Bayesian Deep Learning (MLSS 2019)

Yarin Gal

University of Oxford
yarin@cs.ox.ac.uk

Reminder

Model

- prior

$$p(w_{k,d}) = \mathcal{N}(w_{k,d}; 0, s^2); \quad W \in \mathbb{R}^{K \times D}$$

- likelihood

$$p(\mathbf{Y}|\mathbf{X}, W) = \prod_n \mathcal{N}(y_n; f^W(x_n), \sigma^2); \quad f^W(x) = W^T \phi(x)$$

- with $\phi(x)$ a K dim feature vector

Posterior

$$p(W|X, Y) = \mathcal{N}(W; \mu', \Sigma')$$

$$\Sigma' = (\sigma^{-2}\Phi(X)^T\Phi(X) + s^{-2}I_K)^{-1}$$

$$\mu' = \Sigma'\sigma^{-2}\Phi(X)^T Y$$

Predictive

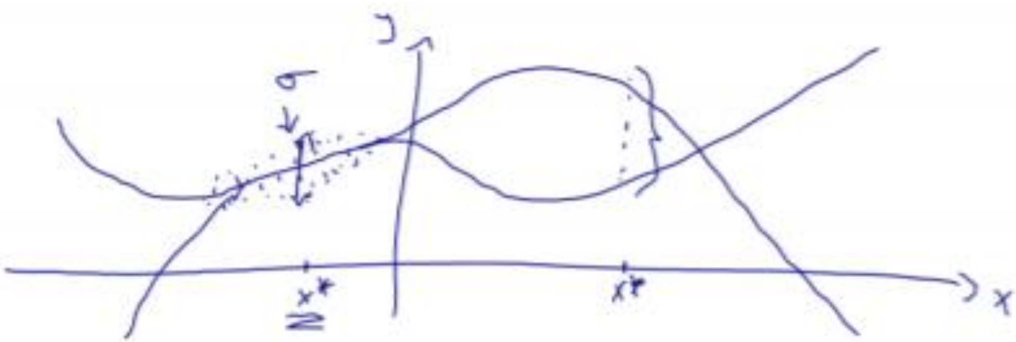
$$p(y^*|x^*, X, Y) = \mathcal{N}(y^*; \mu'^T\phi(x^*), \sigma^2 + \phi(x^*)^T\Sigma'\phi(x^*))$$

Decomposing uncertainty

$$p(y^*|x^*, X, Y) = \mathcal{N}(y^*; \mu'^T \phi(x^*),$$
$$\sigma^2 + \phi(x^*)^T \Sigma' \phi(x^*))$$

Uncertainty has two components:

- σ^2 – from likelihood
- $\phi(x^*)^T \Sigma' \phi(x^*)$ – from posterior



Aleatoric uncertainty

- first term in predictive uncertainty
  $\sigma^2 + \phi(x^*)^T \Sigma' \phi(x^*)$
- same as likelihood σ^2 – obs noise / corrupting additive noise eg measurement error
- can be found via MLE rather than assume known in advance (we'll see later)
- from Latin aleator 'dice player', from alea 'die'
  - roll a pair of dice again and again – will not reduce uncertainty

Epistemic uncertainty

- second term in predictive uncertainty $\sigma^2 + \phi(x^*)^T \Sigma' \phi(x^*)$
- uncertainty over **function values** before noise corruption

$$f^* = W^T \phi(x^*)$$

$$\mathrm{Var}_{p(f^*|x^*,X,Y)}[f^*] = \phi(x^*)^T \Sigma' \phi(x^*)$$

- high for x^* "far away" from the data, even in noiseless case (ie likelihood noise is zero)
- will diminish given label for x^*
- from Ancient Greek episteme 'knowledge, understanding'

Approximate variational inference

- to evaluate predictive need to invert post cov matrix – a K by K matrix
  - difficult when K is large...
- instead, let's try to approximate posterior w a simpler dist to allow easier computations
- in approx inference we approx posterior $p(W|X,Y)$ w a different dist $q_\theta(W)$ param by theta
  - q also called "variational distribution"
  - θ also called "variational params"
  - technique is also known as "variational inference (VI)"
- eg q Gaussian w params $\theta = \{\mu_{\text{VI}}, \Sigma_{\text{VI}}\}$
  - $q_\theta(W) = \mathcal{N}(W; \mu_{\text{VI}}, \Sigma_{\text{VI}})$
  - often omit θ from subscript to avoid clutter, write $q(W)$ or q
  - often swap θ for μ, Σ back and forth

Underlying principle of VI

- eg: I have posterior $p(W|X, Y) = \mathcal{N}(0, 1)$; I give you 2 approx dists

$$q_1(W) = \mathcal{N}(10, 1), \qquad q_2(W) = \mathcal{N}(0, 10)$$

- which would you choose? and now? ... and now?
  - the one that gives best preds?
  - will fail: best preds are at $\mu = \mu_{\text{MLE}}, \Sigma = 0$
- need some measure of how "similar" dists are to posterior...
  - choose a measure of "similarity" between dists $\tilde{D}$ (not necessarily a distance!)
  - then min whatever measure we commit to
  - ie if $\tilde{D}(q_1, \text{posterior}) < \tilde{D}(q_2, \text{posterior})$ then the core principle of VI says that q_1 **should be chosen over** q_2

Underlying principle of VI

- $\tilde{D}(q_1, \text{posterior}) < \tilde{D}(q_2, \text{posterior}) \rightarrow q_1$ **should be chosen over** q_2
  - what if we have two divergences $\tilde{D}_1$ and $\tilde{D}_2$, one saying to select q_1 and the other q_2?

  ! a difference to full Bayesian inference... (where there's only one way of doing things 'correctly')

  - "from **dogmatic** Bayes to **pragmatic** Bayes";
  - often choose $\tilde{D}$ that is mathematically convenient
- eg Kullback Leibler

$$\mathrm{KL}(q, p) = \int q(x) \log \frac{q(x)}{p(x)} dx$$

KL properties (eg w discrete distributions)

- K dim discrete prob vectors q, p: $\mathrm{KL}(q,p) = \sum_k q_k \log q_k/p_k$
- when the two dists are the same we get exactly 0
- when the two dists are different the divergence is positive
- KL is not symmetric
- if q_k is zero it is ignored in KL
- whenever $q_k > 0$ it must be that $p_k > 0$ for the KL to be finite
- Homework: find examples for all properties; eg $q = [1/8, 3/8, 4/8]$, $p = [3/8, 4/8, 1/8]$;

KL for cnts rvs

What if we want to approx cnts rv like W?

- $q(x) = N(x; \mu_0, s_0^2)$, $p(x) = N(x; \mu_1, s_1^2)$; KL for Gaussians:

$$\mathrm{KL}(q, p) = 1/2(s_1^{-2} s_0^2 + s_1^{-2}(\mu_1 - \mu_0)^2 - 1 + \log(s_1^2/s_0^2))$$

- nice property: if X_1 and X_2 are independent under p and q then

$$\mathrm{KL}(q(X_1, X_2), p(X_1, X_2)) = \mathrm{KL}(q(X_1), p(X_1)) + \mathrm{KL}(q(X_2), p(X_2))$$

- multivariate diagonal Gaussians (K dims):
  write $\mathbf{x} = [x_1, .., x_K]$

$$q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_0, S_0) \qquad \text{with } S_0 = \mathrm{diag}([s_{01}^2, ..., s_{0K}^2])$$
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_1, S_1) \qquad \text{with } S_1 = \mathrm{diag}([s_{11}^2, ..., s_{1K}^2])$$

Then from indep of $x_1, .., x_K$:

$$\mathrm{KL}(q, p) = \sum_k 1/2(s_{1k}^{-2} s_{0k}^2 + s_{1k}^{-2}(\mu_{1k} - \mu_{0k})^2 - 1 + \log(s_{1k}^2/s_{0k}^2))$$

KL for approx inference

- want to approx $p(W|X,Y)$ using some $q_\theta(W)$
- min
$$\mathrm{KL}(q_\theta(W), p(W|X,Y))$$
wrt θ (remember def $\mathrm{KL}(q,p) = \int q(x) \log \frac{q(x)}{p(x)} dx$)
- $\log p(Y|X) \geq \int q(W) \log p(Y|X,W) \mathrm{d}W - \mathrm{KL}(q(W), p(W))$
  - pops out a bound on evidence for free
  - also called "evidence lower bound" (**ELBO**)
  - min KL to posterior = max ELBO
- what does it mean to max ELBO?
  - **first term**: how well we "explain the data"; if possible, q should put all mass at MLE!
  - **second term**: how close we are to the prior (get simplest q that can still explain data well); if possible, q should be prior itself!

KL for approx inference

- max

$$\int q_\theta(W) \log p(Y|X, W) \mathrm{d}W - \mathrm{KL}(q_\theta(W), p(W))$$

  wrt θ

- which terms can we compute?
  - for Gaussian prior and q, can compute KL to prior
  - for Gaussian lik can compute expected log lik as well (analytic – try this at home using tools from earlier!)
  - but in more complicated likelihoods (like in **classification**) can't eval above...
  - for this we'll look at **stochastic** approximate inference

Classification NN

Let's try to do a classification task

- want to get notion of epistemic uncertainty in classification
- generative story
  - Nature chose function $p(x) : \mathbb{R}^Q \to [0, 1]^C$
  - $p(x)$ a prob vector as a function of x
  - eg p softmax func
  - for $n = 1..N$ generate label $y_n \sim \text{Categorical}(p(x_n))$

$f^w(x)_1$ $f^w(x)_c$

$x = [x_1 \ldots x_c]$

$$\text{softmax}(\underline{x}) = \left[\frac{e^{x_1}}{\sum_i e^{x_i}}, \ldots, \frac{e^{x_c}}{\sum_i e^{x_i}} \right]$$

- encode y_n as a one hot vector $\mathbf{y}_n$ (eg $[0, 0, 1, 0]$ with $C = 4$ classes and $y_n = 2$)

Classification NN

UNIVERSITY OF OXFORD

Model:

- likelihood
  - model prob func by **function** $p^W(x)$ with W a K by C matrix; then lik is **def'd** as elem c in prob vec

$$p(y = c|x, W) = p^W(x)_c,$$

$$p(Y|X, W) = \prod_n p^W(x_n)_{y_n=c}$$
$$= \prod_n \mathbf{y}_n^T p^W(x_n)$$

- prior over W
  - vectorise W (still write W instead of vec(W))
  - same prior as before: $p(W) = \mathcal{N}(W; 0_{CK}, s^2 I_{CK})$

For each c:

$w_c : k \times 1$

Write

$W = [w_1, \ldots w_C]$

$: k \times c$

Def "logits"

$f^W(x) =$

$[w_1^T \phi(x), \ldots$

$w_C^T \phi(x)]$

Def prob func

$p^W(x) =$

$\text{Softmax}(\underline{f^W(x)})$

Classification NN

Model:

- to do predictions

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

- need posterior. But product of softmax and Gaussian is not Gaussian, so can't use tricks from before.. for posterior need evidence:

$$p(Y|X) = \int \prod_n [\mathbf{y}_n^T \mathrm{softmax}(f^W(x_n)_1, ..f^W(x_n))] N(W; 0, s^2 I)\mathrm{d}W$$

can't integrate/sum explicitly.. will use VI instead to approx posterior

Approx inference in classification NN

- For approx inf need log lik of softmax$(f_1, .., f_C) = [\frac{e^{f_1}}{e^{f_1}+...+e^{f_C}}, ...]$

$$\log p(y = c|x, W) = f_c - \log(e^{f_1} + ... + e^{f_C})$$

with $[f_1, ..., f_C]$ the logits vector $[w_1^T \phi(x), .., w_C^T \phi(x)]$

- then expected log likelihood is

$$L(\theta) = \sum_{x_n, y_n = c} \int \left[f^W(x_n)_c - \log \left(\sum_{c'} e^{f^W(x_n)_{c'}} \right) \right] N(W; \mu_{\text{VI}}, \Sigma_{\text{VI}}) \mathrm{d}W \\ - \text{KL}(q, p)$$

with $f^W(x)_c = w_c^T \phi(x)$

- can't integrate analytically either (log sum exp); need new tools...

MC integration

Useful tool to estimate expectations

- let $p(x)$ be some dist which is easy to sample from
- let $f(x)$ be some function of x
- assume it to be difficult to eval $E := E_p[f(x)]$
- can use MC integration instead:
  - generate $\hat{x}_1, .., \hat{x}_T \sim p(x)$
  - estimate $\hat{E} := 1/T \sum_t f(\hat{x}_t)$
  - an estimator $\hat{E}$ of E is called *unbiased* if in expectation equals E
  - $\hat{E}$ is an unbiased estimator of E (prove at home!)

Integral derivative estimation

- We actually need an estimator of the **derivative of an integral**
- let $G(\theta)$ be the gradient of $L(\theta)$; will interchangeably use
  - G (grad of L)
  - $(L(\theta))' =$ derivative of L wrt θ
  - $\frac{\partial}{\partial W(\theta)} L(W(\theta)) =$ derivative of L wrt $W(\theta)$
- if had unbiased derivative estimator $\hat{G}(\theta)$ (estimator of $G(\theta)$) can use a stochastic iterative method to optimise $L(\theta)$:

$$\theta_{n+1} \leftarrow \theta_n + \frac{1}{n}\hat{G}(\theta)$$

  go in direction of steepest ascent, on average
  - this is called stochastic gradient descent (well, ascent here)
  - SGD

Example of integral derivative estimation

- $L(\mu, \sigma) = \int (W + W^2) N(W; \mu, \sigma^2) \mathrm{d}W$
  - can actually eval analytically as $L = \mu + \sigma^2 + \mu^2$
  - so integral derivative is $G(\mu) = 1 + 2\mu$; will write $G(\mu) := \partial L / \partial \mu$
- Let's try MC integration first – $\hat{L}(\hat{W}; \mu, \sigma) = \hat{W} + \hat{W}^2$ with realisations (numbers) $\hat{W} \sim \mathcal{N}(\mu, \sigma^2)$, so

$$\hat{G}(\mu) = \partial(\hat{W} + \hat{W}^2)/\partial\mu \stackrel{?}{=} 0$$

(no μ in $\hat{L}$)

  - but $\hat{L}$ clearly depends on μ;
  - eg increasing μ increases expectation of $\hat{L}$
  - doesn't look correct... what's going on?

Example of integral derivative estimation

- $\hat{L}$ deps on μ through $\hat{W}$; $\hat{W}$ is actually a function of μ as well as a rv $\hat{\epsilon}$ indep of θ:

$$\hat{W} \sim \mathcal{N}(\mu, \sigma^2) \qquad \leftrightarrow \qquad \hat{W} = \hat{W}(\theta, \hat{\epsilon}) = \mu + \sigma\hat{\epsilon}; \quad \hat{\epsilon} \sim \mathcal{N}(0, 1)$$

- then can rewrite $\hat{L}$ as

$$\hat{L}(\mu, \sigma, \hat{\epsilon}) = (\mu + \sigma\hat{\epsilon}) + (\mu + \sigma\hat{\epsilon})^2 = \mu + \mu^2 + 2\mu\sigma\hat{\epsilon} + \sigma\hat{\epsilon} + \sigma^2\hat{\epsilon}^2$$

and

$$\hat{G}(\mu) = 1 + 2\mu + 2\sigma\hat{\epsilon}$$

- check:

$$E_{p(\epsilon)}[\hat{G}] = 1 + 2\mu = G$$

ie $\hat{G}$ is an unbiased estimator of G

Reparametrisation trick

- technique known in literature as the *re-parametrisation trick*
  - also known as a *pathwise derivative estimator*, *infinitesimal perturbation analysis*, and *stochastic backpropagation*

- in general:
  - given func $f(W)$, dist $q_\theta(W)$
  - want to estimate gradients of $L(\theta) = \int f(W) q_\theta(W) \mathrm{d}W$
  - if W can be reparam as $W = g(\theta, \epsilon)$ with ϵ not dependent on θ, and g is **differentiable wrt** θ
  - then $\hat{G}(\theta, \hat{\epsilon}) = f'(g(\theta, \hat{\epsilon})) \cdot \partial g(\theta, \hat{\epsilon}) / \partial \theta$
  - .. and plug into a stochastic optimiser

- eg, for Gaussian q...
  - $W = g([\mu, \sigma], \epsilon) = \mu + \sigma\epsilon$
  - $\partial g([\mu, \sigma], \epsilon)/\partial\mu = 1$ and $\partial g([\mu, \sigma], \epsilon)/\partial\sigma = \epsilon$
  - so $\hat{G}(\hat{\epsilon}; \mu) = f'(\mu + \sigma\hat{\epsilon}) \cdot 1$ and $\hat{G}(\hat{\epsilon}; \sigma) = f'(\mu + \sigma\hat{\epsilon})\hat{\epsilon}$
    - with $\hat{\epsilon} \sim \mathcal{N}(0, I)$
  - can substitute in $\hat{W} = \mu + \sigma\hat{\epsilon}$: sample $\hat{W} \sim q_\theta(W)$; then $\hat{G}(\hat{W}; \mu) = f'(\hat{W})$ and $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

Back to approx inference in classification NN

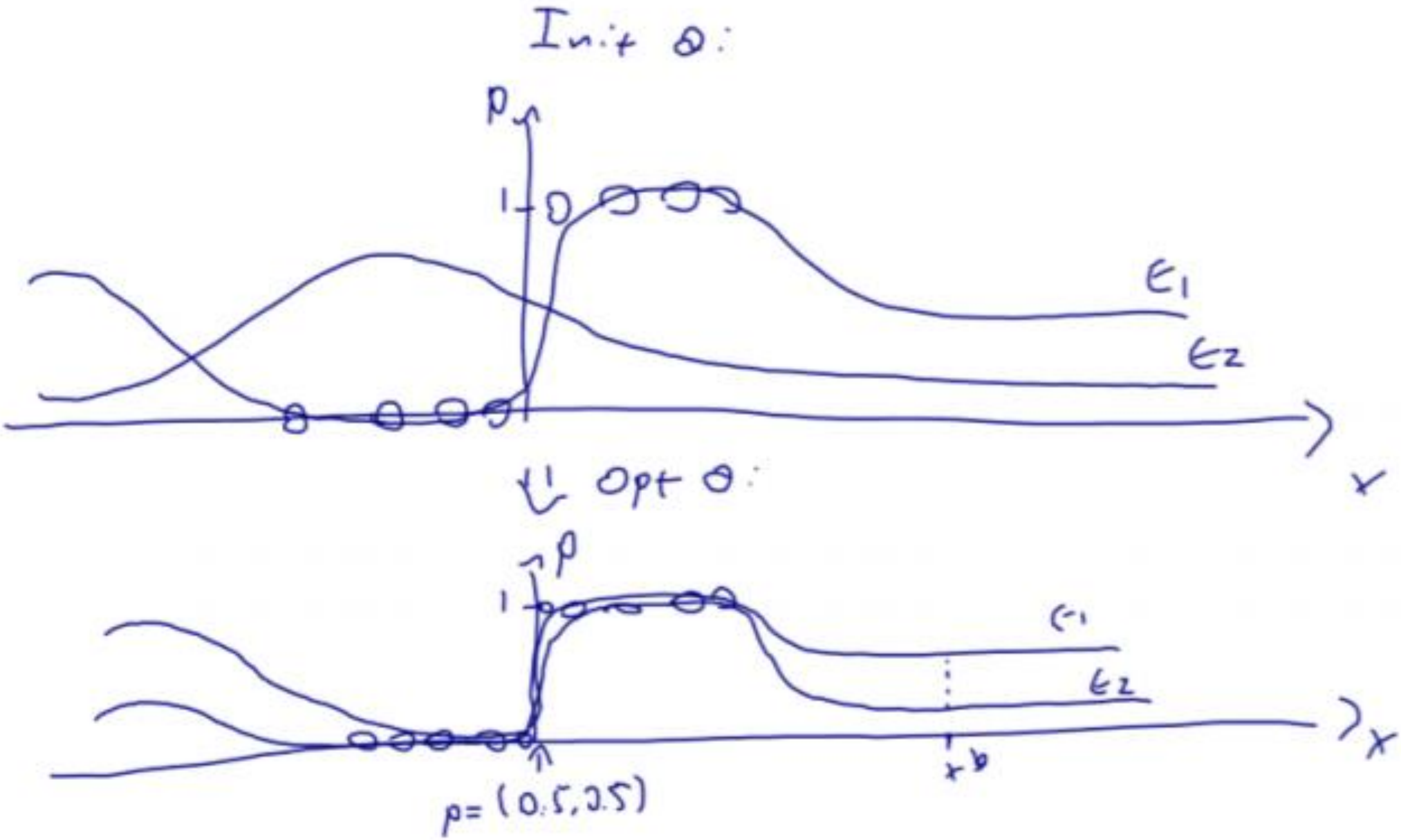
Remember prev ELBO which we couldn't eval

$$L(\theta) = \sum_{x_n, y_n = c} \int \left[f^W(x_n)_c - \log\left(\sum_{c'} e^{f^W(x_n)_{c'}} \right) \right] N(W; \mu_{\text{VI}}, \Sigma_{\text{VI}}) \mathrm{d}W$$
$$- \text{KL}(q, p)$$

W vectorised w dim CK by 1, so is μ_{VI}, and assume Σ_{VI} is diagonal w dim CK by CK

- using MC integration
  - sample $\hat{\epsilon} \sim \mathcal{N}(0, I_{CK})$
  - write $\text{vec}\hat{W}(\theta, \hat{\epsilon}) = \mu_{\text{VI}} + \Sigma_{\text{VI}}^{1/2}\hat{\epsilon}$
  - reshape $\text{vec}\hat{W}$ to K by C: $\hat{W}(\theta, \hat{\epsilon})$
  - write $f^{\theta, \hat{\epsilon}}(x) = f^{\hat{W}(\theta, \hat{\epsilon})}(x)$
  - giving
  $$\hat{L}(\theta, \hat{\epsilon}) = \sum_{x_n, y_n = c} f^{\theta, \hat{\epsilon}}(x_n)_c - \log\left(\sum_{c'} e^{f^{\theta, \hat{\epsilon}}(x_n)_{c'}} \right) - \text{KL}(q, p)$$
- with $E_{p(\epsilon)}[\hat{L}(\theta, \epsilon)] = L(\theta)$, $E_{p(\epsilon)}[\hat{G}(\theta, \epsilon)] = G(\theta)$

Back to approx inference in classification NN



Uncertainty in classification NN

Epistemic uncertainty in classification (vs regression)

- finally have tools to get epistemic uncertainty for classification
- but quantifying uncertainty in classification is not as straightforward as in regression...
- use various *measures of uncertainty* from the field of Information Theory, which have different properties
- each capturing different uncertainty desiderata

Useful tools

- Entropy $H_{p(X)}[X] = -\sum_{\text{outcomes } x} p(X = x) \log p(X = x)$
  - high when p is **uniform**, 0 when one outcome is **certain**

- Mutual information of rvs X and Y

$$MI(X, Y) = H_{p(X)}[X] - E_{p(Y)}[H_{p(X|Y)}[X]]$$

  - "how much information on X we would get if we had observed Y"

Uncertainty in classification NN

A quick overview:

- *Predictive Entropy*
  - entropy of predictive distribution $p(y = y^*|x^*, D)$

$$H_{p(y^*|x^*,\mathcal{D})}[y^*] = -\sum_{y^*=c} p(y^* = c|x^*, \mathcal{D}) \log p(y^* = c|x^*, \mathcal{D})$$

- *Mutual Information* (MI)
  - between model params rv W and model output rv y^* on input x^*

$$MI(y^*, W|\mathcal{D}, x^*) = H_{p(y^*|x^*,\mathcal{D})}[y^*] - E_{p(W|\mathcal{D})}[H_{p(y^*|x^*,W)}[y^*]]$$

  - satisfies

$$0 \leq MI[x^*] \leq H[x^*]$$

Uncertainty in classification NN

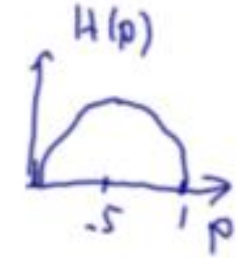
Predictive entropy

$$H_{p(y^*|x^*,\mathcal{D})}[y^*] = -\sum_{y^*=c} p(y^* = c|x^*, \mathcal{D}) \log p(y^* = c|x^*, \mathcal{D})$$

- MC approximation

$$p(y^* = c|x^*, \mathcal{D}) \approx \frac{1}{T}\sum_t p^{\hat{W}_t}(x^*)_c$$

with $\hat{W}_t \sim q_\theta(W)$ and $p^{\hat{W}_t}(x^*) = \text{softmax}(f^{\hat{W}_t}(x^*))$



- high when predictive is near uniform
- so, high **either** when we have inherent ambiguity
  - eg when a point x has training labels both 0 and 1
  - for ambiguous input x loss is $\log p(x) + \log(1 - p(x))$
  - cross entropy loss minimiser (=ELBO miximiser) is to predict $p = .5$
  - all func draws will go through (.5,.5) (ie high entropy)
- **or** when far away from data: eg half draws=1 and half draws=0

UNIVERSITY OF OXFORD

Mutual information

$$MI(y^*, W|\mathcal{D}, x^*) = H_{p(y^*|x^*,\mathcal{D})}[y^*] - E_{p(W|\mathcal{D})}[H_{p(y^*|x^*,W)}[y^*]]$$

- MI MC approx (second term)

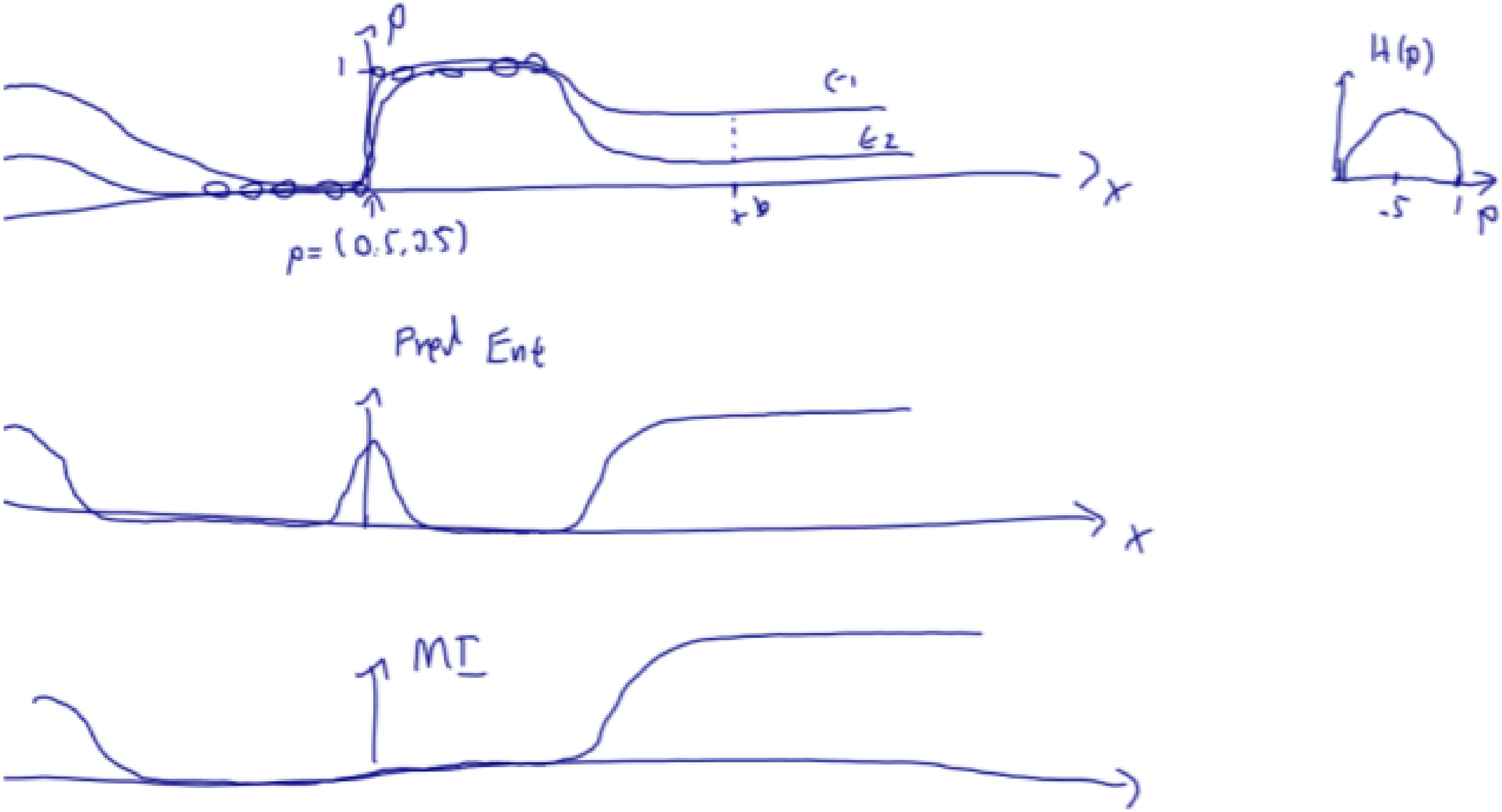
$$\int p(W|\mathcal{D}) \sum_{y^*=c} p(y^*=c|x^*, W) \log p(y^*=c|x^*, W) \mathrm{d}W$$

$$\approx \frac{1}{T} \sum_{t, y^*=c} p^{\hat{W}_t}(x^*)_c \log p^{\hat{W}_t}(x^*)_c$$

with $\hat{W}_t \sim q_\theta(W)$

- high **only** when we are far away from data
  - has "second term = first term" if all func draws same for input x
  - "second term = 0" when func preds are confident and all over the place
- ie, capturing **only** epistemic uncertainty (vs pred ent capturing epistemic *and aleatoric* uncertainty)

Uncertainty in classification NN



Predictive: $p(y^* = c|x^*, \mathcal{D}) \approx \frac{1}{T}\sum_t p^{\hat{W}_t}(x^*)_c$

MI: $MI(y^*, W|\mathcal{D}, x^*) = H_{p(y^*|x^*,\mathcal{D})}[y^*] - E_{p(W|\mathcal{D})}[H_{p(y^*|x^*,W)}[y^*]]$

Stochastic Approximate Inference in Deep NN

Summary so far

- Model for regression (D outputs) / classification
- ELBO $L(\theta) = \int q_\theta(W) \log p(Y|X,W) \mathrm{d}W - \mathrm{KL}(q, \text{prior})$
- log likelihood eg
  $\log p(Y|X,W) = -\frac{1}{2\sigma^2} \sum ||y_n - f^W(x_n)||_2^2 - \frac{N}{2} \log 2\pi\sigma^2$
- approx post eg $q_\theta(w_{kd}) = N(w_{kd}; m_{kd}, \sigma_{kd}^2)$
  - $\mathrm{KL}(q, \text{prior}) = \sum_{kd} 1/2(s^{-2}\sigma_{kd}^2 + s^{-2}{m_{kd}}^2 - 1 + \log(s^2/\sigma_{kd}^2))$
- MC integration:
  - sample $\hat{\epsilon}_{kd} \sim \mathcal{N}(0,1)$ and write $\hat{w}_{kd} = m_{kd} + \sigma_{kd}\hat{\epsilon}_{kd}$, $\hat{\epsilon} = \{\hat{\epsilon}_{kd}\}$
  - giving a K by D stochastic weight matrix: $\hat{W}(\theta, \hat{\epsilon})$
  - write $f^{\theta,\hat{\epsilon}}(x) = \hat{W}(\theta,\hat{\epsilon})^T \phi(x)$
  - giving

$$\hat{L}(\theta, \{\hat{\epsilon}_n\}) = -\frac{1}{2\sigma^2} \sum_{x_n, y_n} ||y_n - f^{\theta,\hat{\epsilon}_n}(x_n)||_2^2 - \frac{N}{2} \log 2\pi\sigma^2 - \frac{1}{2s^2}||M||_2^2..$$

Stochastic VI in deep models

- until now we only did inference over W (last layer weights)
- because doing inference on preceding layers was too challenging (intractable / non-conjugate)
- but with our new techniques we can **easily** extend to W, b of all layers in model (denoted ω)
- these models (where all layers have dists over) are known as Bayesian neural networks (BNNs)
  - X of dim N by Q (and Y of dim N by D)
  - W^1 of dim Q by K, b^1 dim K
  - W^2 of dim K by D, b^2 dim D
  - ϕ elem-wise non-linearity
  - $\omega = \{W^1, W^2, b^1, b^2\}$
  - $f^\omega(x) = \phi(x^T W^1 + b^1) W^2 + b^2$
    note: could be a deep net with thousands of layers
- Long history (Hopfield [1987] $\rightarrow$ LeCun [1991] $\rightarrow$ MacKay [1992] $\rightarrow$ Hinton [1993] $\rightarrow$ Neal [1995] $\rightarrow$ Barber and Bishop [1998])

UNIVERSITY OF OXFORD

BNNs

- model
  - as before, but swap W^1, W^2, f^ω instead of W and f^W
- approx inference
  - log likelihood – same
  - approx post – Gaussians w means $\{m^1_{qk}, m^2_{kd}\}$ and stds $\{\sigma^1_{qk}, \sigma^2_{kd}\}$
    KL to prior $\text{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - ELBO

$$\hat{W}^1_{qk} = m^1_{qk} + \sigma^1_{qk}\hat{\epsilon}^1_{qk}$$
$$\hat{W}^2_{kd} = m^2_{kd} + \sigma^2_{kd}\hat{\epsilon}^2_{kd}$$

  with $\hat{\epsilon}^1_{qk}, \hat{\epsilon}^2_{kd} \sim \mathcal{N}(0,1)$ and $\hat{\epsilon} = \{\hat{\epsilon}^1_{qk}, \hat{\epsilon}^2_{kd}\}$
  - swap $f^{\theta,\hat{\epsilon}}(x) = \hat{W}(\theta, \hat{\epsilon})^T \phi(x)$ with

$$f^{\theta,\hat{\epsilon}}(x) = \phi(x^T \hat{W}^1(\theta, \hat{\epsilon}) + b^1)\hat{W}^2(\theta, \hat{\epsilon}) + b^2$$

  and plug into $\hat{L}(\theta, \{\hat{\epsilon}_n\})$.
- Proposed as *MDL* from compression literature [Graves, 2011], in Bayesian modelling known as *mean-field variational inference (MFVI)*; Also referred to as *Bayes by Backprop* [Blundell, 2015].

Inference in Very Large Deep Models

Issue with above...

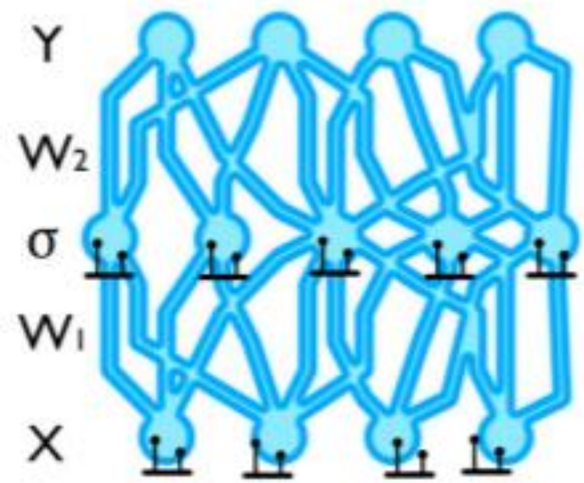
- when we use large models we usually use 10s-100s of millions of params – models as big as can fit on GPU
- when using Gaussian approx we need at least two params for each NN weight
- doubling num of params... so having to reduce model size by 2!
- can we scale the ideas above to very large models?

(a) Input Image (b) Semantic Segmentation (c) Epistemic Uncertainty

UNIVERSITY OF OXFORD

Stochastic regularisation

- lots of techniques in deep learning inject noise into large models to help with regularisation
- eg dropout (but lots of others which mostly work the same)
  - at training time, randomly set network units to zero with prob p (Bern)
    - call this "stochastic forward pass"
  - at test time multiply each unit by $1/(1-p)$ and do not drop
    - call this "deterministic forward pass"
  - noise is added to units (feature space)
  - implemented in *every* deep learning framework (from TF/PyTorch to TensorRT for embedded devices)



Y : $Q \times 1$

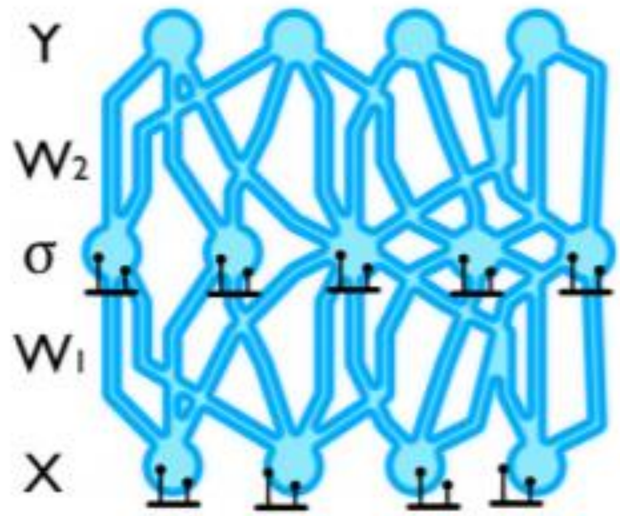
$\hat{x} := x^T \hat{\epsilon}^1$, $[\hat{\epsilon}_1]_{qq} \sim \mathrm{Bern}(p^1)$, a Q by Q matrix (zero off diag)

$h := \phi(\hat{x} M^1 + b^1)$

$\hat{h} := h \hat{\epsilon}^2$, $[\hat{\epsilon}_2]_{kk} \sim \mathrm{Bern}(p^2)$

$\hat{y} := \hat{h} M^2 + b_2$

$\hat{J} = \frac{1}{N} \sum_n \| y_n - \hat{y}_n \|_2^2 + \lambda_1 \|M^1\|_2^2 + \lambda_2 \|M^2\|_2^2$



Feature space noise to weight space

- Can transform noise to param (weight) space

$$\hat{y} = \left(\phi\big[(x\hat{\epsilon}^1)M^1 + b^1\big]\hat{\epsilon}^2\right)M^2 + b^2$$
$$= \phi\big[x(\hat{\epsilon}^1 M^1) + b^1\big](\hat{\epsilon}^2 M^2) + b^2$$

writing $\hat{W}^1 := \hat{\epsilon}^1 M^1$ and $\hat{W}^2 := \hat{\epsilon}^2 M^2$ gives

$$\hat{y} = \phi(x\hat{W}^1 + b^1)\hat{W}^2 + b^2 = f^{\hat{\omega}}(x)$$

with $\hat{\omega} = \{\hat{W}^1, \hat{W}^2\}$

- so at training time dropout samples weights matrices... looks v familiar!
- let's see if we can make this connection more formal
- develop approx inf in BNNs with a new approx dist...

Feature space noise to weight space

- model
  - prior - same
  - lik - same
- approx inference
  - approx dist $q_\theta(W^1) = M^1\epsilon$ with $\epsilon_{qq} = \text{Bernoulli}(p^1)$ and zero otherwise, and $\theta = \{M^1, p^1, M^2, p^2\}$

$$\text{KL}(q,p) \approx \frac{1-p^1}{2s^2}||M^1||_2^2 - QH(p^1) + \frac{1-p^2}{2s^2}||M^2||_2^2 - KH(p^2) + \text{const}$$

  - ELBO

$$\hat{L}(\theta, \{\hat{\epsilon}_n\}) = -\frac{1}{2\sigma^2}\sum_{x_n, y_n}||y_n - f^{\theta,\hat{\epsilon}_n}(x_n)||_2^2 - \frac{N}{2}\log 2\pi\sigma^2 - \frac{1-p^1}{2s^2}||M^1||_2^2$$

with $f^{\theta,\hat{\epsilon}_n}(x_n)$ a dropout stochastic forward pass

  - can rewrite as min obj (multiply by $-2\sigma^2/N$)

$$J = \frac{1}{N}\sum_n ||y_n - \hat{y}_n||_2^2 + \lambda^1||M^1||_2^2 + \lambda^2||M^2||_2^2 + \text{const}$$

with $\hat{y}_n = f^{\theta,\hat{\epsilon}_n}(x_n)$ and defining $\lambda^1 = \sigma^2\frac{1-p^1}{s^2N}$..

Feature space noise to weight space

- This is the standard dropout objective
- ie any standard NN in which you use dropout, you can view as a BNN
- note: need to tune p as a variational param
  - can't diff wrt p (used in Bern in obj; can't use reparam trick..)
  - but when you do grid search over p on a validation set, use $\hat{L}(\theta, \{\hat{\epsilon}_n\})$ to select p which max ELBO or validation log predictive
  - Can also use continuous relaxation for dropout (see Concrete Dropout, 2017)
- Example:

Epistemic uncertainty in regression BNNs

UNIVERSITY OF OXFORD

Using MC estimators can estimate epistemic uncertainty in BNNs almost trivially...

- predictive mean
  - $$E_{p(y^*|x^*,\mathcal{D})}[y^*] \approx \frac{1}{T}\sum_t f^{\hat{\omega}_t}(x)$$

    with $\hat{\omega}_t \sim q_\theta(\omega)$.
    ie, average multiple stochastic forward passes
- predictive variance
  - again, collect some stochastic forward passes...

$$\begin{aligned}\mathrm{Var}_{p(y^*|x^*,\mathcal{D})}[y^*] &= E_{p(y^*|x^*,\mathcal{D})}[(y^*)^2] - E_{p(y^*|x^*,\mathcal{D})}[y^*]^2 \\ &\approx \sigma^2 + \frac{1}{T}\sum_t f^{\hat{\omega}_t}(x)^2 - \left(\frac{1}{T}\sum_t f^{\hat{\omega}_t}(x)\right)^2\end{aligned}$$

How to visualise BNNs?

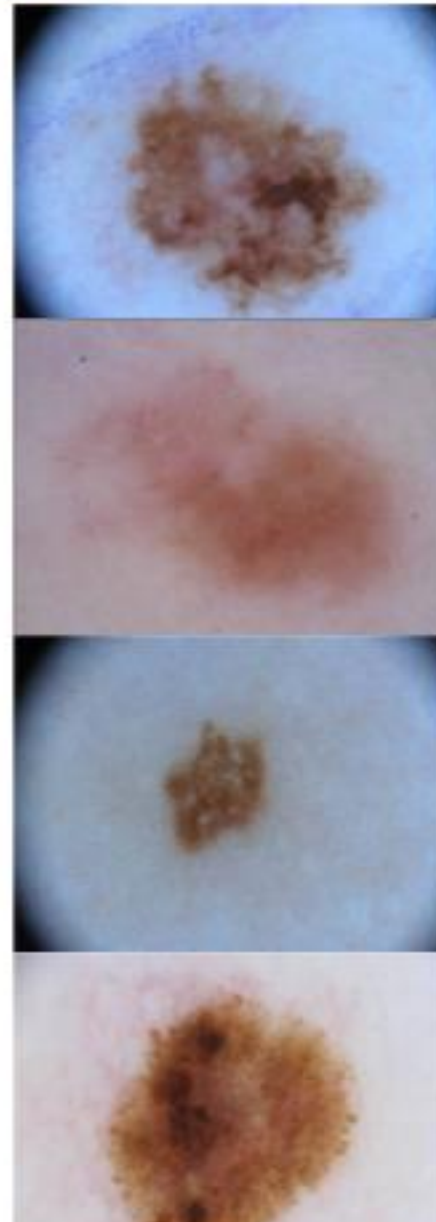
A useful tool for debugging

- sample from weights $\omega \sim p(\omega|\mathcal{D})$ = function sample $f^{\omega}(\cdot)$
- evaluate over interval $[-10, 10]$
- eg:
    - sample ω and def $f^{\omega}(\cdot)$
    - for each x_i in $\{-10, -9.95, -9.9, ..., 9.9, 9.95, 10\}$
        - evaluate $y_i = f^{\omega}(x_i)$ and plot (x_i, y_i)
- note: if using dropout inference, use same dropout mask for all inputs x
- Visualisation: bdl101.ml/vis

Real-world Applications of Model Uncertainty

Small data big models

- we use machine learning to aid experts working in laborious fields
- automate small parts of the expert's work
    - eg melanoma (cancer) diagnosis based on lesion images
- but deep learning often requires large amounts of labelled data
    - increases with the complexity of problem
    - complexity of the input data
    - eg image inputs require large models
    - hundreds of gigabytes in ImageNet
- sometimes can't afford to label huge data...
    - eg automating lesion image analysis
    - would require expert to spend expensive time annotating large number of lesion images (for every cancer type of interest)
- instead, could use *active learning*



Principles of active learning

- active learning
  - *agent* chooses which unlabelled data is most informative
  - asks external "oracle" (eg human annotator) for a label only for that
  - acquisition function: ranks points based on their potential informativeness
  - eg, epistemic uncertainty