Bayesian Inference and Deep Learning

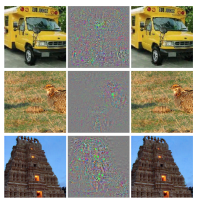
Dr. Richard E. Turner (ret26@cam.ac.uk)
Computational and Biological Learning Lab, Department of Engineering,
University of Cambridge

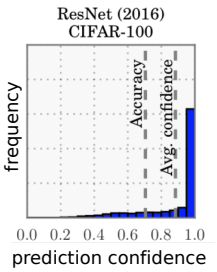
Limitations of Deep Learning

Robust Deep Learning

fragile (adversarial examples)

well calibrated uncertainty estimates:
deep learning is often confidently wrong





Data-Efficient Deep Learning

small data, big models (few-shot learning
and reinforcement learning)

leverage heterogenous data sources (multi-task learning)

Flexible Deep Learning

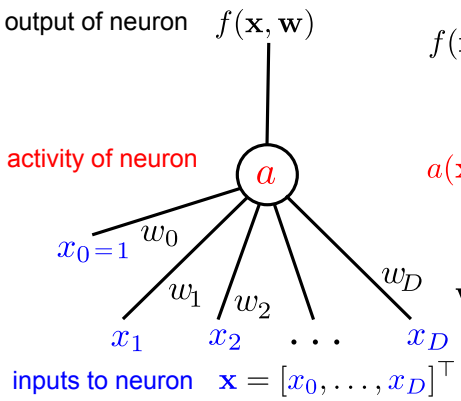
continual learning (online learning & model building)

active learning (RL exploration-exploitation)

probabilistic modelling
+
probabilistic inference

Logistic Regression as a motivating example

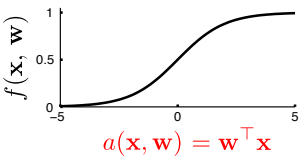
Logistic regression: A single neuron

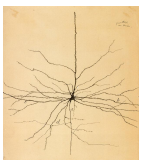


$$f(\mathbf{x}, \mathbf{w}) = P(y = 1 | \mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-a(\mathbf{x}, \mathbf{w}))}$$

$$a(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{d=1}^{D} w_d x_d = \mathbf{w}^\top \mathbf{x}$$

$$\mathbf{w} = [w_0, \ldots, w_D]^\top$$

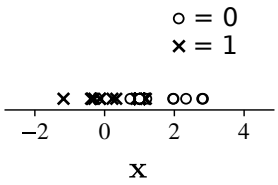




Logistic regression: Maximum Likelihood Estimation

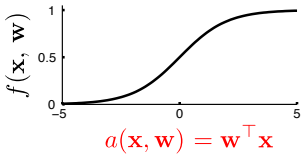
observe data: estimate parameters

$$\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^N$$
$$\mathcal{Y} = \{y_n\}_{n=1}^N$$



$$f(\mathbf{x}, \mathbf{w}) = P(y = 1|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-a(\mathbf{x}, \mathbf{w}))}$$

$$a(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{d=1}^{D} w_d x_d = \mathbf{w}^\top \mathbf{x}$$
$$= w_0 + w_1 x_1$$



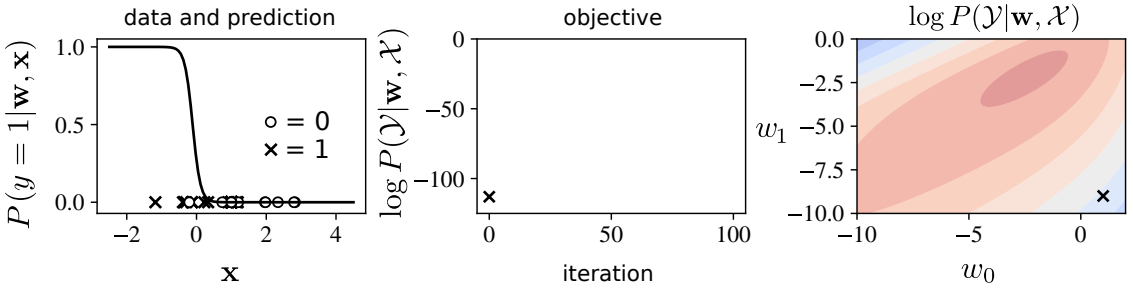
maximum likelihood estimate: parameters that make observed data most probable

$$P(\mathcal{Y}|\mathbf{w}, \mathcal{X}) = \prod_{n=1}^{N} P(y_n|\mathbf{w}, \mathbf{x}_n) = \prod_{n=1}^{N} f(\mathbf{w}^\top \mathbf{x}_n)^{y_n} (1 - f(\mathbf{w}^\top \mathbf{x}_n))^{(1-y_n)}$$

$$\mathbf{w}^{\text{ML}} = \arg\max_{\mathbf{w}} \log P(\mathcal{Y}|\mathbf{w}, \mathcal{X}) = \arg\max_{\mathbf{w}} \sum_{n=1}^{N} \left[y_n \log f(\mathbf{w}^\top \mathbf{x}_n) + (1 - y_n) \log(1 - f(\mathbf{w}^\top \mathbf{x}_n)) \right]$$

Logistic regression: Maximum Likelihood learning

observe data: estimate parameters



model:

$$f(\mathbf{x}, \mathbf{w}) = P(y = 1|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1 + \exp(-(w_0 + w_1 x_1))}$$

data:

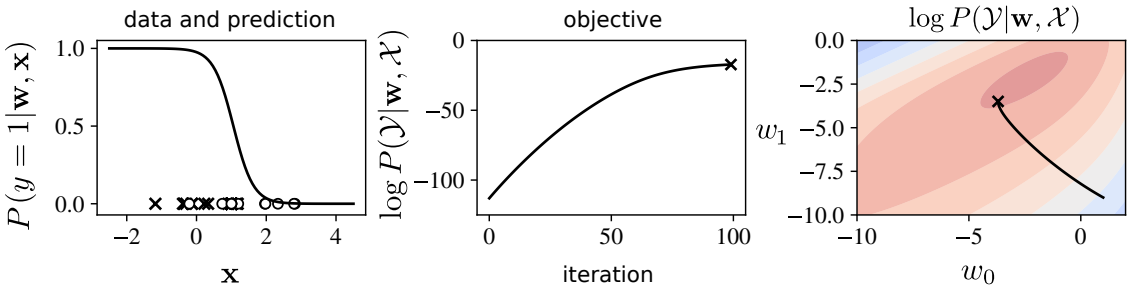
$$\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^N$$
$$\mathcal{Y} = \{y_n\}_{n=1}^N$$

maximum likelihood estimate: parameters that make observed data most probable

$$\mathbf{w}^{\text{ML}} = \arg\max_{\mathbf{w}} \log P(\mathcal{Y}|\mathbf{w}, \mathcal{X}) = \arg\max_{\mathbf{w}} \sum_{n=1}^{N} \left[y_n \log f(\mathbf{w}^\top \mathbf{x}_n) + (1 - y_n) \log(1 - f(\mathbf{w}^\top \mathbf{x}_n)) \right]$$

Logistic regression: Maximum Likelihood learning

observe data: estimate parameters



model:

$$f(\mathbf{x},\mathbf{w}) = P(y=1|\mathbf{w},\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1+\exp(-(w_0+w_1 x_1))}$$

data:

$$\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^N$$
$$\mathcal{Y} = \{y_n\}_{n=1}^N$$

maximum likelihood estimate: parameters that make observed data most probable

$$\mathbf{w}^{\text{ML}} = \arg\max_{\mathbf{w}} \log P(\mathcal{Y}|\mathbf{w},\mathcal{X}) = \arg\max_{\mathbf{w}} \sum_{n=1}^{N} \left[y_n \log f(\mathbf{w}^\top \mathbf{x}_n) + (1-y_n)\log(1-f(\mathbf{w}^\top \mathbf{x}_n)) \right]$$

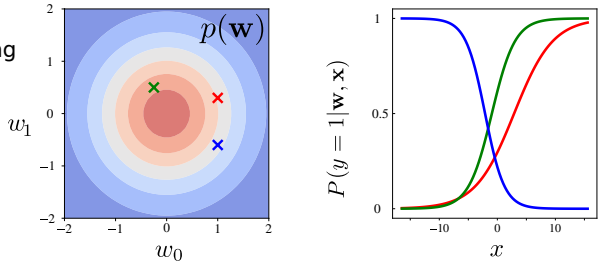
Bayesian approaches to logistic regression

Probabilistic model

encodes prior assumptions in a recipe for generating datasets

$$p(\mathbf{w}) = \mathcal{G}(\mathbf{w}; \mathbf{0}, \sigma_{\mathbf{w}}^2 \mathrm{I})$$

$$P(y=1|\mathbf{w},\mathbf{x}) = \frac{1}{1+\exp(-a(\mathbf{x},\mathbf{w}))}$$



Probabilistic inference

1. "right" answer is a probability distribution over all possible settings of the weights that indicates the plausiblity of that setting given data
2. apply the sum and product rules of probability to compute the plausibility of any setting of any unknown variable

only way to be coherent, Cox 1946

only way to protect against Dutch books, Ramsey 1926

implies

sum rule: $P(A|C) = \sum_B P(A,B|C)$

product rule: $P(A,B|C) = P(B|C)P(A|B,C) = P(A|C)P(B|A,C)$

Bayes' rule: $P(A|B,C) = \frac{1}{P(B|C)} P(A|C)P(B|A,C)$

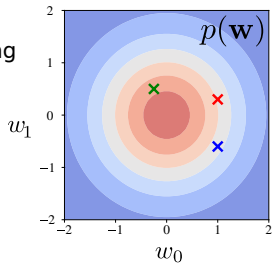
The posterior distribution over weights

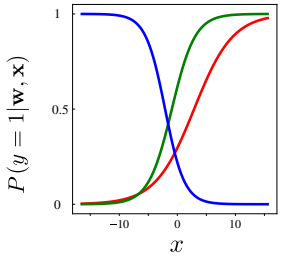
Probabilistic model

encodes prior assumptions in a recipe for generating datasets

$$p(\mathbf{w}) = \mathcal{G}(\mathbf{w}; \mathbf{0}, \sigma_{\mathbf{w}}^2 \mathrm{I})$$

$$P(y = 1|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-a(\mathbf{x}, \mathbf{w}))}$$





Probabilistic inference

structure of model

$A = \mathbf{w}$ $B = \mathcal{Y}$ $C = \mathcal{X}$

$$p(\mathbf{w}|\mathcal{Y}, \mathcal{X}) = \frac{1}{P(\mathcal{Y}|\mathcal{X})}\, p(\mathbf{w})\, P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$$

posterior ∝ prior × likelihood of $\mathbf{w}$

posterior: what we know after seeing data

prior: what we knew before seeing data

likelihood of $\mathbf{w}$: what the data told us

sum rule:	$P(A\|C) = \sum_B P(A, B\|C)$
product rule:	$P(A, B\|C) = P(B\|C)P(A\|B, C) = P(A\|C)P(B\|A, C)$
Bayes' rule:	$P(A\|B, C) = \frac{1}{P(B\|C)} P(A\|C)P(B\|A, C)$

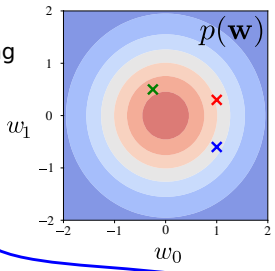
The predictive distribution over class labels

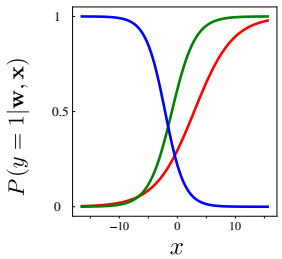
Probabilistic model

encodes prior assumptions in a recipe for generating datasets

$$p(\mathbf{w}) = \mathcal{G}(\mathbf{w}; \mathbf{0}, \sigma_{\mathbf{w}}^2 \mathrm{I})$$

$$P(y = 1|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-a(\mathbf{x}, \mathbf{w}))}$$





Probabilistic inference

$$p(\mathbf{w}|\mathcal{Y}, \mathcal{X}) = \frac{1}{P(\mathcal{Y}|\mathcal{X})}\, p(\mathbf{w})\, P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$$

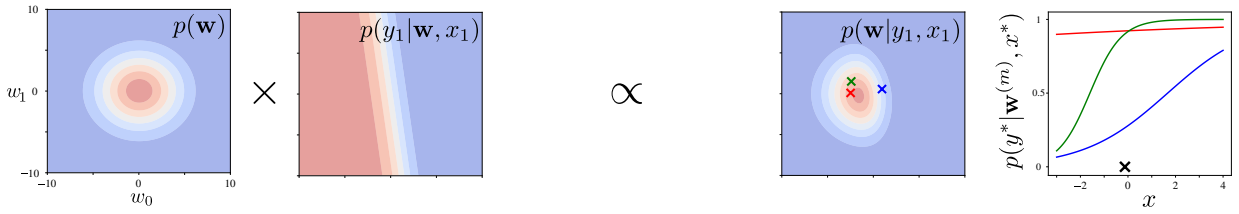
structure of model

$$p(y^*|x^*, \mathcal{Y}, \mathcal{X}) = \int p(y^*, \mathbf{w}|x^*, \mathcal{Y}, \mathcal{X}) \mathrm{d}\mathbf{w} = \int p(\mathbf{w}|\mathcal{Y}, \mathcal{X}) p(y^*|\mathbf{w}, x^*) \mathrm{d}\mathbf{w}$$

$A = y^*$ $B = \mathbf{w}$ $C = x^*, \mathcal{Y}, \mathcal{X}$

sum rule:	$P(A\|C) = \sum_B P(A, B\|C)$
product rule:	$P(A, B\|C) = P(B\|C)P(A\|B, C) = P(A\|C)P(B\|A, C)$
Bayes' rule:	$P(A\|B, C) = \frac{1}{P(B\|C)} P(A\|C)P(B\|A, C)$

Bayesian Inference in Action: 1D Classification Example



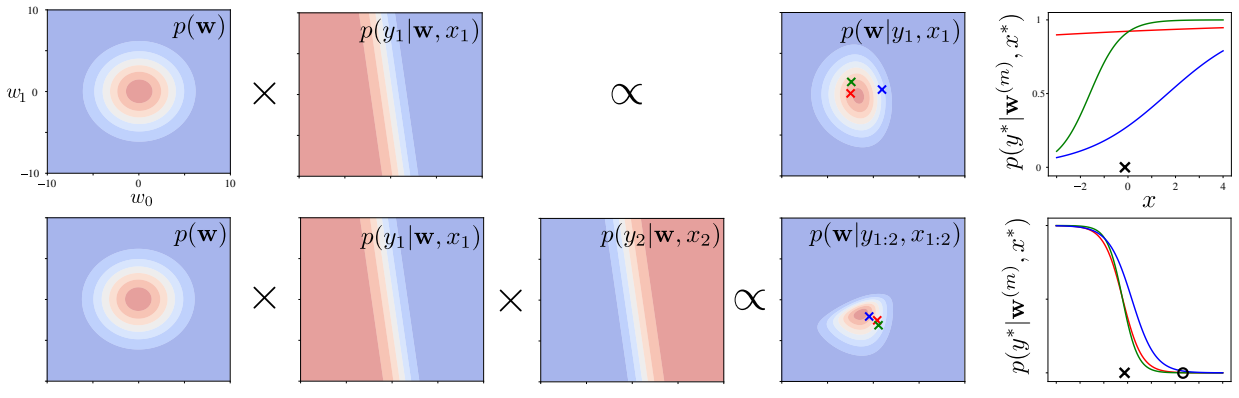
$p(\mathbf{w})$ × $P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$ ∝ $p(\mathbf{w}|\mathcal{Y}, \mathcal{X})$

prior
what we knew
before seeing data

likelihood of $\mathbf{w}$
what the
data told us

posterior
what we know
after seeing data

Bayesian Inference in Action: 1D Classification Example



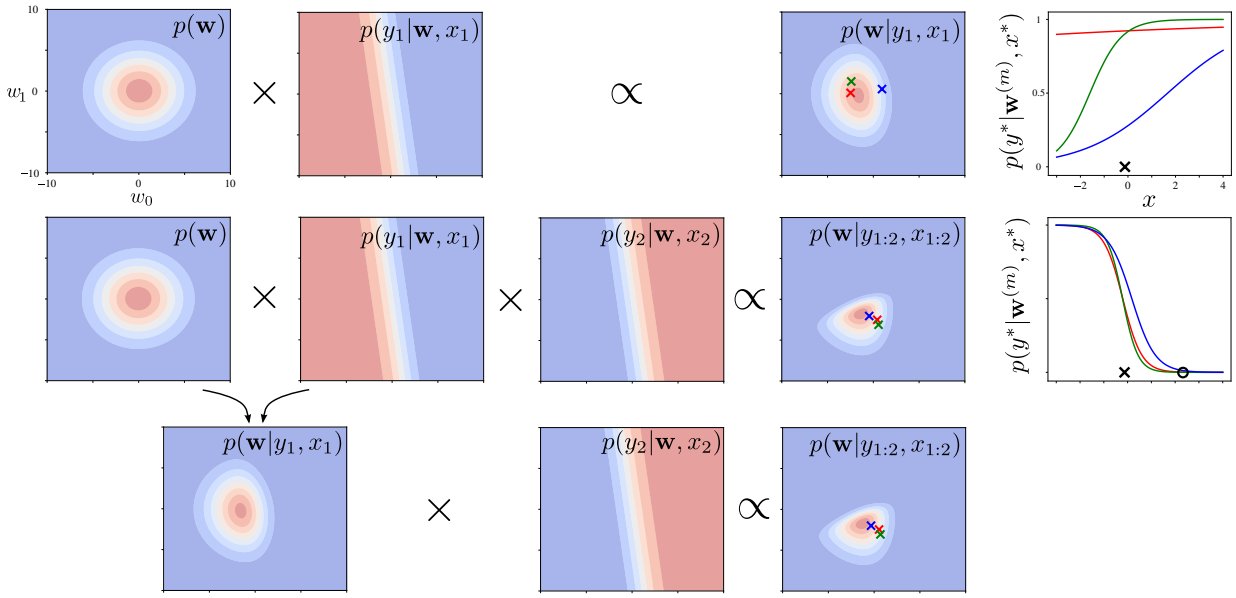
$p(\mathbf{w})$ × $P(\mathcal{Y}|\mathbf{w}, \mathcal{X})$ ∝ $p(\mathbf{w}|\mathcal{Y}, \mathcal{X})$

prior
what we knew
before seeing data

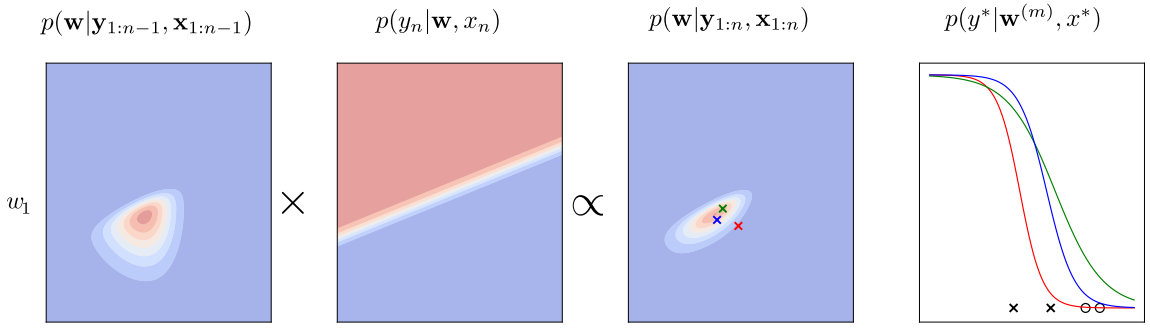
likelihood of $\mathbf{w}$
what the
data told us

posterior
what we know
after seeing data

Bayesian Inference in Action: 1D Classification Example



Bayesian Inference in Action: 1D Classification Example

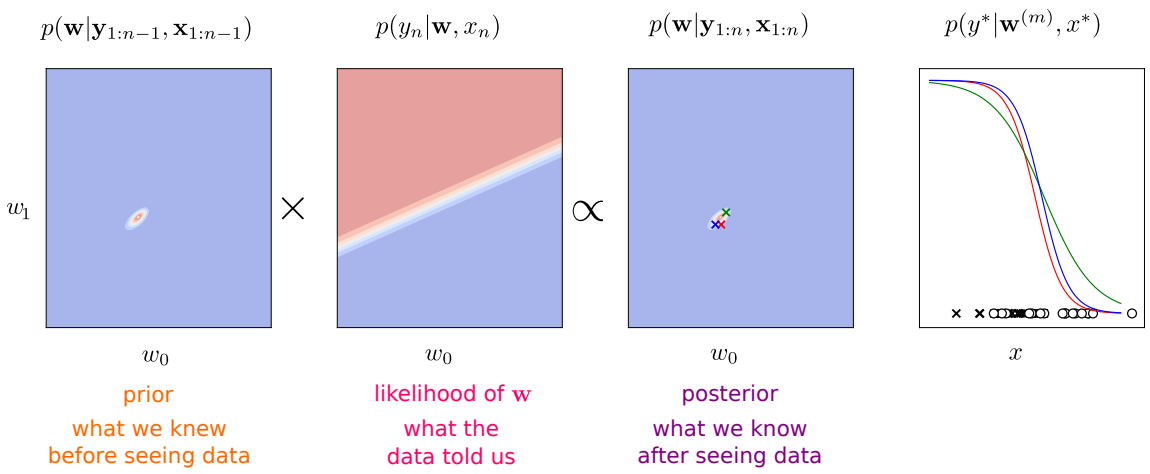


prior
what we knew before seeing data

likelihood of $\mathbf{w}$
what the data told us

posterior
what we know after seeing data

Bayesian Inference in Action: 1D Classification Example



prior
what we knew before seeing data

likelihood of $\mathbf{w}$
what the data told us

posterior
what we know after seeing data

Why be Bayesian (distributional estimates of weights)?

	maximum-likelihood	Bayes
learning	$\mathbf{w}^{\text{ML}} = \arg\max_{\mathbf{w}} \log P(\mathcal{Y}\|\mathbf{w}, \mathcal{X})$	$p(\mathbf{w}\|\mathcal{Y}, \mathcal{X}) = \frac{1}{P(\mathcal{Y}\|\mathcal{X})} p(\mathbf{w})\ P(\mathcal{Y}\|\mathbf{w}, \mathcal{X})$
prediction	$p(y^*\|x^*, \mathcal{Y}, \mathcal{X}) \approx p(y^*\|\mathbf{w}^{\text{ML}}, x^*)$	$p(y^*\|x^*, \mathcal{Y}, \mathcal{X}) = \int p(\mathbf{w}\|\mathcal{Y}, \mathcal{X}) p(y^*\|\mathbf{w}, x^*) \mathrm{d}\mathbf{w}$
	↑ single weight setting	↑ ensemble over weight settings; ↑ requires approximation

Robust Deep Learning

point estimates over-confident, averaging over weight settings less so
Bayesian methods are more robust to adversarial examples (hard to fool ensemble of networks + uncertainty)

Data-efficient Deep Learning

small data, big model: build models 'the size of a house' & let data prune/learn structure
leverage heterogeneous data sources (multi-task learning) using shared parameters

Flexibile Deep Learning

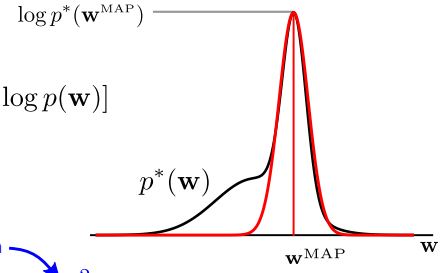
continual learning: use old posterior as prior
active learning: select data that are expected to reduce uncertainty in parameter estimates the most

Approximate Bayesian Inference

Laplace's approximation: MacKay 1991 (Saddle point approximation)

$$p(\mathbf{w}|\mathcal{Y},\mathcal{X}) \propto p(\mathcal{Y},\mathbf{w}|\mathcal{X}) = p^*(\mathbf{w})$$

$$\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{Y},\mathcal{X}) = \arg\max_{\mathbf{w}} \left[\log p(\mathcal{Y}|\mathbf{w},\mathcal{X}) + \log p(\mathbf{w})\right]$$



Taylor expand log-prob to 2nd order about MAP est:

$$\log p^*(\mathbf{w}) \approx \log p^*(\mathbf{w}^{\text{MAP}}) + \underbrace{(\mathbf{w}-\mathbf{w}^{\text{MAP}})^\top \frac{\mathrm{d}\log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}}}_{0} + \frac{1}{2}(\mathbf{w}-\mathbf{w}^{\text{MAP}})^\top \frac{\mathrm{d}^2\log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}^\top}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}} (\mathbf{w}-\mathbf{w}^{\text{MAP}})$$

Hessian

$$p^*(\mathbf{w}) \approx p^*(\mathbf{w}^{\text{MAP}}) \exp\left(\frac{1}{2}(\mathbf{w}-\mathbf{w}^{\text{MAP}})^\top \frac{\mathrm{d}^2\log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}^\top}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}} (\mathbf{w}-\mathbf{w}^{\text{MAP}})\right)$$

$$p^*(\mathbf{w}) \approx \underbrace{p^*(\mathbf{w}^{\text{MAP}})\det(2\pi\Sigma)^{1/2}}_{\approx p(\mathcal{Y}|\mathcal{X})} \underbrace{\mathcal{G}(\mathbf{w};\mathbf{w}^{\text{MAP}},\Sigma)}_{q(\mathbf{w})} \qquad \Sigma^{-1} = -\frac{\mathrm{d}^2\log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}^\top}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}}$$

simple Monte Carlo

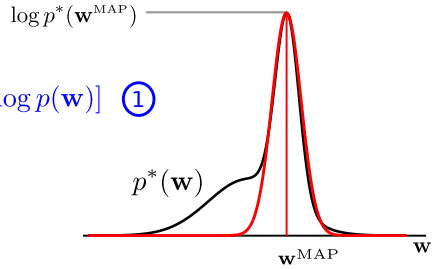
$$\mathbf{w}^{(m)} \sim q(\mathbf{w})$$

Prediction requires additional approx.: $p(y^*|\mathbf{x}^*,\mathcal{X},\mathcal{Y}) = \int p(y^*|\mathbf{w},\mathbf{x}^*)q(\mathbf{w})\mathrm{d}\mathbf{w} \approx \frac{1}{M}\sum_{m=1}^{M} p(y^*|\mathbf{w}^{(m)},\mathbf{x}^*)$

Laplace's approximation: MacKay 1991 (Saddle point approximation)

$$p(\mathbf{w}|\mathcal{Y},\mathcal{X}) \propto p(\mathcal{Y},\mathbf{w}|\mathcal{X}) = p^*(\mathbf{w})$$

$$\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{Y},\mathcal{X}) = \arg\max_{\mathbf{w}} \left[\log p(\mathcal{Y}|\mathbf{w},\mathcal{X}) + \log p(\mathbf{w})\right] \quad (1)$$



1. optimise to find MAP solution

   might not converge → Hessian not positive definite

2. compute Hessian at optimum to form approximate posterior and marginal likelihood

   cubic cost in the number of parameters

3. predict using local linearisation of network about MAP weights

$$p^*(\mathbf{w}) \approx \underbrace{p^*(\mathbf{w}^{\text{MAP}})\det(2\pi\Sigma)^{1/2}}_{\approx p(\mathcal{Y}|\mathcal{X})} \underbrace{\mathcal{G}\left(\mathbf{w};\mathbf{w}^{\text{MAP}},\Sigma\right)}_{q(\mathbf{w})}$$

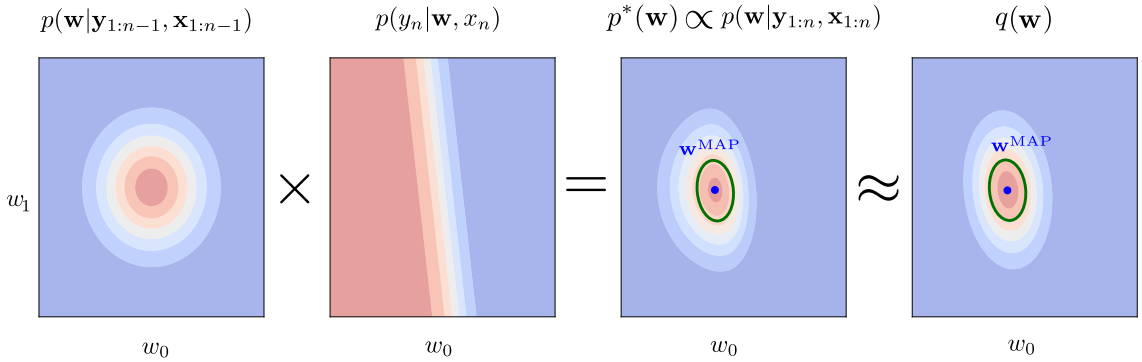
(2)
$$\Sigma^{-1} = -\frac{\mathrm{d}^2 \log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}^\top}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}}$$

simple Monte Carlo

$$\mathbf{w}^{(m)} \sim q(\mathbf{w})$$

(3)
$$\text{Prediction requires additional approx.: } p(y^*|\mathbf{x}^*,\mathcal{X},\mathcal{Y}) = \int p(y^*|\mathbf{w},\mathbf{x}^*)q(\mathbf{w})\mathrm{d}\mathbf{w} \approx \frac{1}{M}\sum_{m=1} p(y^*|\mathbf{w}^{(m)},\mathbf{x}^*)$$

Laplace's approximation: Classification Example



prior
what we knew before seeing data

likelihood of $\mathbf{w}$
what the data told us

$\propto$ posterior
what we know after seeing data

Laplace approx.
$\mathcal{G}\left(\mathbf{w};\mathbf{w}^{\text{MAP}},\Sigma\right)$

(1) $\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{Y},\mathcal{X})$

(2) $\Sigma^{-1} = -\frac{\mathrm{d}^2 \log p^*(\mathbf{w})}{\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}^\top}\Big|_{\mathbf{w}=\mathbf{w}^{\text{MAP}}}$

Variational Inference: the KL Divergence

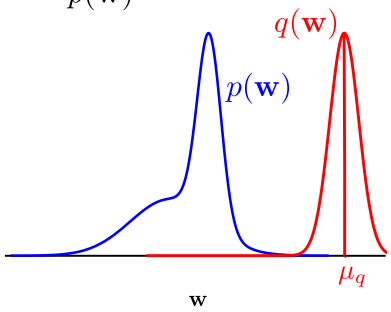
Kullback–Leibler (KL) divergence $\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w})} \mathrm{d}\mathbf{w}$

1. non-negative

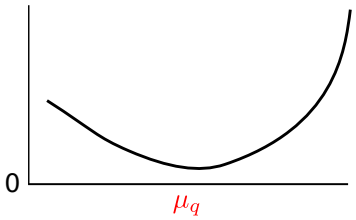
$$\frac{\delta^2}{\delta q^2} \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \frac{1}{q(\mathbf{w})} \geq 0$$

2. zero (minimised) when $q(\mathbf{w}) = p(\mathbf{w})$

$$\mathrm{KL}(p(\mathbf{w})||p(\mathbf{w})) = \int p(\mathbf{w}) \log \frac{p(\mathbf{w})}{p(\mathbf{w})} \mathrm{d}\mathbf{w} = 0$$



$\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}))$



Variational Inference: the KL Divergence

Kullback–Leibler (KL) divergence $\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w})} \mathrm{d}\mathbf{w}$

1. non-negative

$$\frac{\delta^2}{\delta q^2} \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \frac{1}{q(\mathbf{w})} \geq 0$$

2. zero (minimised) when $q(\mathbf{w}) = p(\mathbf{w})$

$$\mathrm{KL}(p(\mathbf{w})||p(\mathbf{w})) = \int p(\mathbf{w}) \log \frac{p(\mathbf{w})}{p(\mathbf{w})} \mathrm{d}\mathbf{w} = 0$$

divergence measures 'distance' between distributions

3. can be computed up to an additive constant w/o needing normalisation for $p(\mathbf{w})$

$$\mathrm{KL}\left(q(\mathbf{w})||\frac{1}{Z}p^*(\mathbf{w})\right) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{\frac{1}{Z}p^*(\mathbf{w})} \mathrm{d}\mathbf{w} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})Z}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w}$$

$$= \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w} + \int q(\mathbf{w}) \log Z \mathrm{d}\mathbf{w} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w} + \log Z$$

→ suitable for approximate inference

Variational Inference

Kullback–Leibler (KL) divergence $\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w})} \mathrm{d}\mathbf{w} \overset{1\&2}{\geq} 0$ equality when $q(\mathbf{w}) = p(\mathbf{w})$

Use KL to optimise an approximation $q(\mathbf{w})$ to the true posterior

$$q^{\mathrm{VI}}(\mathbf{w}) \overset{1\&2}{=} \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathcal{Y}, \mathcal{X})) \overset{3}{=} \arg\min_{q \in \mathcal{Q}} \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w})p(\mathcal{Y}|\mathbf{w}, \mathcal{X})} \mathrm{d}\mathbf{w}$$

$$\approx \arg\min_{q \in \mathcal{Q}} \frac{1}{M} \sum_{m=1}^{M} \log \frac{q(\mathbf{w}^{(m)})}{p(\mathbf{w}^{(m)})p(\mathcal{Y}|\mathbf{w}^{(m)}, \mathcal{X})} \qquad \mathbf{w}^{(m)} \sim q(\mathbf{w})$$

requires reparameterisation trick

3. can be computed up to an additive constant w/o needing normalisation for $p(\mathbf{w})$

$$\mathrm{KL}\left(q(\mathbf{w})||\frac{1}{Z}p^*(\mathbf{w})\right) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{\frac{1}{Z}p^*(\mathbf{w})} \mathrm{d}\mathbf{w} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})Z}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w}$$

$$= \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w} + \int q(\mathbf{w}) \log Z \mathrm{d}\mathbf{w} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p^*(\mathbf{w})} \mathrm{d}\mathbf{w} + \log Z$$

→ suitable for approximate inference

Variational Inference: Classification Example

$$q^{\mathrm{VI}}(\mathbf{w}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathcal{Y}, \mathcal{X}))$$

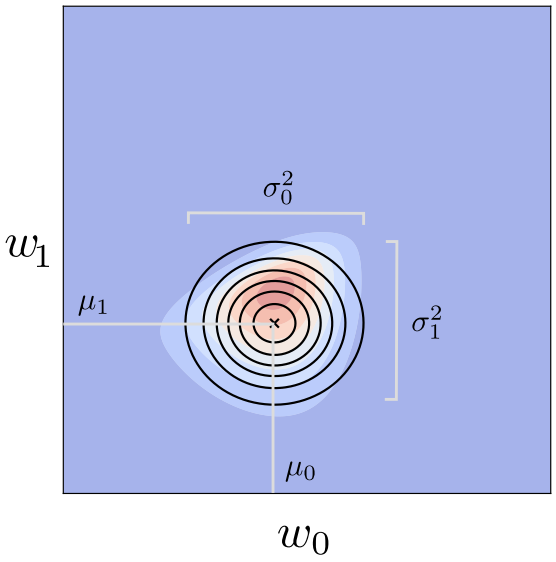
$$\approx \arg\min_{q \in \mathcal{Q}} \frac{1}{M} \sum_{m=1}^{M} \log \frac{q(\mathbf{w}^{(m)})}{p(\mathbf{w}^{(m)})p(\mathcal{Y}|\mathbf{w}^{(m)}, \mathcal{X})}$$

approximate family: factorised Gaussian

$$q(\mathbf{w}) = \mathcal{G}(w_0; \mu_0, \sigma_0^2) \times \mathcal{G}(w_1; \mu_1, \sigma_1^2)$$

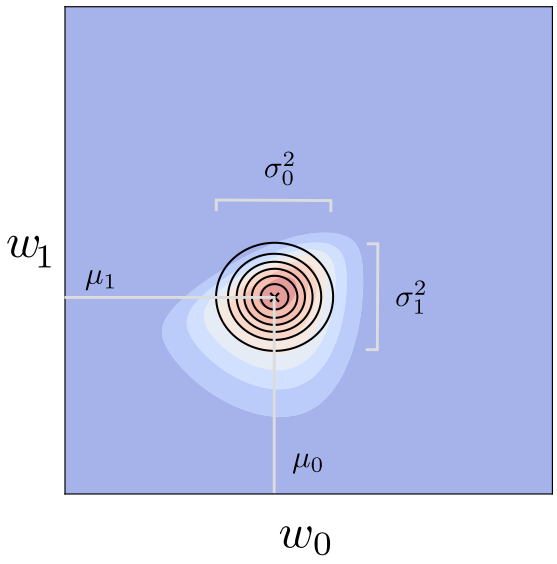
optimise w.r.t. $\mu_0, \sigma_0^2, \mu_1, \sigma_1^2$

VI converts inference into optimization

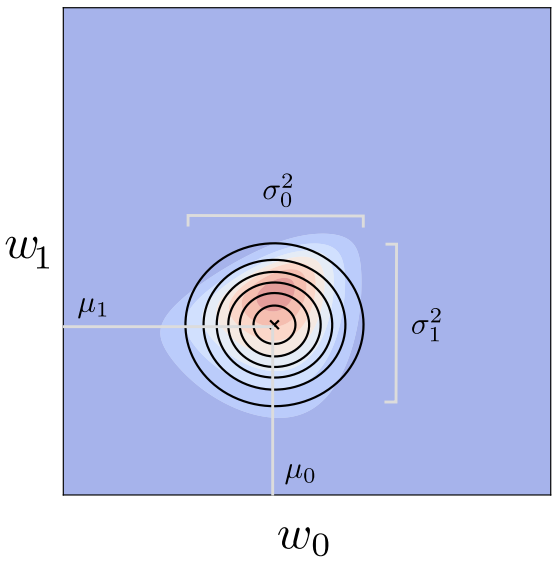


Variational Inference vs Laplace: Classification Example

Laplace Approximation (diagonal Hessian)



Variational Inference (factorised)

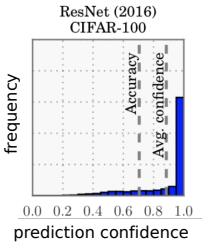


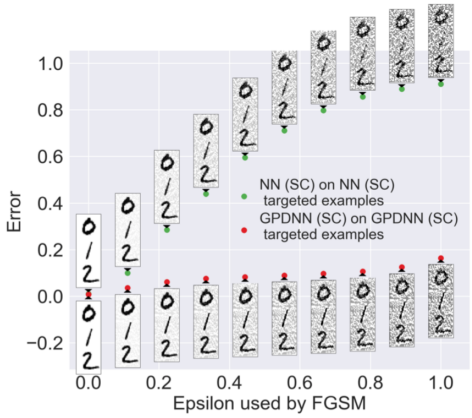
Case study 1. Robust Deep Learning
Uncertainty calibration & adversarial examples

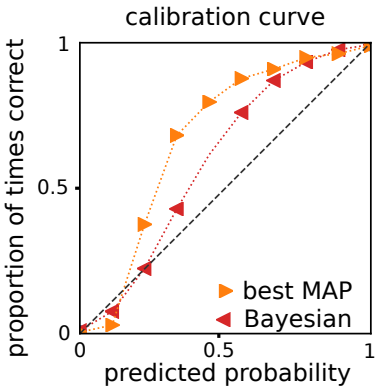
Towards Robust Deep Learning

fragile (adversarial examples)

well calibrated uncertainty estimates: deep learning is often confidently wrong



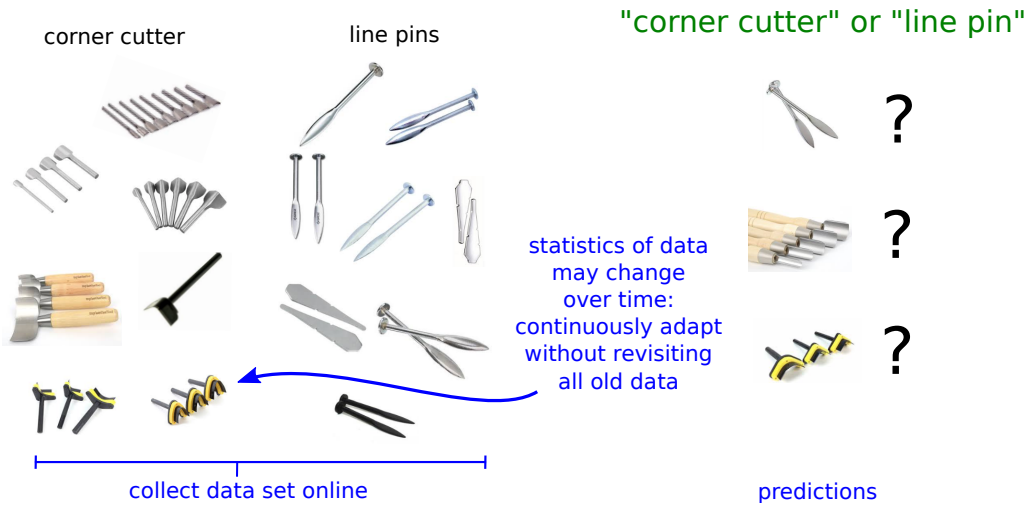




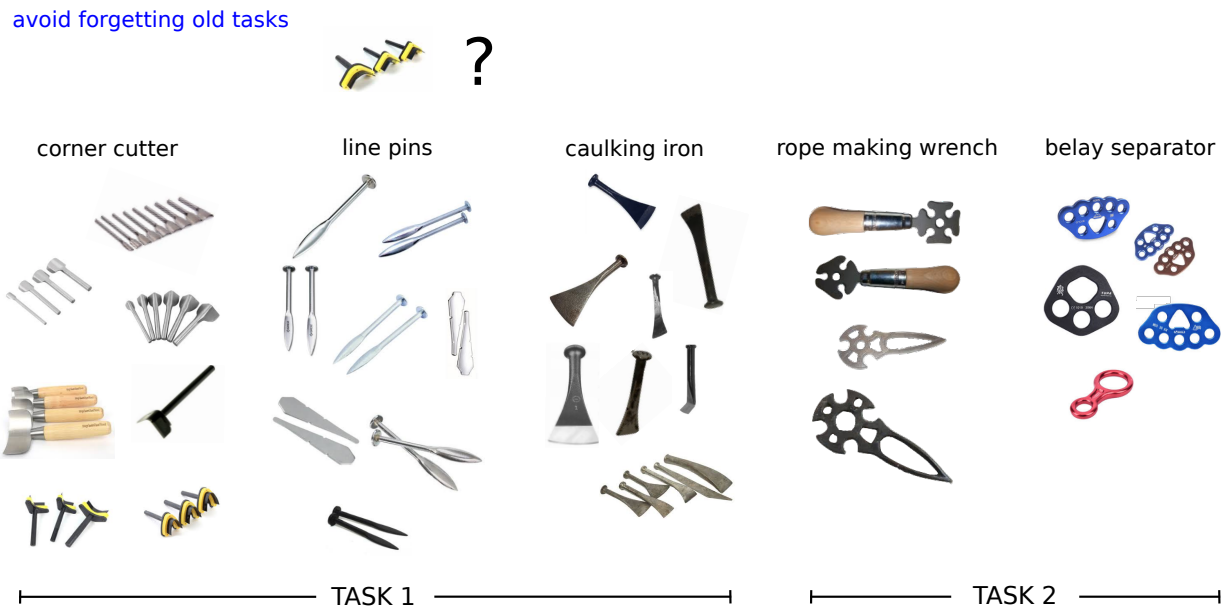
Case study 2. Flexible Deep Learning

Continual multi-task learning

What is Continual Learning?



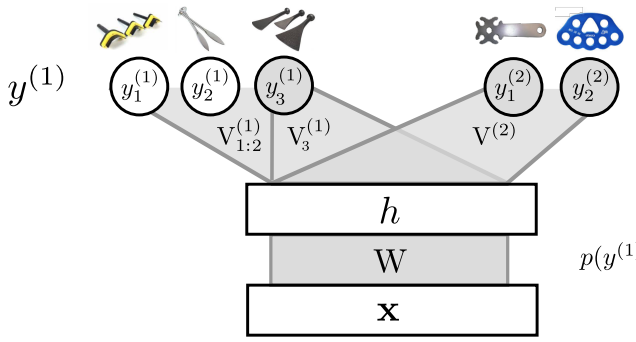
What is Continual Learning?



A zoo of discriminative continual learning tasks

3. new classes
(k-shot learning)

4. new tasks
(online multi-task learning
transfer learning)



$p(y^{(1)}, y^{(2)} | \mathrm{V}^{(1)}, \mathrm{V}^{(2)}, \mathrm{W}, \mathbf{x})$

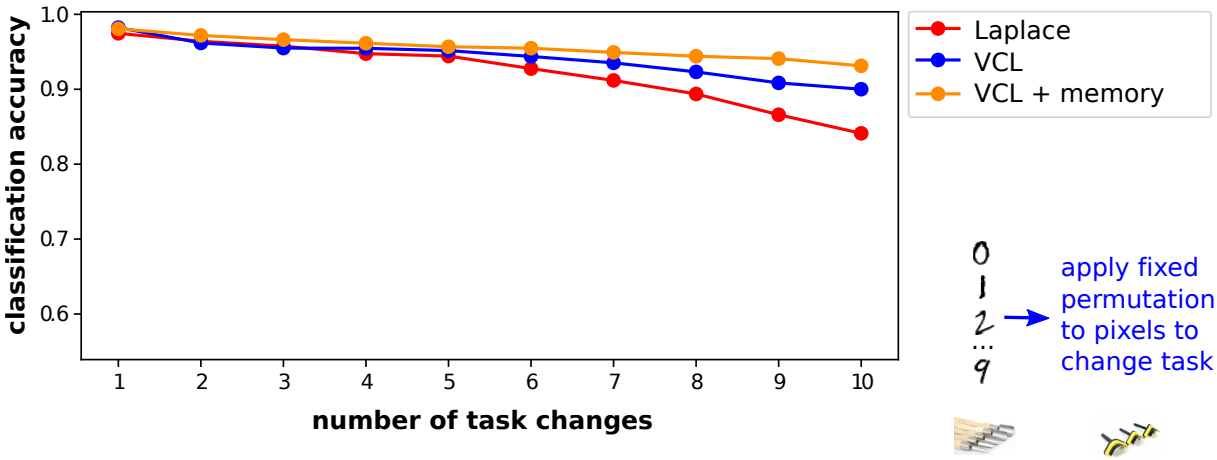
1. online iid data
(online learning)

$p(\mathbf{x}_{1:N}) = \prod_{n=1}^{N} p(\mathbf{x}_n)$

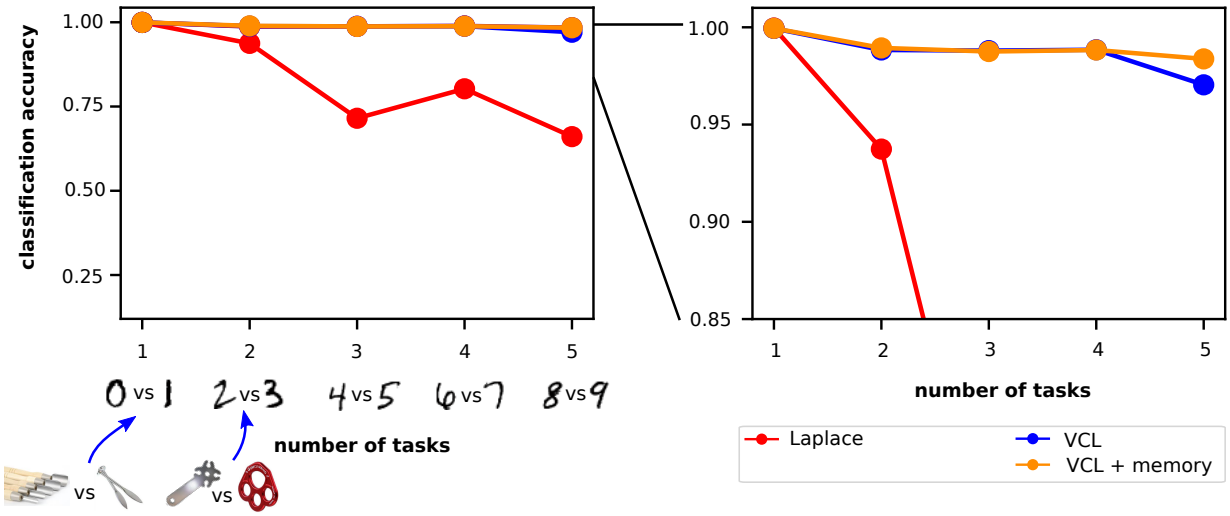
2. online non-iid inputs
(covariate shift)

$p(\mathbf{x}_{1:N}) = \prod_{n=1}^{N} p_n(\mathbf{x}_n | \mathbf{x}_{1:n-1})$

Continual Learning Test 1: Permuted MNIST (online non-iid inputs, single head)



Continual Learning Test 2: Split MNIST (new tasks, multi-head)



Summary

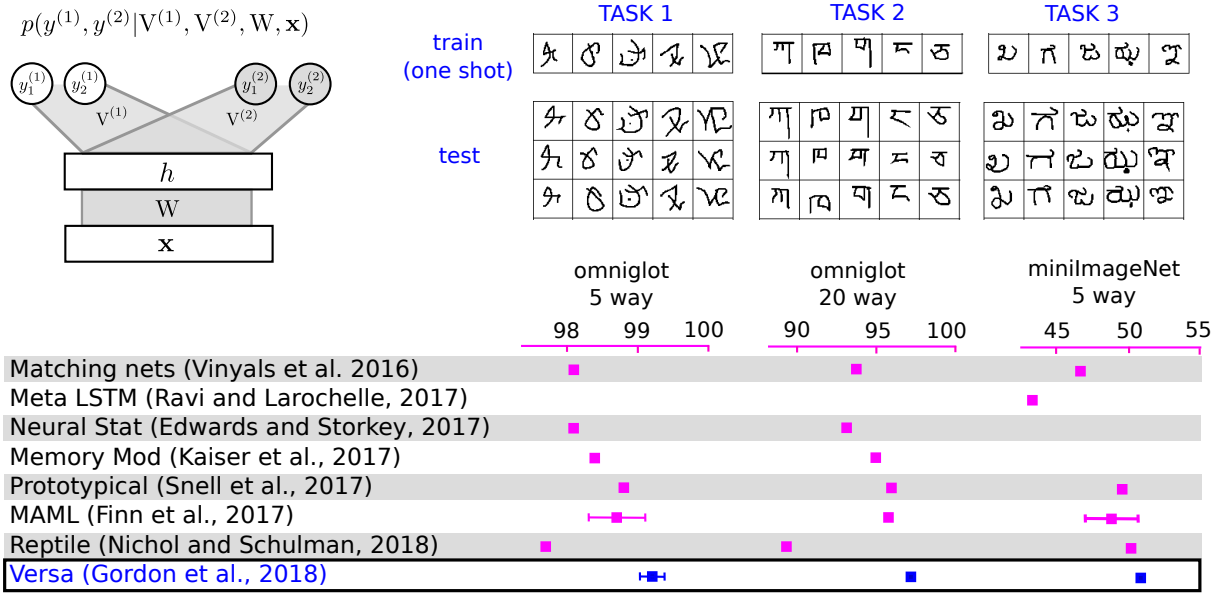
- **Continual learning** is naturally handled by **Bayesian inference**: allows multi-task transfer and avoids catastrophic forgetting
- **Variational Continual Learning** is a **state-of-the-art continual learning method**
- Orthogonal research directions: **complex models** (adapting more than just the head of the network) and **online automatic model building**

Variational Continual Learning, ICLR 2018
Streaming Sparse Gaussian Process Approximations, NIPS 2017

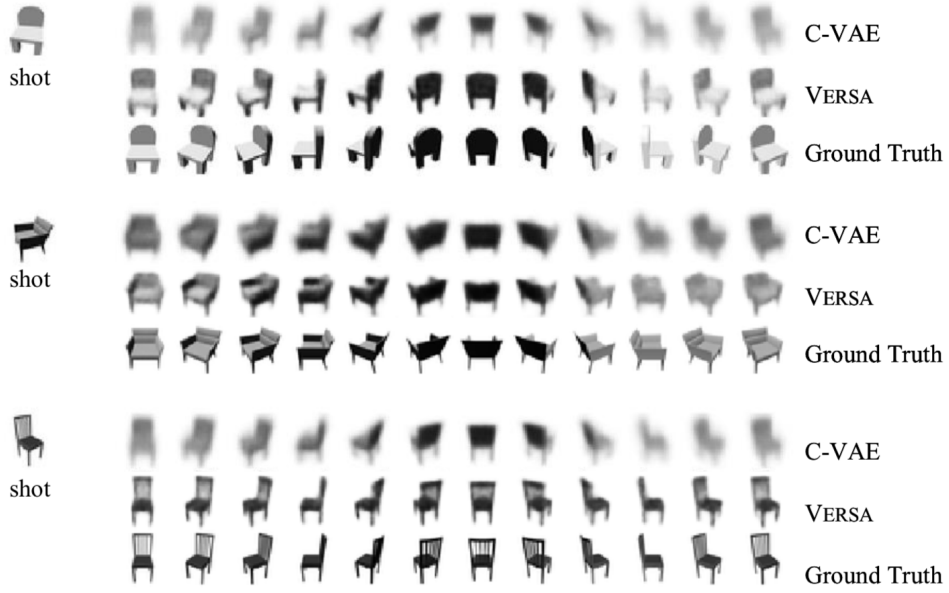
Case study 3: Data Efficient Deep Learning

One-shot learning

One shot learning using Approximate Bayesian inference



One shot learning using Approximate Bayesian inference



One shot learning using Approximate Bayesian inference