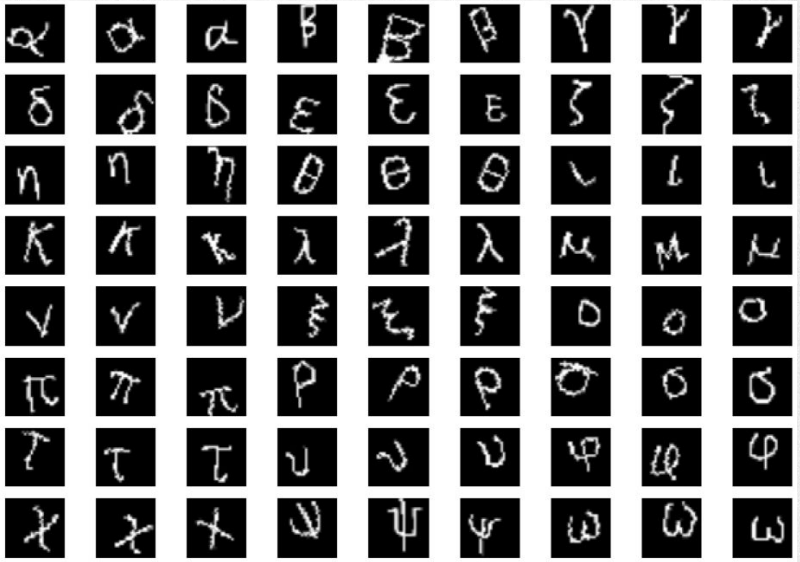
The most challenging experiment we consider is the omniglot dataset

Learn sequentially to classify 50 different alphabets (Latin, Greek, etc)

There are 50 tasks

Each task/alphabet is a multi-class classification problem where inputs are images

We use a convnet to construct the shared feature vector



Experiments

Table 2: Results on sequential Omniglot. Baseline results are taken from Schwarz et al. (2018). Shown are mean and standard deviation over 5 random task permutations. Note that methods *'Single model per Task'* and *'Progressive Nets'* are not directly comparable due to unrealistic assumptions, but serve as an upper bound on the performance for the remaining continual learning methods.

Algorithm	**Test Accuracy**		
Single model per Task	88.34		
Progressive Nets	86.50 $\pm$ 0.9		
Finetuning	26.20 $\pm$ 4.6		
Learning Without Forgetting	62.06 $\pm$ 2.0		
Elastic Weight Consolidation (EWC)	67.32 $\pm$ 4.7		
Online EWC	69.99 $\pm$ 3.2		
Progress & Compress	70.32 $\pm$ 3.3		
Methods evaluated in this paper	**1 point/char**	**2 points/char**	**3 points/char**
BASELINE	42.73 $\pm$ 1.2	57.17 $\pm$ 1.2	65.32 $\pm$ 1.1
FRCL (RANDOM)	69.74 $\pm$ 1.1	80.32 $\pm$ 2.5	**81.42** $\pm$ 1.2
FRCL (TRACE)	**72.02** $\pm$ 1.3	**81.47** $\pm$ 1.6	81.01 $\pm$ 1.1

Conclusion

A new algorithm for continual learning that combines Bayesian inference and experience replay

It uses concepts from Gaussian processes and function space inference and regularisation

Future work: Apply the method to **reinforcement learning**

Further details can be found in:

M. K. Titsias, J. Schwarz, A. G. de G. Matthews, R. Pascanu and Y. W.Teh. Functional Regularisation for Continual Learning with Gaussian Processes. International Conference on Learning Representations,2020, to appear