Bayesian Deep Learning (MLSS 2019)

Yarin Gal

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

Previous Lecture

Previously..

- Bayesian probabilistic modelling of functions
- Analytical inference of W (mean)

Contents

UNIVERSITY OF OXFORD

Today:

- Uncertainty over functions (and decomposing uncertainty)
- Scaling ideas up (approximate inference)
- Scaling up even more (stochastic approximate inference)
- Uncertainty in shallow classification models
- Stochastic approximate inference in deep NN
- Inference in very large deep models
- Real-world applications of model uncertainty

UNIVERSITY OF OXFORD

All resources (including these slides): bdl101.ml

SLIDES

Slide decks from the talks.

SLIDE DECK 1

SLIDE DECK 2

DEMO

Demoes mentioned in the slides

UNCERTAINTY PLAYGROUND

UNCERTAINTY VISUALISATION

RECAP

A quick recap of useful stuff.

GAUSSIANS RECAP

NOTATION

Notation used in the slides:.

MORE STUFF

OATML

Uncertainty over Functions

Reminder

UNIVERSITY OF OXFORD

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

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

Reminder

UNIVERSITY OF OXFORD

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

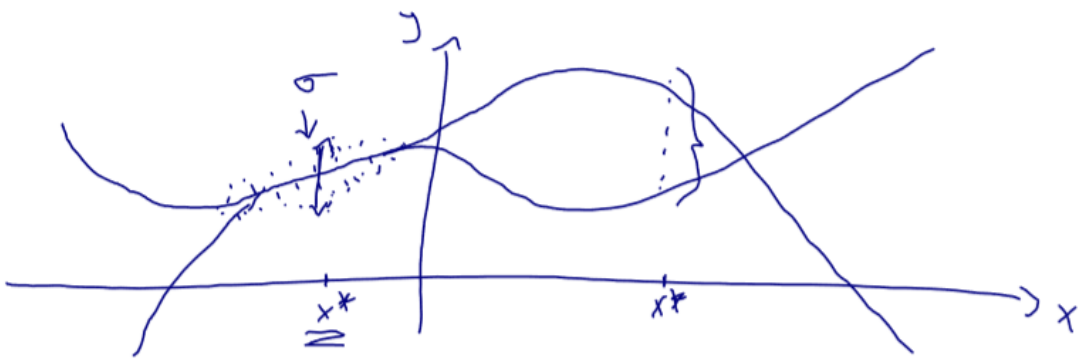
$$p(y^*|x^*, X, Y) = \mathcal{N}(y^*; \mu'^T\phi(x^*), \sigma^2 + \phi(x^*)^T\Sigma'\phi(x^*))$$

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

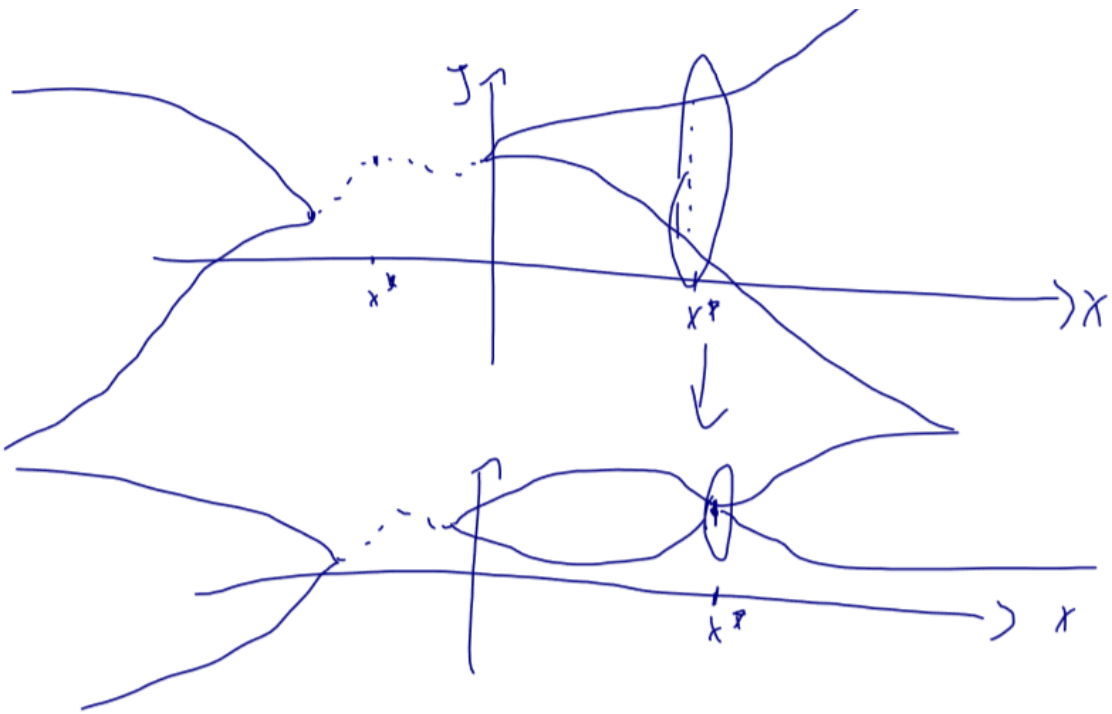


Decomposing uncertainty

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

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 Inference

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

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

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

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

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

- which would you choose?
  - 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

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

$$q_1(W) = \mathcal{N}(0, 2), \qquad q_2(W) = \mathcal{N}(0, 10)$$

- which would you choose? 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

- 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

- 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

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

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)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 properties (eg w discrete distributions)

UNIVERSITY OF OXFORD

- 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 $\mathbf{x} = [x_1, .., x_K]$

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

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 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 $\mathbf{x} = [x_1, .., x_K]$

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

  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 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 $x_1, .., x_K$:

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

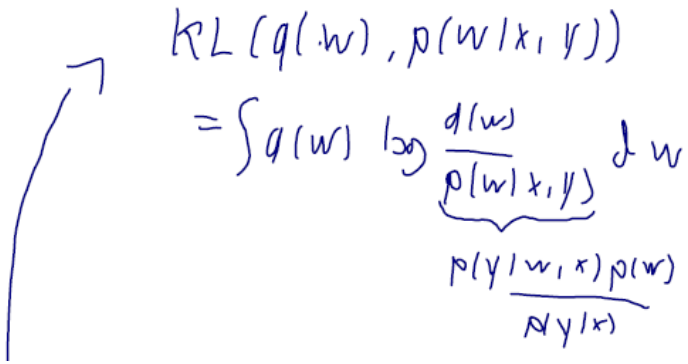
KL for approx inference

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

KL for approx inference



KL for approx inference

$$KL(q(w), p(w|x,y))$$

$$= \int q(w) \log \frac{q(w)}{p(w|x,y)} dw$$

$$p(w|x,y) = \frac{p(y|w,x)p(w)}{p(y|x)}$$

$$= \int q(w) \log \frac{q(w)p(y|x)}{p(y|w,x)p(w)} dw$$

KL for approx inference

$$KL(q(w), p(w|x,y))$$

$$= \int q(w) \log \frac{q(w)}{p(w|x,y)} dw$$

$$p(w|x,y) = \frac{p(y|w,x)p(w)}{p(y|x)}$$

$$= \int q(w) \log \frac{q(w)p(y|x)}{p(y|w,x)p(w)} dw$$

$$= \int q(w) \underbrace{\log \frac{1}{p(y|w,x)}}_{=-\log p(\cdots)} dw + \underbrace{\int q(w) \log \frac{q(w)}{p(w)} dw}_{=KL(q(w),p(w))} - \int q(w) \log p(y|x) dw$$

KL for approx inference

$$KL(q(w), p(w|x,y))$$

$$= \int q(w) \log \frac{q(w)}{p(w|x,y)} dw$$

$$p(w|x,y) = \frac{p(y|w,x)p(w)}{p(y|x)}$$

$$= \int q(w) \log \frac{q(w)p(y|x)}{p(y|w,x)p(w)} dw$$

$$= \int q(w) \underbrace{\log \frac{1}{p(y|w,x)}}_{= -\log p(\cdots)} dw + \underbrace{\int q(w) \log \frac{q(w)}{p(w)} dw}_{= KL(q(w), p(w))} - \int q(w) \log p(y|x) dw$$

$$= -\int q(w) \log p(y|w,x) dw + KL(q(w), p(w)) - \log p(y|x)$$

KL for approx inference

$$KL(q(w), p(w|x,y))$$

$$= \int q(w) \log \frac{q(w)}{p(w|x,y)} dw$$

$$p(w|x,y) = \frac{p(y|w,x)p(w)}{p(y|x)}$$

$$= \int q(w) \log \frac{q(w)p(y|x)}{p(y|w,x)p(w)} dw$$

$$= \int q(w) \log \frac{1}{p(y|w,x)} dw + \int q(w) \log \frac{q(w)}{p(w)} dw - \int q(w) \log p(y|x) dw$$

$\log p(\dots)$ = $-\log p(\dots)$

$\int q(w) \log \frac{q(w)}{p(w)} dw = KL(q(w), p(w))$

$$= -\int q(w) \log p(y|w,x) dw + KL(q(w), p(w)) + \log p(y|x)$$

$$\Rightarrow \underbrace{\int q(w) \log p(y|x,w) dw}_{\text{Exp log lik}} - \underbrace{KL(q(w), p(w))}_{\text{KL to prior}} = \underbrace{\log p(y|x)}_{\text{log evidence}} - KL(q(w), p(w|x,y))$$

ELBO

KL for approx inference

$$KL(q(w), p(w|x,y))$$

$$= \int q(w) \log \frac{q(w)}{p(w|x,y)} dw$$

$$p(w|x,y) = \frac{p(y|w,x)p(w)}{p(y|x)}$$

$$= \int q(w) \log \frac{q(w)p(y|x)}{p(y|w,x)p(w)} dw$$

$$= \int q(w) \log \frac{1}{p(y|w,x)} dw + \int q(w) \log \frac{q(w)}{p(w)} dw + \int q(w) \log p(y|x) dw$$

$$= -\log p(\cdots)$$

$$= KL(q(w), p(w))$$

$$= -\int q(w) \log p(y|w,x) dw + KL(q(w), p(w)) + \log p(y|x)$$

LHS $\leq$ RHS

$$\Rightarrow \underbrace{\int q(w) \log p(y|x,w) dw - KL(q(w), p(w))}_{\text{ELBO}} \leq \underbrace{\log p(y|x)}_{\text{log evidence}} - KL(q(w), p(w|x,y))$$

exp log lik — KL to prior

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

- 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

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

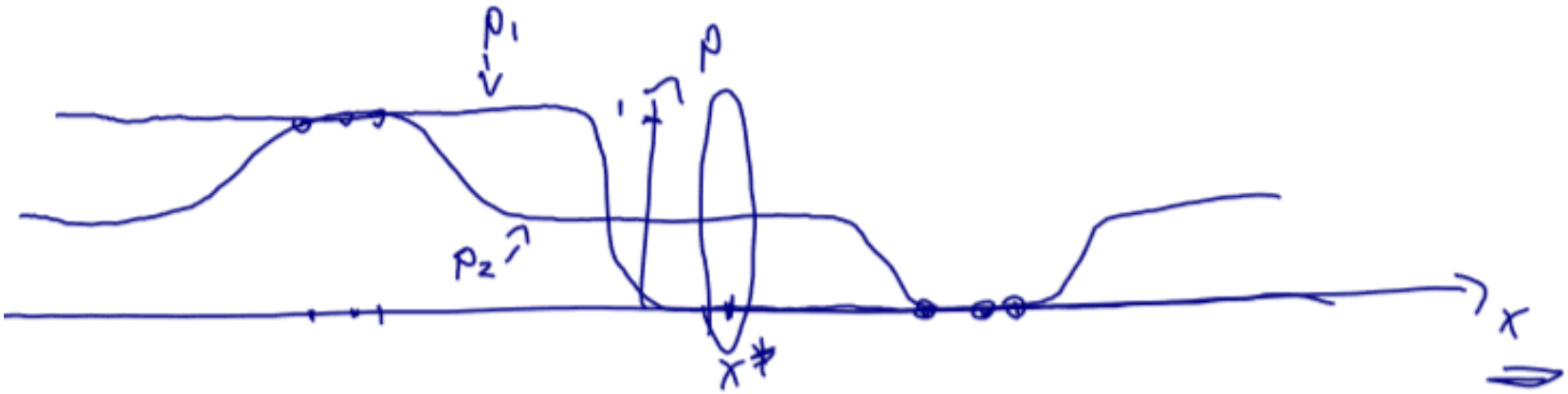
- which terms can we compute?
  - for Gaussian prior and q, can compute KL to prior
  - for Gaussian lik can compute expected log lik as well (analytic – try this at home using tools from earlier!)
  - but in more complicated likelihoods (like in **classification**) can't eval above...
  - for this we'll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

Stochastic Approximate Inference

Classification NN

Let's try to do a classification task

- want to get notion of epistemic uncertainty in classification

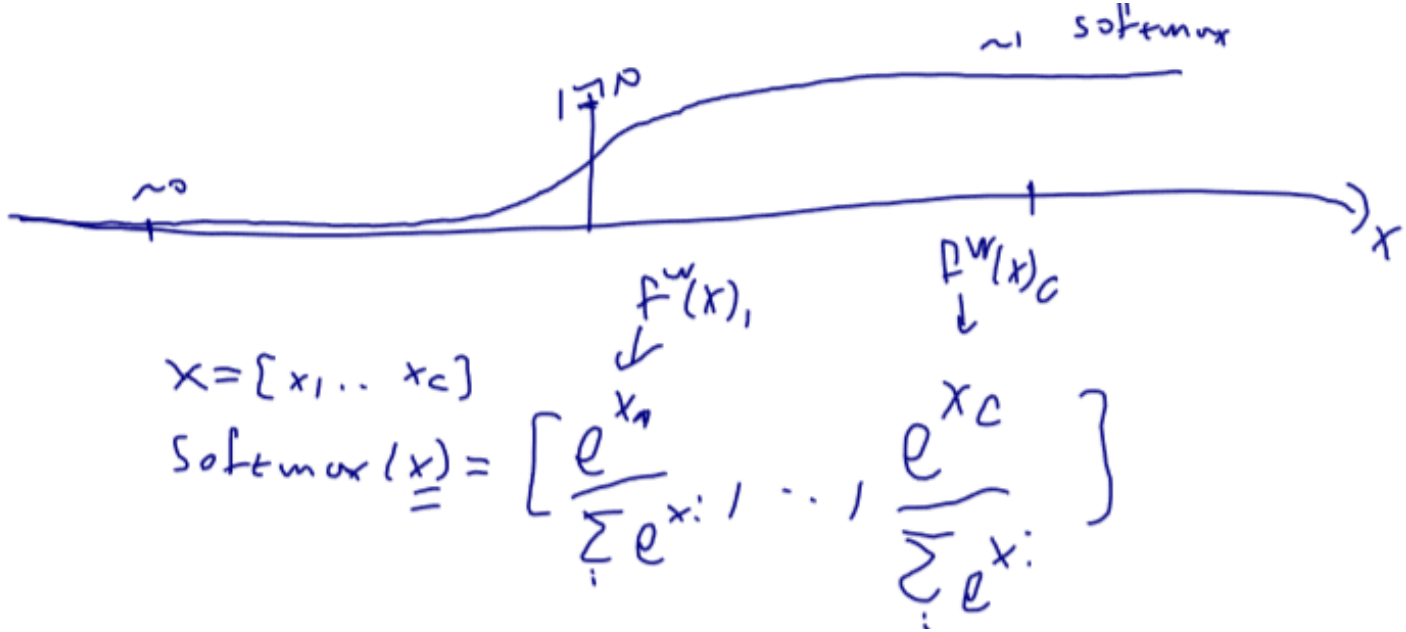


- generative story
  - Nature chose function $p(x) : \mathbb{R}^Q \to [0,1]^C$
  - $p(x)$ a prob vector as a function of x
  - eg p softmax func
  - for $n = 1..N$ generate label $y_n \sim \text{Categorical}(p(x_n))$
- encode y_n as a one hot vector $\mathbf{y}_n$ (eg $[0,0,1,0]$ with $C = 4$ classes and $y_n = 2$)

Classification NN

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 $y_n \sim \text{Categorical}(p(x_n))$

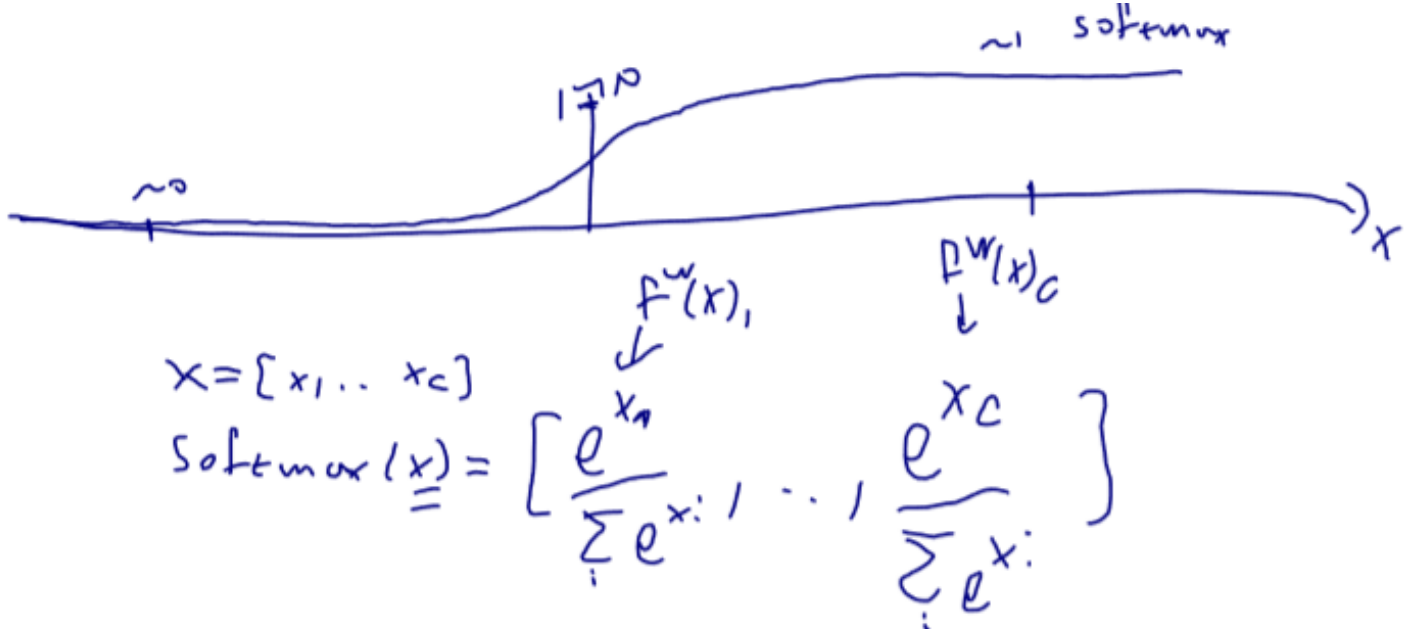


- encode y_n as a one hot vector $\mathbf{y}_n$ (eg $[0,0,1,0]$ with $C = 4$ classes and $y_n = 2$)

Classification NN

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 $y_n \sim \text{Categorical}(p(x_n))$



- 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

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

$p^W(x) =$

$\mathrm{Softmax}(f^w(x))$

Classification NN

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

$p^W(x) =$

$\mathrm{Softmax}(f^w(x))$

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:

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

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

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

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

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

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

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

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

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

Approx inference in classification NN

UNIVERSITY OF OXFORD

- 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!)

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

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

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

($\hat{L}$ 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

- $L(\mu, \sigma) = \int (W + W^2) N(W; \mu, \sigma^2) \mathrm{d}W$
  - 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

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

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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 $\hat{G}(\hat{W}; \sigma) = f'(\hat{W})(\hat{W} - \mu)/\sigma$.

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

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

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 $\mathrm{vec}\hat{W}(\theta, \hat{\epsilon}) = \mu_{\text{VI}} + \Sigma_{\text{VI}}^{1/2} \hat{\epsilon}$
  - reshape $\mathrm{vec}\hat{W}$ to K by C: $\hat{W}(\theta, \hat{\epsilon})$
  - write $f^{\theta, \hat{\epsilon}}(x) = f^{\hat{W}(\theta, \hat{\epsilon})}(x)$
  - giving

$$\hat{L}(\theta, \hat{\epsilon}) = \sum_{x_n, y_n = c} f^{\theta, \hat{\epsilon}}(x_n)_c - \log \left(\sum_{c'} e^{f^{\theta, \hat{\epsilon}}(x_n)_{c'}} \right) - \mathrm{KL}(q, p)$$

- 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

Remember prev ELBO which we couldn't eval

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

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 $\mathrm{vec}\hat{W}(\theta, \hat{\epsilon}) = \mu_{\text{VI}} + \Sigma_{\text{VI}}^{1/2} \hat{\epsilon}$
  - reshape $\mathrm{vec}\hat{W}$ to K by C: $\hat{W}(\theta, \hat{\epsilon})$
  - write $f^{\theta, \hat{\epsilon}}(x) = f^{\hat{W}(\theta, \hat{\epsilon})}(x)$
  - giving

$$\hat{L}(\theta, \hat{\epsilon}) = \sum_{x_n, y_n = c} f^{\theta, \hat{\epsilon}}(x_n)_c - \log \left(\sum_{c'} e^{f^{\theta, \hat{\epsilon}}(x_n)_{c'}} \right) - \mathrm{KL}(q, p)$$

- 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

Remember prev ELBO which we couldn't eval

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

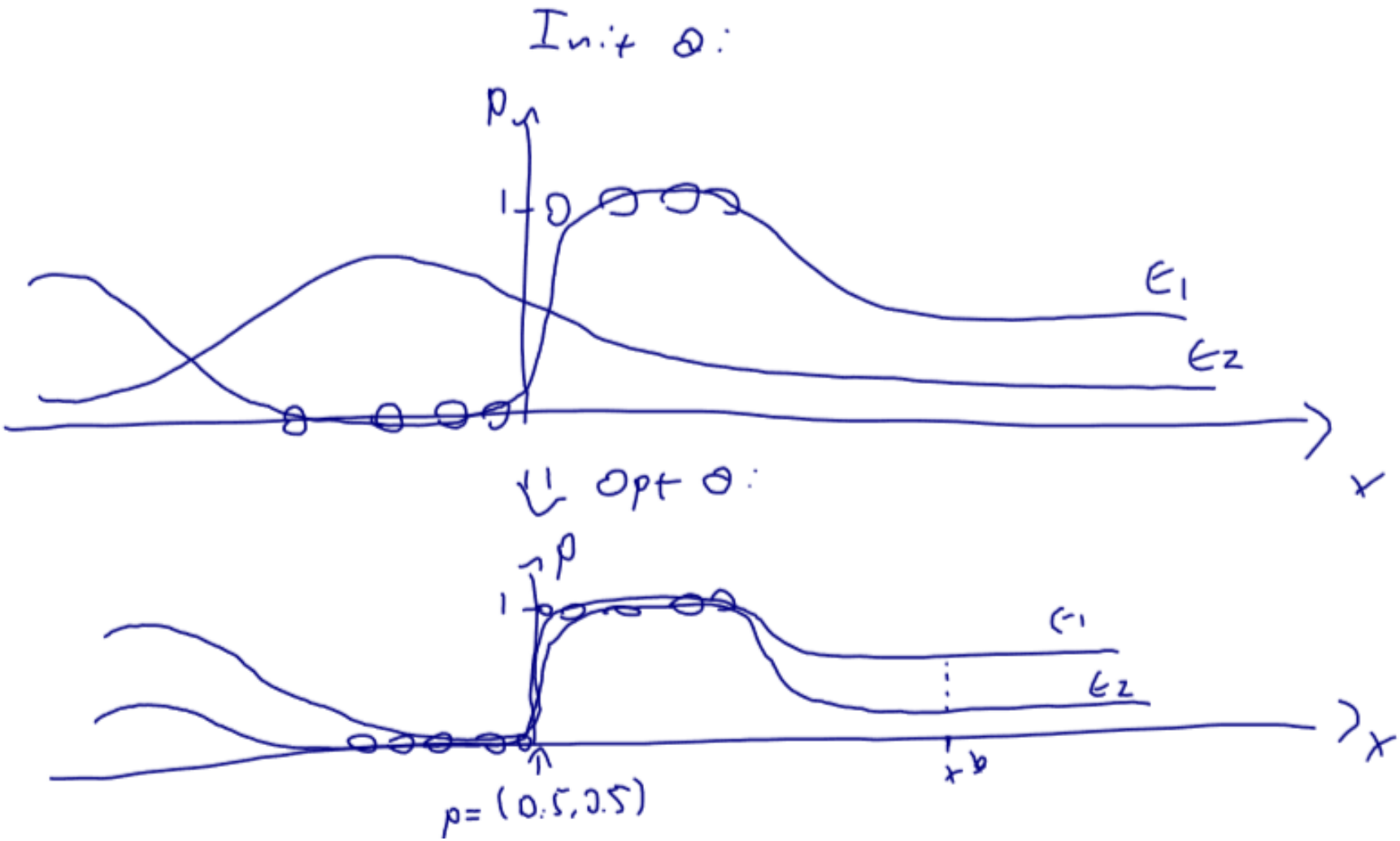
W vectorised w dim CK by 1, so is μ_{VI}, and assume Σ_{VI} is diagonal w dim CK by CK

- using MC integration
  - sample $\hat{\epsilon} \sim \mathcal{N}(0, I_{CK})$
  - write $\mathrm{vec}\hat{W}(\theta, \hat{\epsilon}) = \mu_{\text{VI}} + \Sigma_{\text{VI}}^{1/2} \hat{\epsilon}$
  - reshape $\mathrm{vec}\hat{W}$ to K by C: $\hat{W}(\theta, \hat{\epsilon})$
  - write $f^{\theta, \hat{\epsilon}}(x) = f^{\hat{W}(\theta, \hat{\epsilon})}(x)$
  - giving

$$\hat{L}(\theta, \hat{\epsilon}) = \sum_{x_n, y_n = c} f^{\theta, \hat{\epsilon}}(x_n)_c - \log \left(\sum_{c'} e^{f^{\theta, \hat{\epsilon}}(x_n)_{c'}} \right) - \mathrm{KL}(q, p)$$

- 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

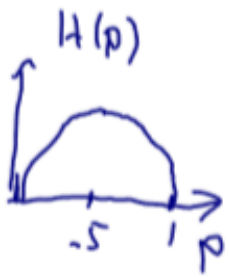
Uncertainty in classification NN

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

Epistemic uncertainty in classification (vs regression)

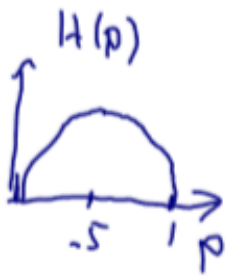
- finally have tools to get epistemic uncertainty for classification
- but quantifying uncertainty in classification is not as straightforward as in regression...
- use various *measures of uncertainty* from the field of Information Theory, which have different properties
- each capturing different uncertainty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



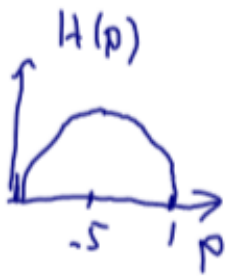
Uncertainty in classification NN

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

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

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

UNIVERSITY OF OXFORD

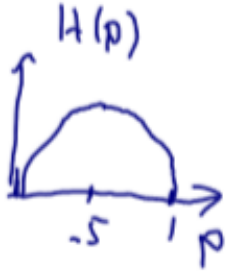
Predictive entropy

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

- 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

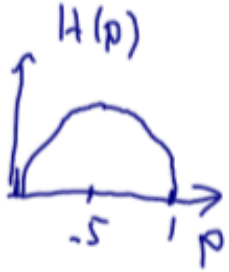
Predictive entropy

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

- 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

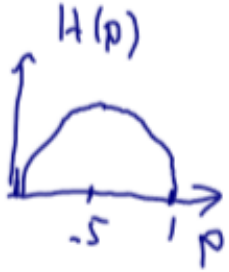
Predictive entropy

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

- 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)

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)

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

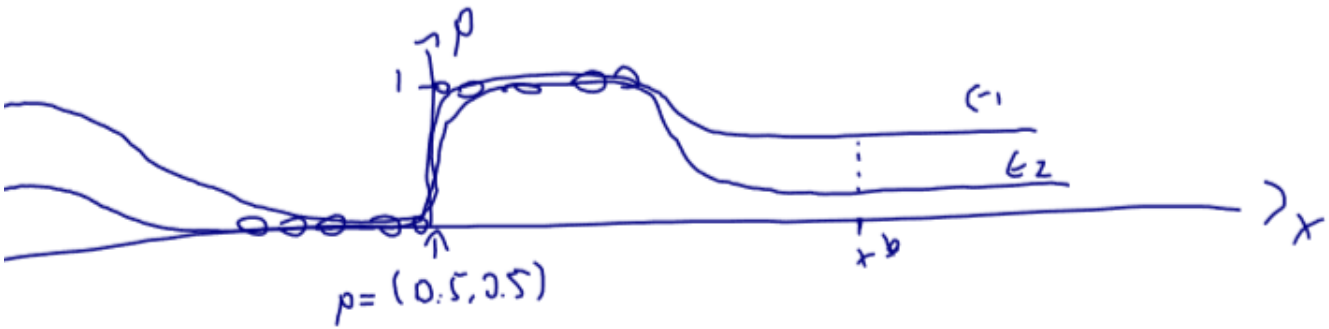
$$\approx \frac{1}{T} \sum_{t,y^*=c} p^{\hat{W}_t}(x^*)_c \log p^{\hat{W}_t}(x^*)_c$$

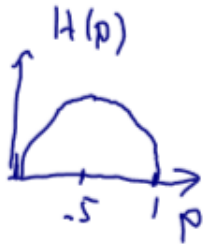
with $\hat{W}_t \sim q_\theta(W)$

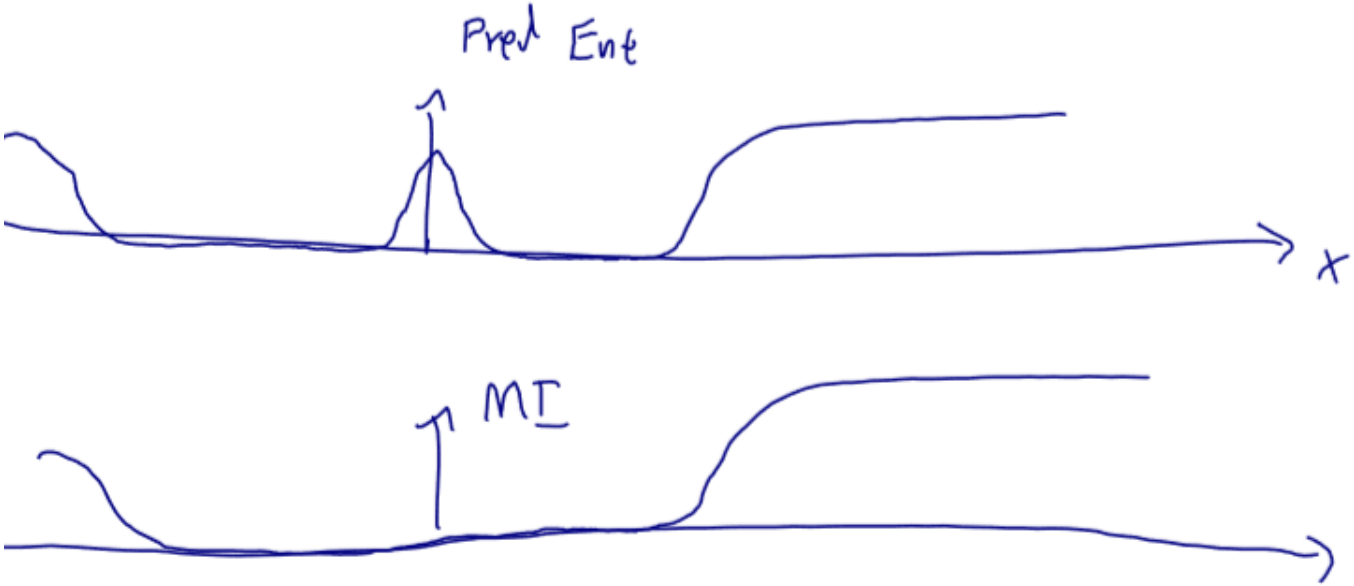
- high **only** when we are far away from data
  - has "second term = first term" if all func draws same for input x
  - "second term = 0" when func preds are confident and all over the place

- 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

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

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

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

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

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

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

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

Summary so far

UNIVERSITY OF OXFORD

- 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])

Stochastic VI in deep models

UNIVERSITY OF OXFORD

- 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])

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 $\mathrm{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - 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].

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 $\mathrm{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - 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].

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 $\mathrm{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - 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].

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 $\mathrm{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - 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].

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 $\mathrm{KL}(q(W^1, W^2), p) = KL(q(W^1), p) + KL(q(W^2), p)$
  - 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?

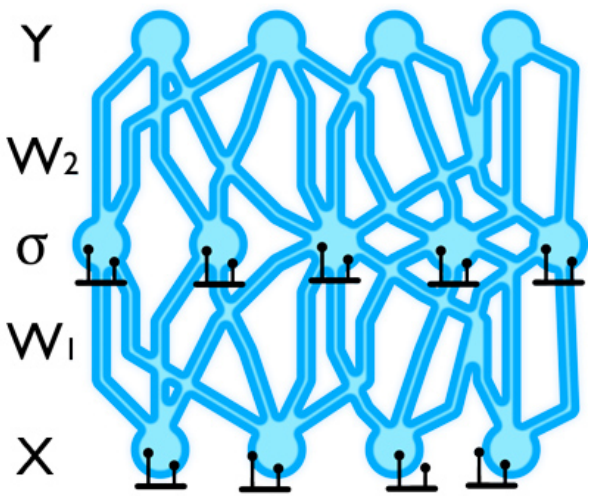
(a) Input Image (b) Semantic Segmentation (c) Epistemic Uncertainty

Inference in Very Large Deep Models

UNIVERSITY OF OXFORD

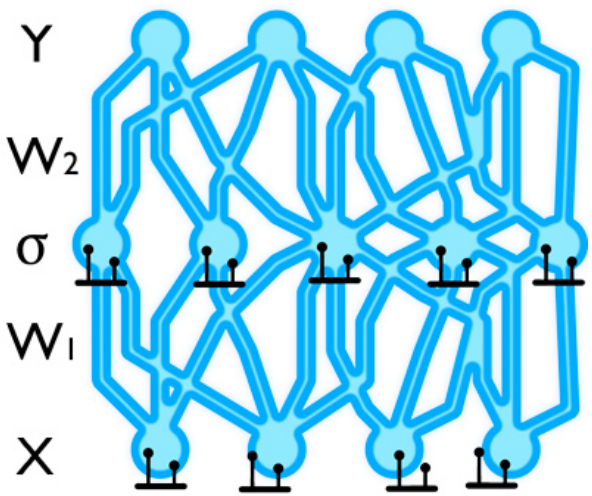
Stochastic regularisation

- 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)



Stochastic regularisation

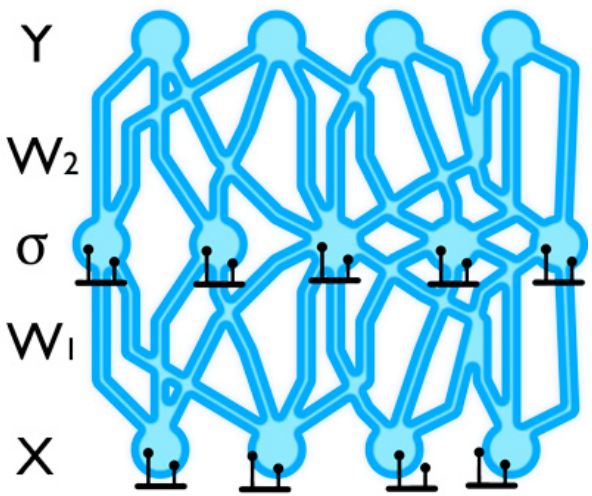
- lots of techniques in deep learning inject noise into large models to help with regularisation
- eg dropout (but lots of others which mostly work the same)
  - at training time, randomly set network units to zero with prob p (Bern)
    - call this "stochastic forward pass"
  - at test time multiply 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)



UNIVERSITY OF OXFORD

Stochastic regularisation

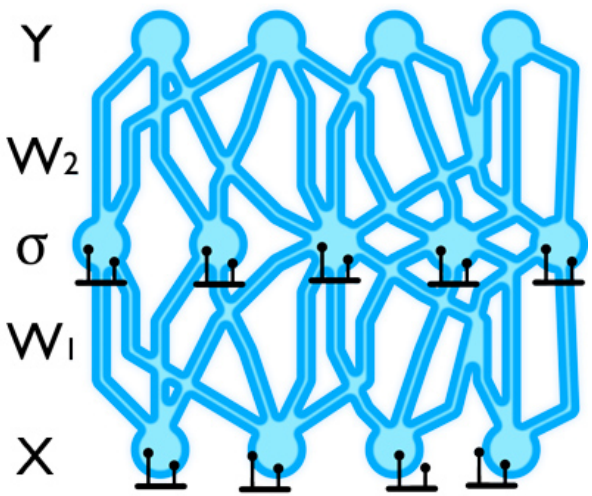
- lots of techniques in deep learning inject noise into large models to help with regularisation
- eg dropout (but lots of others which mostly work the same)
  - at training time, randomly set network units to zero with prob p (Bern)
    - call this "stochastic forward pass"
  - at test time multiply 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)



UNIVERSITY OF OXFORD

Stochastic regularisation

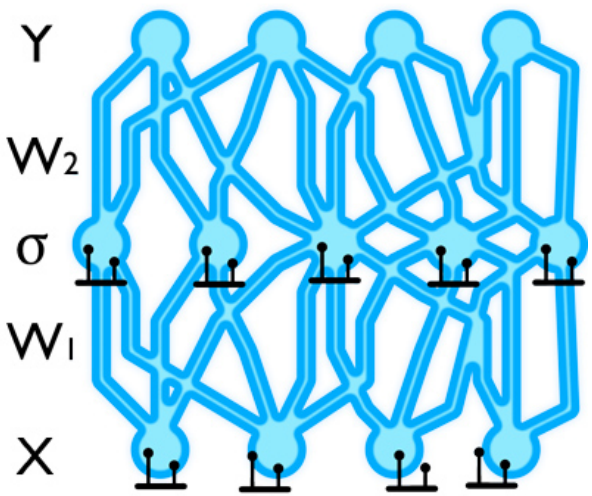
- lots of techniques in deep learning inject noise into large models to help with regularisation
- eg dropout (but lots of others which mostly work the same)
  - at training time, randomly set network units to zero with prob p (Bern)
    - call this "stochastic forward pass"
  - at test time multiply 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)



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)



Stochastic VI in deep models

$Y : Q \times 1$

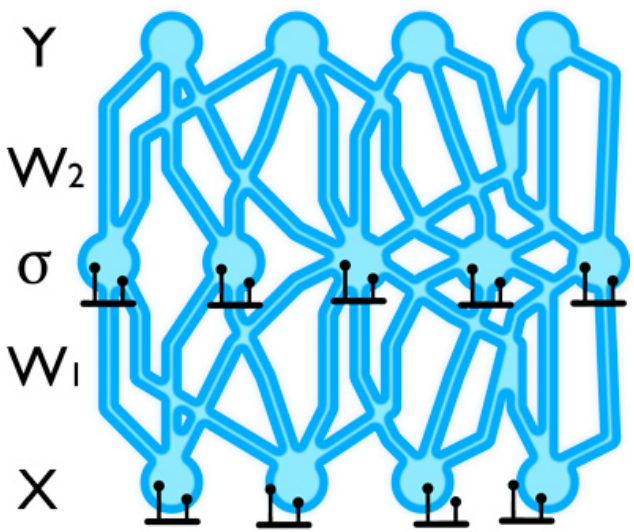
$\hat{x} := x^T \hat{\epsilon}^1$, $[\hat{\epsilon}_1]_{qq} \sim \text{Bern}(p^1)$, a Q by Q matrix (zero off diag)

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

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

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

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



Feature space noise to weight space

- Can transform noise to param (weight) space

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

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

- Can transform noise to param (weight) space

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

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 $\epsilon_{gg} = \text{Bernoulli}(p^1)$ 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 $\lambda^1 = \sigma^2 \frac{1-p^1}{s^2 N}$ ..

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

- 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 $\lambda^1 = \sigma^2 \frac{1-p^1}{s^2 N}$..

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 $\lambda^1 = \sigma^2 \frac{1-p^1}{s^2 N}$..

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:

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:

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:

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:

Dropout uncertainty example

UNIVERSITY OF OXFORD

Define model and train on data x_train, y_train:

```
from tensorflow.keras.layers import Input, Dense, Dropou
from tf.keras.regularizers import l2

reg = sigma**2 * (1-p) / (s**2 * N)

inputs = Input(shape=(512,))
x = Dense(1024, activation="relu",
          kernel_regularizer=l2(reg))(inputs)
x = Dropout(p)(x, training=True)
x = Dense(1024, activation="relu",
          kernel_regularizer=l2(reg))(x)
x = Dropout(p)(x, training=True)
outputs = Dense(1, kernel_regularizer=l2(reg))(x)

model = tf.keras.Model(inputs, outputs)
model.compile(loss="mean_squared_error",
              optimizer="adam")
model.fit(x_train, y_train)
```

Epistemic uncertainty in regression BNNs

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...

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

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...

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

Epistemic uncertainty in regression BNNs

Predictive mean:

$$\mathbb{E}_{p(y^*|x^*,\mathcal{D})}[y^*]$$

$$= \int p(y^*|x^*,\mathcal{D})\, y^*\, dy^*$$

$$= \int \left(\int p(y^*|x^*,w)\, p(w|\mathcal{D})\, dw \right) y^*\, dy^*$$

$$= \int \left(\int y^* \underbrace{p(y^*|x^*,w)}_{=\mathcal{N}(f^w(x),\sigma^2)} dy^* \right) p(w|\mathcal{D})\, dw$$

$$= \int f^w(x) \underbrace{p(w|\mathcal{D})}_{\approx q(w)} dw$$

VI

$$\approx \int f^w(x)\, q(w)\, dw$$

MC

$$\approx \frac{1}{T} \sum_t f^{\hat{w}_t}(x), \quad \hat{w}_t \sim q(w)$$

Epistemic uncertainty in regression BNNs

UNIVERSITY OF OXFORD

Predictive Variance:

$$\mathbb{E}_{p(y^*|x^*,\mathcal{D})}[y^{*2}]$$

$$= \int \left(\underbrace{\int y^{*2} p(y^*|x^*,w)\, dy^*}_{\mathbb{E}_{lik}[y^{*2}] = \mathrm{Var}_{lik}(y^*) + \mathbb{E}_{lik}[y^*]^2} \right) p(w|\mathcal{D})\, dw$$

$$\overset{lik}{=} \int \left(\sigma^2 + f^w(x)^2 \right) p(w|\mathcal{D})\, dw$$

$$\overset{VI}{\approx} \sigma^2 + \int f^w(x)^2\, q(w)\, dw$$

$$\overset{MC}{\approx} \sigma^2 + \frac{1}{T} \sum_t f^{\hat{w}_t}(x)^2$$

$$\Rightarrow \mathrm{Var}_{p(y^*|x^*,\mathcal{D})} = \mathbb{E}_{p(y^*|x^*,\mathcal{D})}[y^{*2}] - \mathbb{E}_{p(y^*|x^*,\mathcal{D})}[y^*]^2$$

$$\approx \sigma^2 + \frac{1}{T} \sum_t f^{\hat{w}_t}(x)^2 - \left(\frac{1}{T} \sum_t f^{\hat{w}_t}(x) \right)^2$$

Example

Do stochastic forward passes on x_test:

```
num_MC_samples = 100
MC_samples = [model.predict(x_test)
                    for _ in range(num_MC_samples)]
```

Predictive mean

```
np.mean(MC_samples, axis=0)
```

Predictive variance

```
sigma**2 + np.var(MC_samples, axis=0)
```

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

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

How to visualise BNNs?

A useful tool for debugging

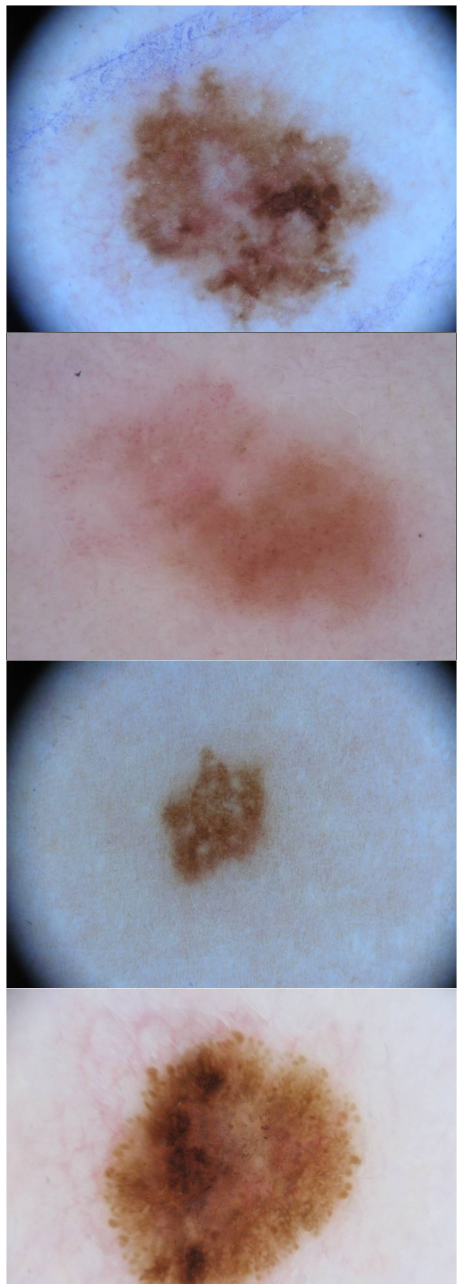
- sample from weights $\omega \sim p(\omega|\mathcal{D})$ = function sample $f^{\omega}(\cdot)$
- evaluate over interval $[-10, 10]$
- eg:
  - sample ω and def $f^{\omega}(\cdot)$
  - for each x_i in $\{-10, -9.95, -9.9, ..., 9.9, 9.95, 10\}$
    - evaluate $y_i = f^{\omega}(x_i)$ and plot (x_i, y_i)
- 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

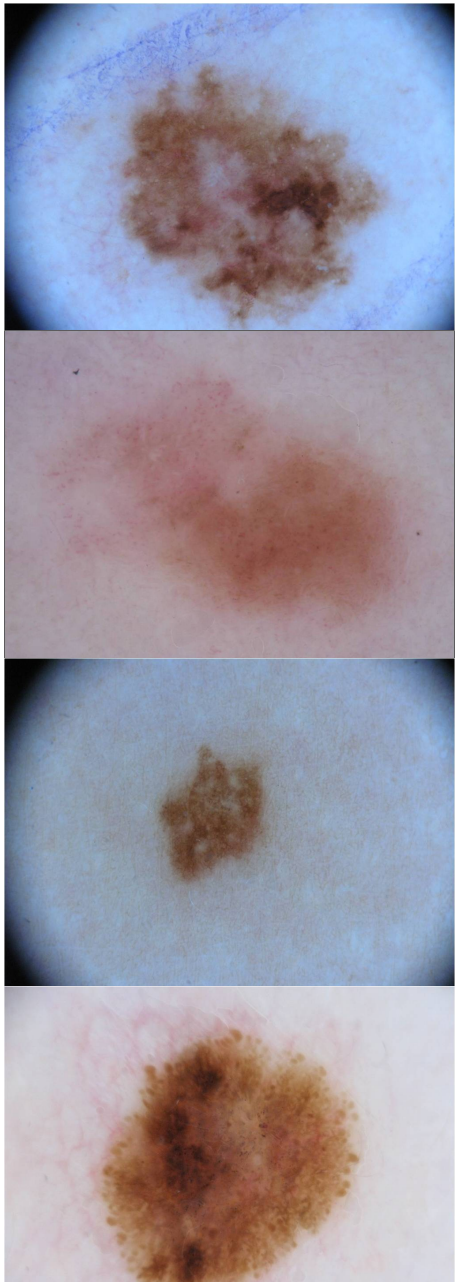
UNIVERSITY OF OXFORD

- 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*



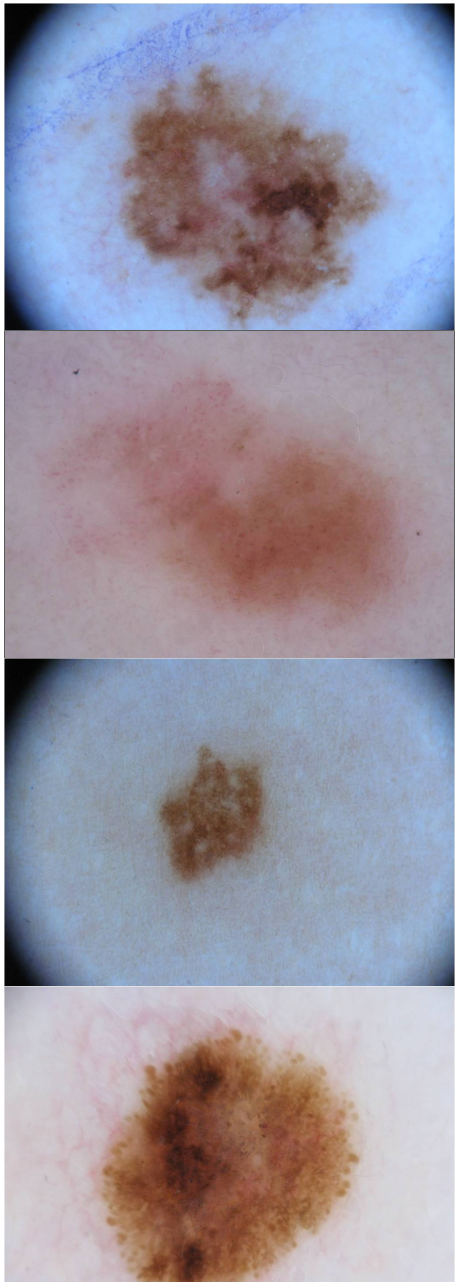
Small data big models

- 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*



Small data big models

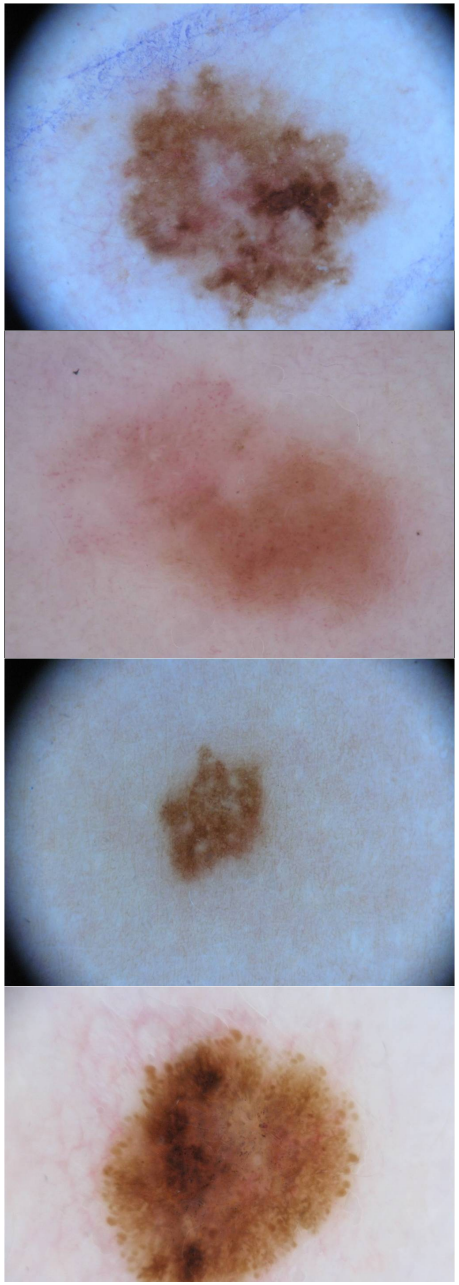
- we use machine learning to aid experts working in laborious fields
- automate small parts of the expert's work
  - eg melanoma (cancer) diagnosis based on lesion images
- but deep learning often requires large amounts of labelled data
  - increases with the complexity of problem
  - complexity of the 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*



Small data big models

UNIVERSITY OF OXFORD

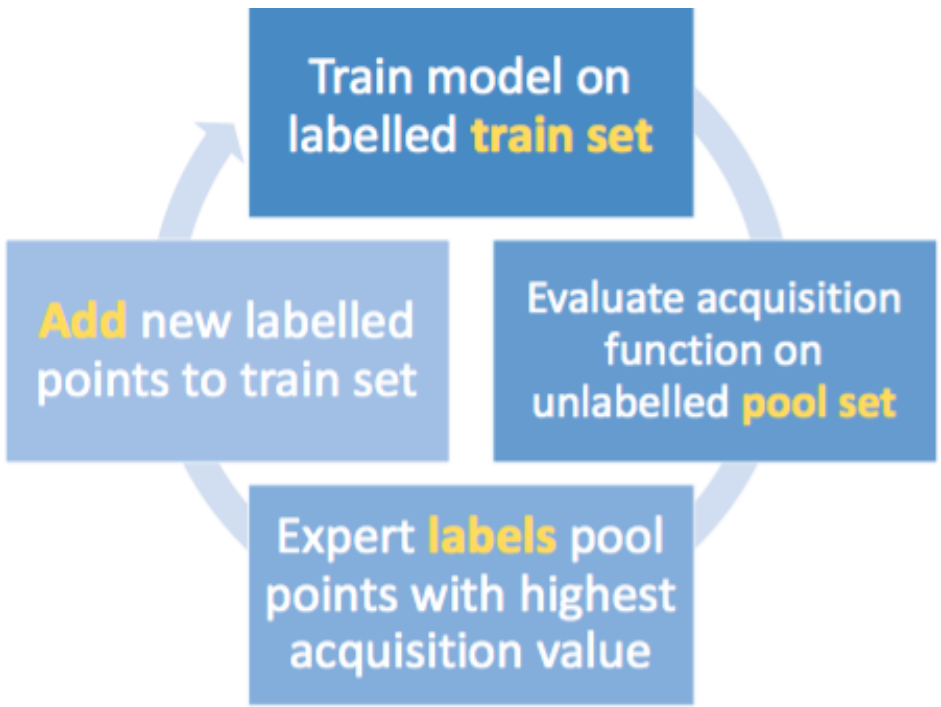
- we use machine learning to aid experts working in laborious fields
- automate small parts of the expert's work
  - eg melanoma (cancer) diagnosis based on lesion images
- but deep learning often requires large amounts of labelled data
  - increases with the complexity of problem
  - complexity of the 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*



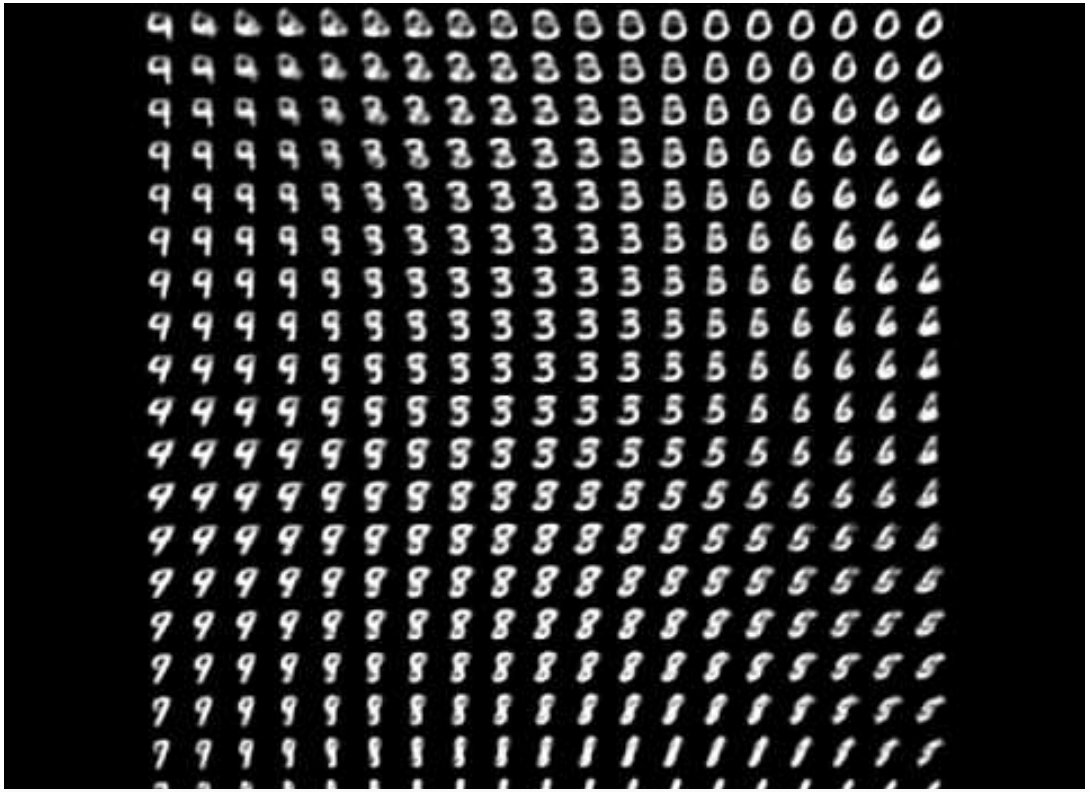
Principles of active learning

UNIVERSITY OF OXFORD

- 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



MNIST active learning



```
1 ...
2 model.compile(loss="categorical_crossentropy",
3                optimizer="adam")
4 model.fit(x_train, y_train)
5 MC_samples = [model.predict(x_test) for _ in range(20)]
```

Need uncertainty for classification...

MNIST active learning

```
...
model.compile(loss="categorical_crossentropy",
              optimizer="adam")
model.fit(x_train, y_train)
MC_samples = [model.predict(x_test) for _ in range(20)]
```

Predictive entropy

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

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

```
expected_p = np.mean(MC_samples, axis=0)
predictive_entropy = -np.sum(expected_p *
                             np.log(expected_p), axis=-1)
```

MNIST active learning

```
...
model.compile(loss="categorical_crossentropy",
              optimizer="adam")
model.fit(x_train, y_train)
MC_samples = [model.predict(x_test) for _ in range(20)]
```

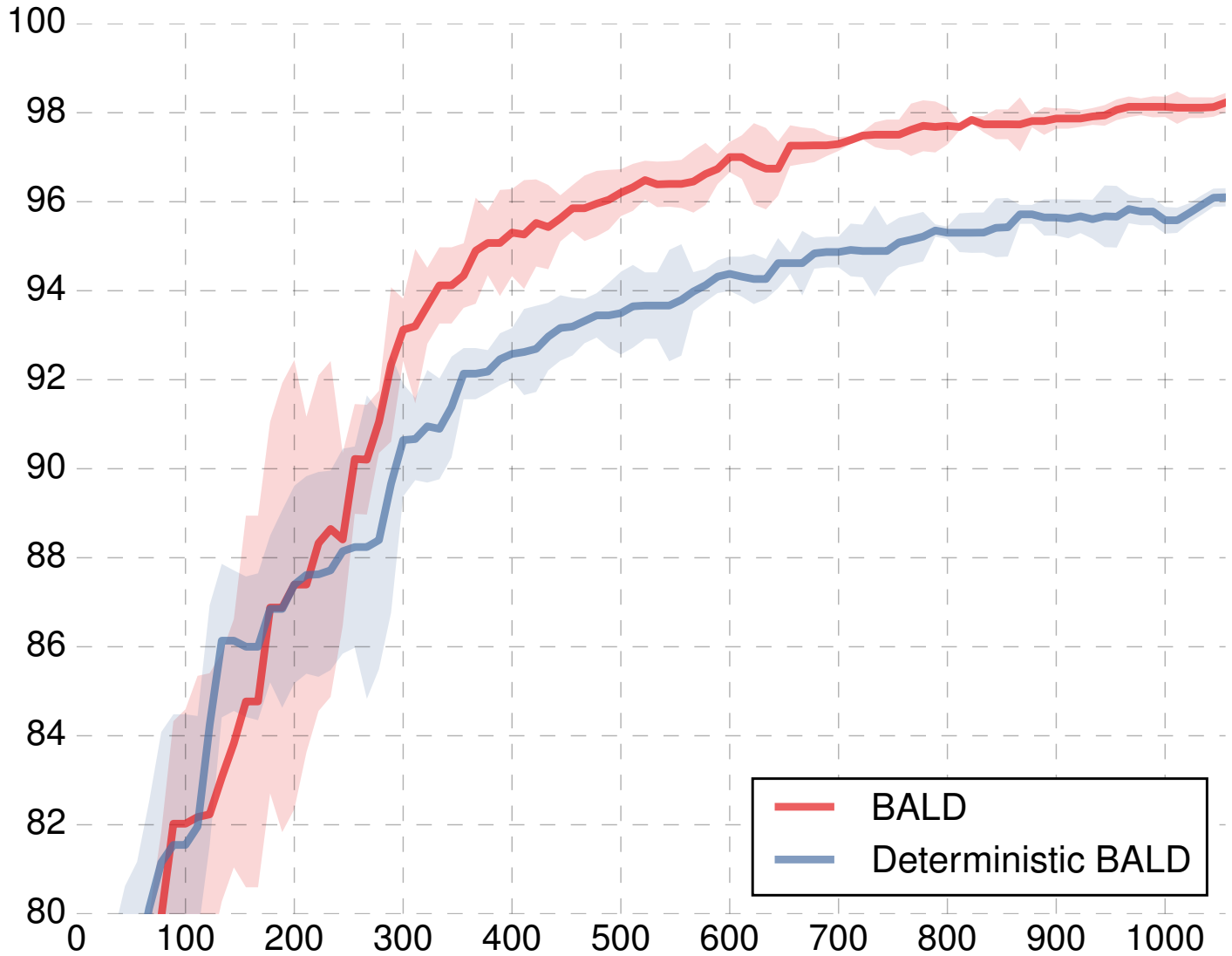
Mutual information (epistemic uncertainty)

$$MI(y^*, W|\mathcal{D}, x^*) = H_{p(y^*|x^*,\mathcal{D})}[y^*] - \frac{1}{T}\sum_{t,y^*=c} p^{\hat{W}_t}(x^*)_c \log p^{\hat{W}_t}(x^*)_c$$

```
MC_entropy = np.sum(MC_samples * np.log(MC_samples),
                    axis=-1)
expected_entropy = -np.mean(MC_entropy, axis=0)
mi = predictive_entropy - expected_entropy
```

Active learning applications

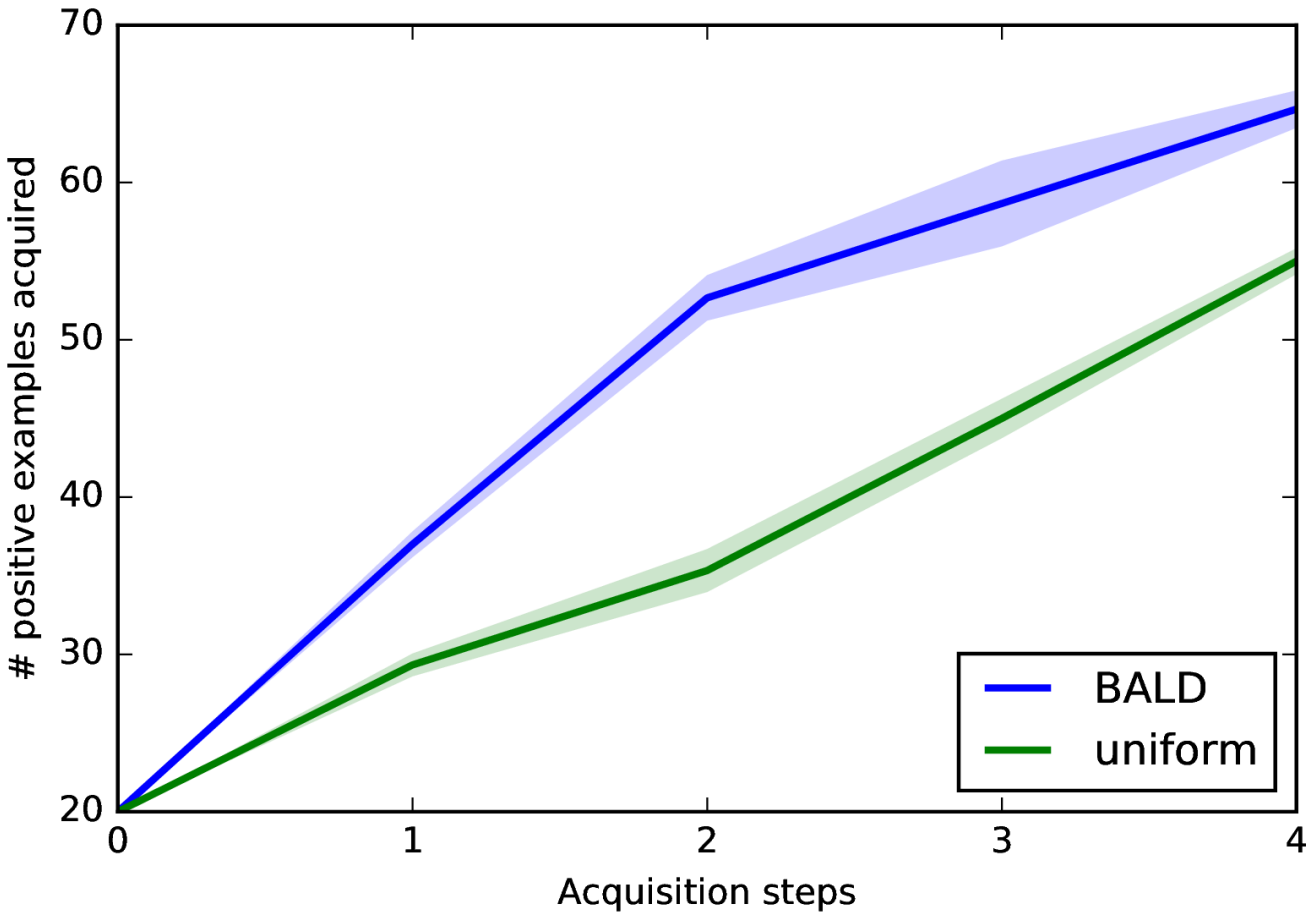
MNIST with only 1,000 images (instead of 60,000)

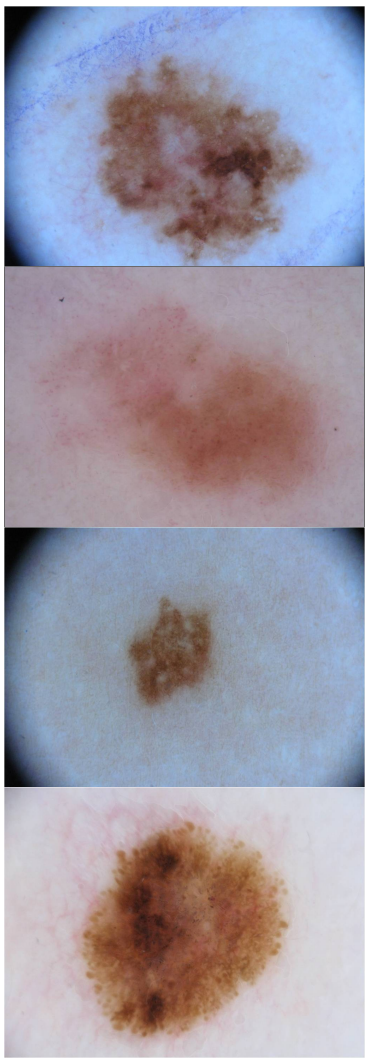


Active learning applications

UNIVERSITY OF OXFORD

Melanoma diagnosis with 300 images



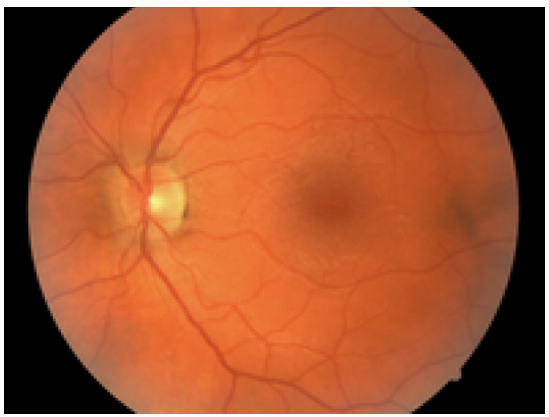


acquired positive examples vs. acquisition

Diabetes retinopathy diagnostics

UNIVERSITY OF OXFORD

- goal is to detect diabetes, and be able to tell when model is guessing at random
- used to pre-screen patients, send only patients with high uncertainty to expert



```
t = tf.layers.Conv2D(initial_conv_
                     kernel_regul
t = tf.layers.dropout(t, rate=drop
t = tf.layers.MaxPooling2D(pool_st

t = tf.layers.Conv2D(initial_conv_
                     kernel_regul
t = tf.layers.dropout(t, rate=drop
t = tf.layers.Conv2D(initial_conv_
                     padding="same
t = tf.layers.dropout(t, rate=drop
t = tf.layers.MaxPooling2D(pool_st
```

Quality of uncertainty measures

How to tell if uncertainty is good or bad?

- define a binary event: 'is diabetes?'; group test set inputs by **prediction** 'yes'/'no' vs **label** 'yes'/'no'
- each corresponds to one of TP, FP, FN, TN
- TPR and FPR are rates of TP and FP
  - TPR = sensitivity = recall = TP / (TP + FN) = 1 - FNR
  - TNR = specificity = TN / (TN + FP) = 1 - FPR
  - FPR = FP / (TN + FP) = 1 - specificity
- want TPR to be high, FPR to be low
- usually given reqs what's the worst we're allowed to perform in order to deploy system
- eg TPR=0.7 and FPR=0.1

Quality of uncertainty measures

How to tell if uncertainty is good or bad?

- define a binary event: 'is diabetes?'; group test set inputs by **prediction** 'yes'/'no' vs **label** 'yes'/'no'
- each corresponds to one of TP, FP, FN, TN
- TPR and FPR are rates of TP and FP
  - TPR = sensitivity = recall = TP / (TP + FN) = 1 - FNR
  - TNR = specificity = TN / (TN + FP) = 1 - FPR
  - FPR = FP / (TN + FP) = 1 - specificity
- want TPR to be high, FPR to be low
- usually given reqs what's the worst we're allowed to perform in order to deploy system
- eg TPR=0.7 and FPR=0.1

Quality of uncertainty measures

How to tell if uncertainty is good or bad?

- define a binary event: 'is diabetes?'; group test set inputs by **prediction** 'yes'/'no' vs **label** 'yes'/'no'
- each corresponds to one of TP, FP, FN, TN
- TPR and FPR are rates of TP and FP
  - TPR = sensitivity = recall = TP / (TP + FN) = 1 - FNR
  - TNR = specificity = TN / (TN + FP) = 1 - FPR
  - FPR = FP / (TN + FP) = 1 - specificity
- want TPR to be high, FPR to be low
- usually given reqs what's the worst we're allowed to perform in order to deploy system
- eg TPR=0.7 and FPR=0.1

Quality of uncertainty measures

How to tell if uncertainty is good or bad?

- define a binary event: 'is diabetes?'; group test set inputs by **prediction** 'yes'/'no' vs **label** 'yes'/'no'
- each corresponds to one of TP, FP, FN, TN
- TPR and FPR are rates of TP and FP
  - TPR = sensitivity = recall = TP / (TP + FN) = 1 - FNR
  - TNR = specificity = TN / (TN + FP) = 1 - FPR
  - FPR = FP / (TN + FP) = 1 - specificity
- want TPR to be high, FPR to be low
- usually given reqs what's the worst we're allowed to perform in order to deploy system
- eg TPR=0.7 and FPR=0.1

Quality of uncertainty measures

How to tell if uncertainty is good or bad?

- define a binary event: 'is diabetes?'; group test set inputs by **prediction** 'yes'/'no' vs **label** 'yes'/'no'
- each corresponds to one of TP, FP, FN, TN
- TPR and FPR are rates of TP and FP
  - TPR = sensitivity = recall = TP / (TP + FN) = 1 - FNR
  - TNR = specificity = TN / (TN + FP) = 1 - FPR
  - FPR = FP / (TN + FP) = 1 - specificity
- want TPR to be high, FPR to be low
- usually given reqs what's the worst we're allowed to perform in order to deploy system
- eg TPR=0.7 and FPR=0.1

Quality of uncertainty measures

- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures

- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures

- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures

- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures

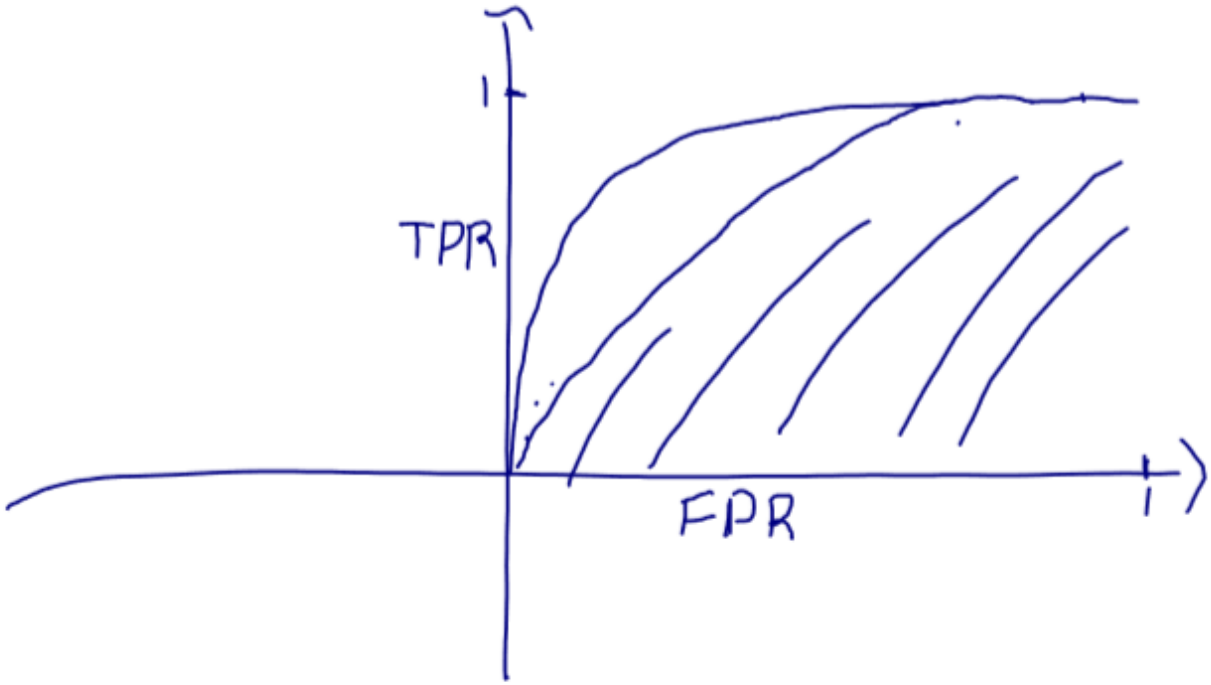
- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures

UNIVERSITY OF OXFORD

- model outputs a predictive prob $p(y|x, D)$; how do we get a recommendation 'yes'/'no'?
  - easiest is to take argmax
  - but what if model outputs 0.51? is this a 'yes'?
- def a threshold t
- if predictive prob is higher than t then say 'yes' otherwise say 'no'
  - for t = 0 says 'yes' to all, ie FN=TN=0, and model has TPR=1,FPR=1
  - for t = 1 says 'no' to all, ie TP=FP=0 and model has TPR=0, FPR=0
- each threshold t gives us a pair (FPR, TPR)
- scatter points for all t (or some discrete steps t)
- this is an ROC plot

Quality of uncertainty measures



Quality of uncertainty measures

- ROC shows tradeoff between TPR and FPR
- each point on the plot corresponds to a choice of t which will give that tradeoff
- aim: find a model which gives highest Area Under Curve (AUC)
  - allows for better tradeoffs generally
  - but not always
- how can we improve AUC? one solution:
  - identify patients for which you are guessing at random (uncertain)
  - select 10% patients you are most uncertain about and remove from test set (send to expert)
  - plot ROC for remaining 90% test set patients
  - if uncertainty correlates to patients you were mistaken on, ROC should improve (higher AUC)

Quality of uncertainty measures

- ROC shows tradeoff between TPR and FPR
- each point on the plot corresponds to a choice of t which will give that tradeoff
- aim: find a model which gives highest Area Under Curve (AUC)
  - allows for better tradeoffs generally
  - but not always
- how can we improve AUC? one solution:
  - identify patients for which you are guessing at random (uncertain)
  - select 10% patients you are most uncertain about and remove from test set (send to expert)
  - plot ROC for remaining 90% test set patients
  - if uncertainty correlates to patients you were mistaken on, ROC should improve (higher AUC)

Quality of uncertainty measures

UNIVERSITY OF OXFORD

- ROC shows tradeoff between TPR and FPR
- each point on the plot corresponds to a choice of t which will give that tradeoff
- aim: find a model which gives highest Area Under Curve (AUC)
  - allows for better tradeoffs generally
  - but not always
- how can we improve AUC? one solution:
  - identify patients for which you are guessing at random (uncertain)
  - select 10% patients you are most uncertain about and remove from test set (send to expert)
  - plot ROC for remaining 90% test set patients
  - if uncertainty correlates to patients you were mistaken on, ROC should improve (higher AUC)

Quality of uncertainty measures

UNIVERSITY OF OXFORD

- ROC shows tradeoff between TPR and FPR
- each point on the plot corresponds to a choice of t which will give that tradeoff
- aim: find a model which gives highest Area Under Curve (AUC)
  - allows for better tradeoffs generally
  - but not always
- how can we improve AUC? one solution:
  - identify patients for which you are guessing at random (uncertain)
  - select 10% patients you are most uncertain about and remove from test set (send to expert)
  - plot ROC for remaining 90% test set patients
  - if uncertainty correlates to patients you were mistaken on, ROC should improve (higher AUC)

Quality of uncertainty measures

Send patients to expert diagnosis if model is uncertain

- use some uncertainty metric to refuse to diagnose a patient if model is uncertain
- what uncertainty measure?
  - MI would be high for far away points but will keep ambiguous points in test set
  - (points for which expert annotation in dataset was noisy)
  - expected entropy would be high for both far away inputs (entropy $\geq$ MI) and ambiguous inputs
- $\rightarrow$ use expected entropy
- can we improve tradeoff by sending a small number of patients to an expert in a real-world system?

Quality of uncertainty measures

Send patients to expert diagnosis if model is uncertain

- use some uncertainty metric to refuse to diagnose a patient if model is uncertain
- what uncertainty measure?
  - MI would be high for far away points but will keep ambiguous points in test set
  - (points for which expert annotation in dataset was noisy)
  - expected entropy would be high for both far away inputs (entropy $\geq$ MI) and ambiguous inputs
- $\rightarrow$ use expected entropy
- can we improve tradeoff by sending a small number of patients to an expert in a real-world system?

Quality of uncertainty measures

Send patients to expert diagnosis if model is uncertain

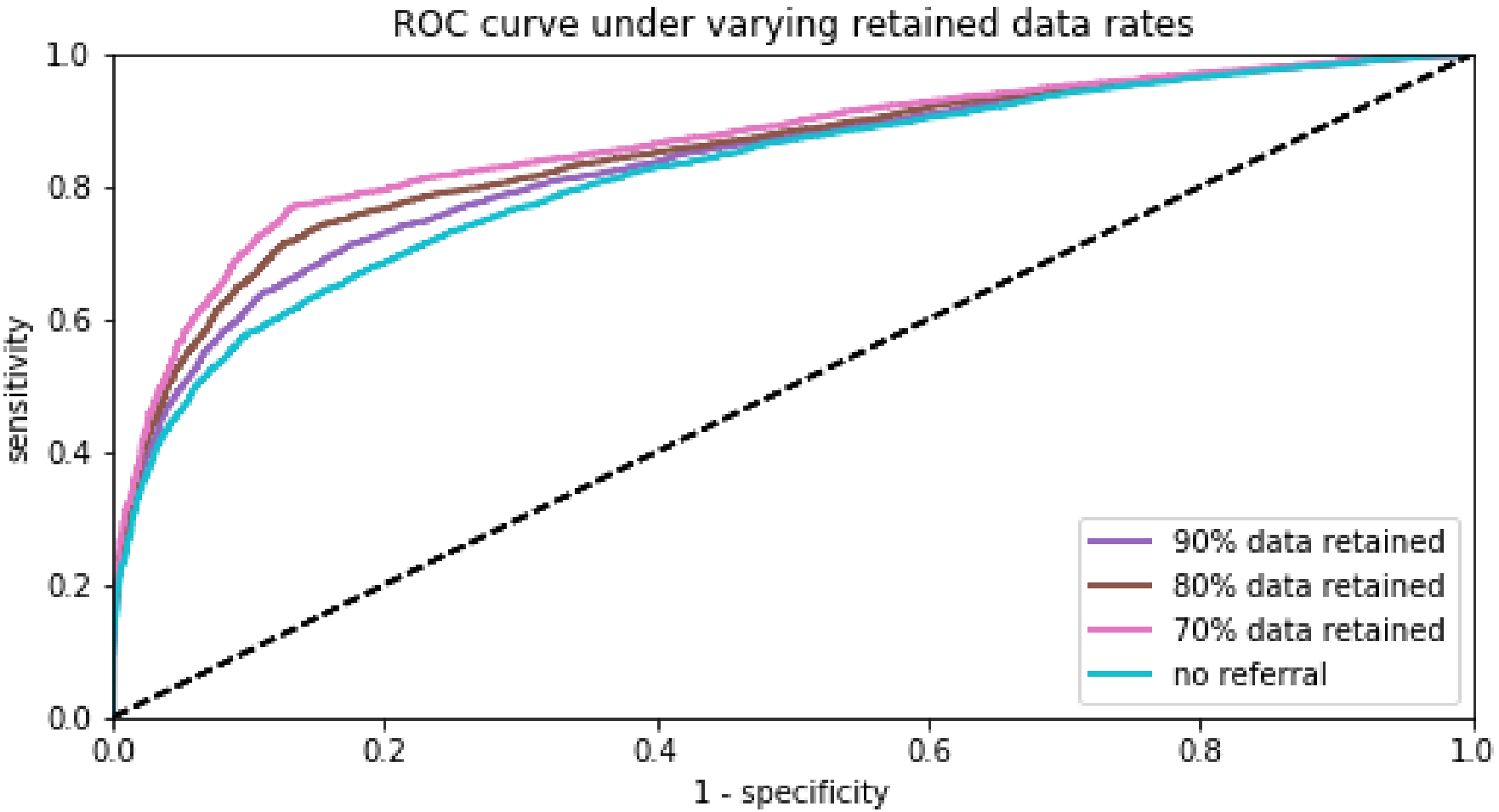
- use some uncertainty metric to refuse to diagnose a patient if model is uncertain
- what uncertainty measure?
  - MI would be high for far away points but will keep ambiguous points in test set
  - (points for which expert annotation in dataset was noisy)
  - expected entropy would be high for both far away inputs (entropy $\geq$ MI) and ambiguous inputs
- $\rightarrow$ use expected entropy
- can we improve tradeoff by sending a small number of patients to an expert in a real-world system?

Quality of uncertainty measures

Send patients to expert diagnosis if model is uncertain

- use some uncertainty metric to refuse to diagnose a patient if model is uncertain
- what uncertainty measure?
  - MI would be high for far away points but will keep ambiguous points in test set
  - (points for which expert annotation in dataset was noisy)
  - expected entropy would be high for both far away inputs (entropy $\geq$ MI) and ambiguous inputs
- $\rightarrow$ use expected entropy
- can we improve tradeoff by sending a small number of patients to an expert in a real-world system?

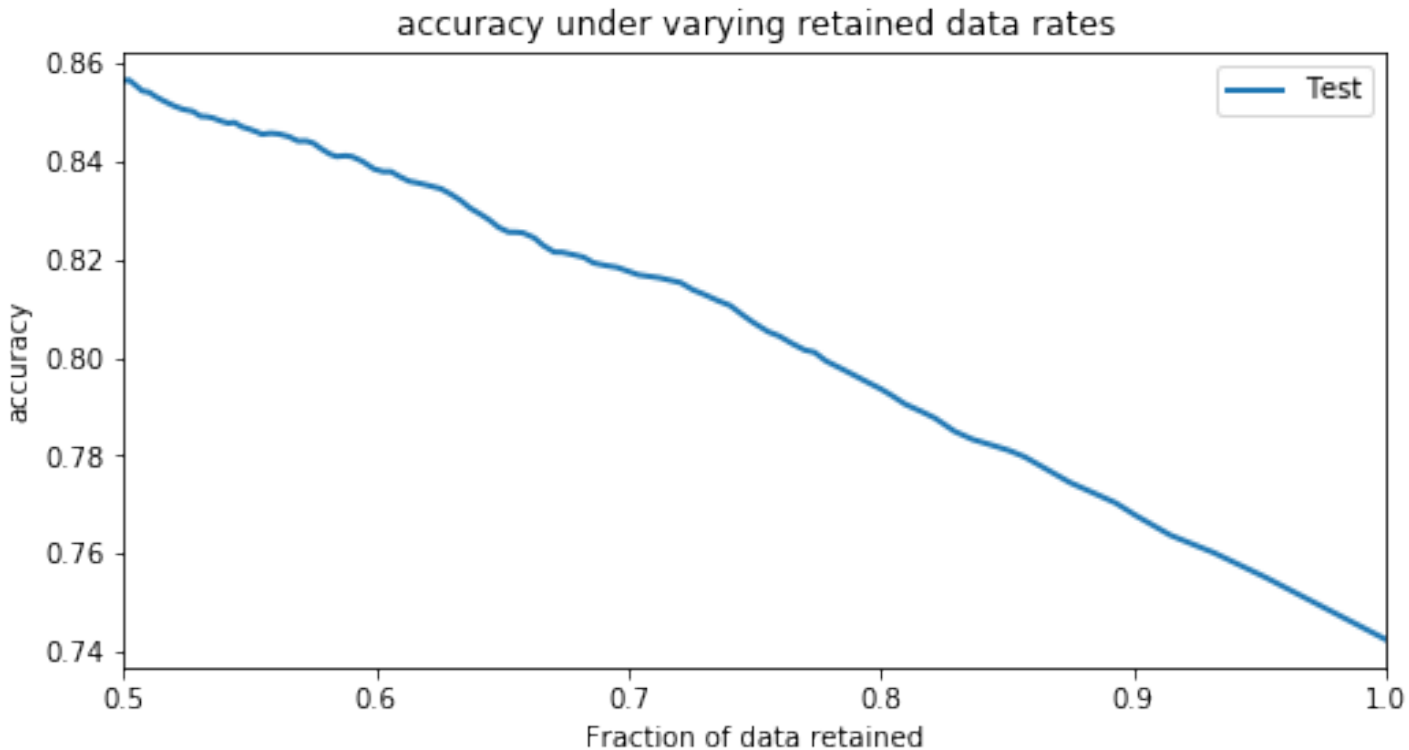
Quality of uncertainty measures



Quality of uncertainty measures

Another measure of uncertainty performance

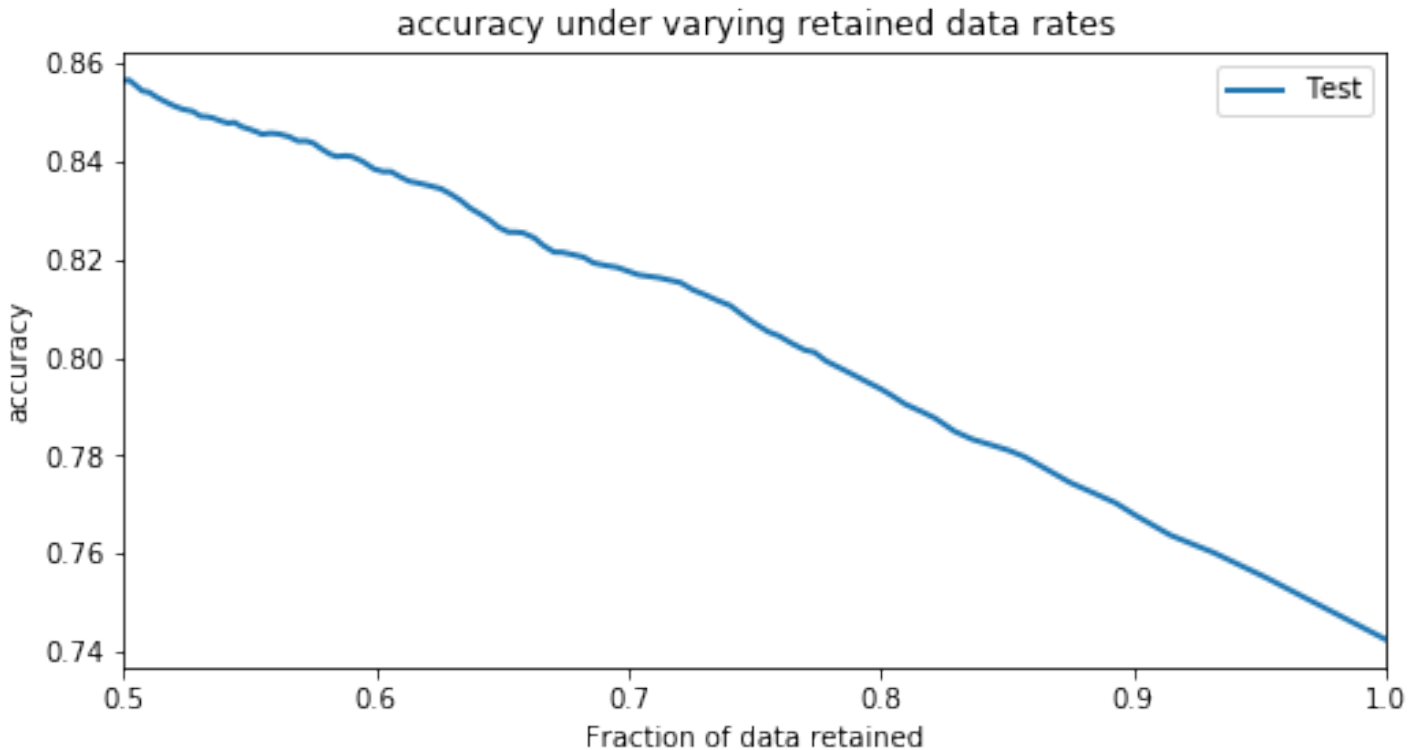
- plot accuracy as a function of % retained data, as sending more and more patients to an expert
- 100% retain data = original accuracy on full dataset
- 10% retain data = accuracy after removing 90% patients with highest uncertainty



Quality of uncertainty measures

Another measure of uncertainty performance

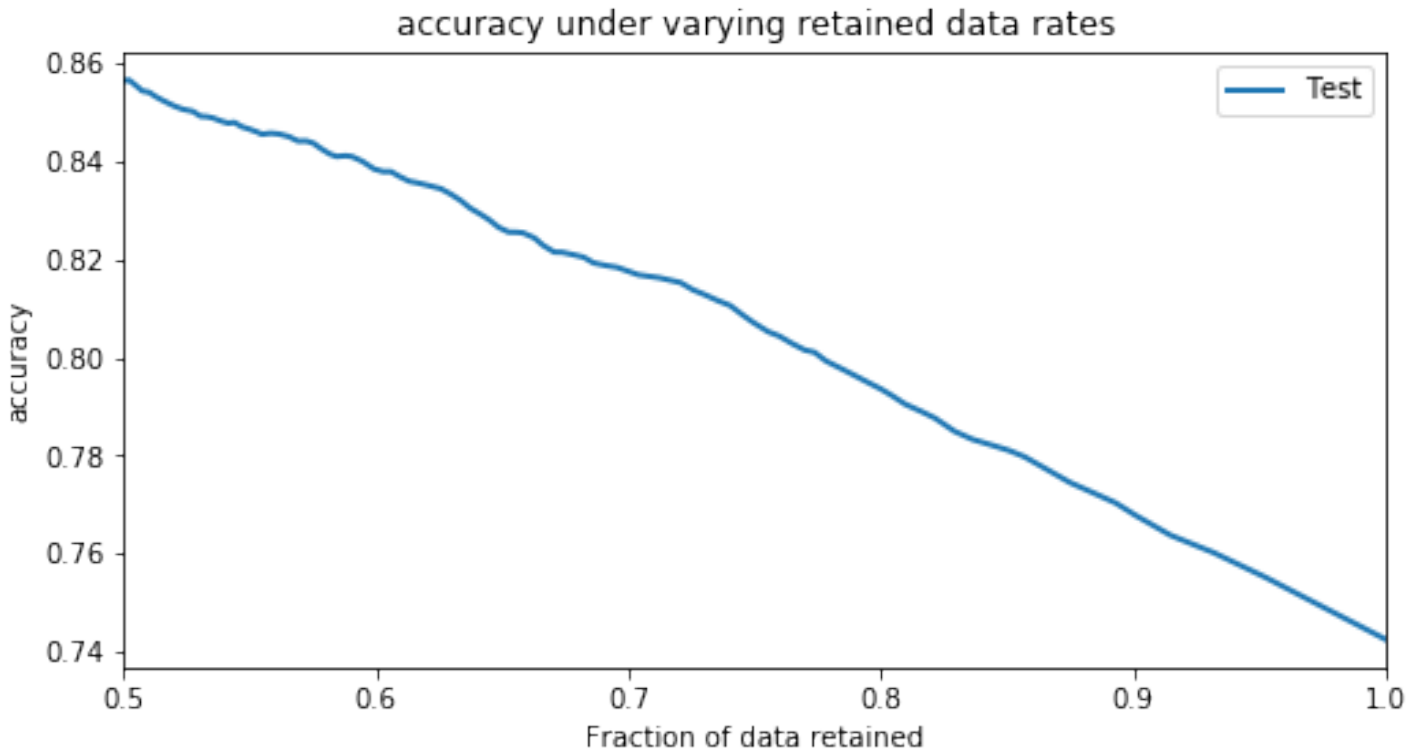
- plot accuracy as a function of % retained data, as sending more and more patients to an expert
- 100% retain data = original accuracy on full dataset
- 10% retain data = accuracy after removing 90% patients with highest uncertainty



Quality of uncertainty measures

Another measure of uncertainty performance

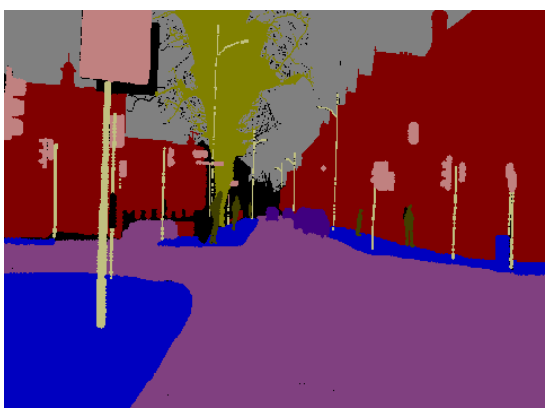
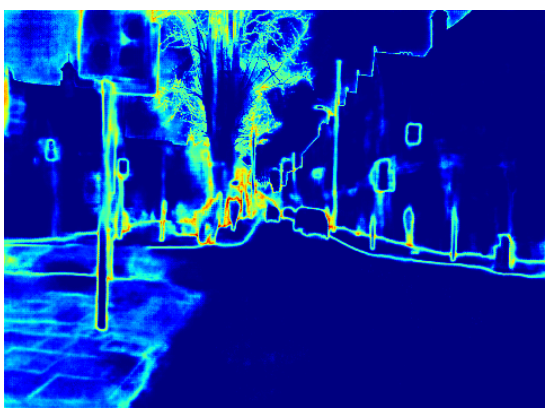
- plot accuracy as a function of % retained data, as sending more and more patients to an expert
- 100% retain data = original accuracy on full dataset
- 10% retain data = accuracy after removing 90% patients with highest uncertainty



Autonomous driving

We'll be looking at semantic segmentation

- input: image in RGB space
- output: image in semantic space
- each pixel is mapped to semantic class (eg road, sky, car, pedestrian) based on its context (near by pixels)

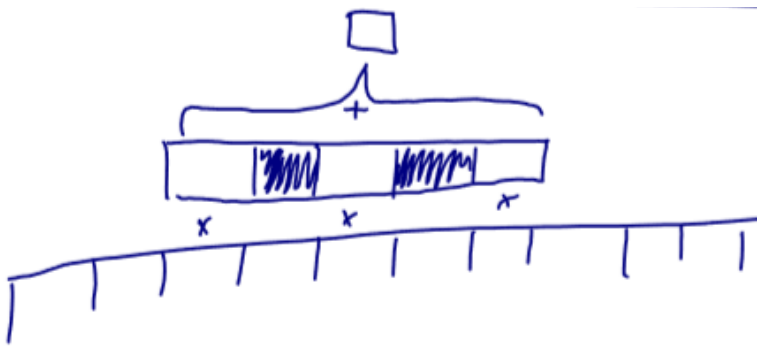




DeepLab

- one of the SOTA NNs for semantic segmentation is DeepLab
- uses atrous (dilated) convolutions (has 'holes')
  - widen field of view over the input feature maps without increasing parameters or pooling
- uses encoder-decoder architectures
  - upsampling replicates pixels then applies eg 1x1 conv which doesn't reduce dim
- can be applied to any base network ('backbone') as long as it is fully convolutional (ie no fully connected layers)

DeepLab

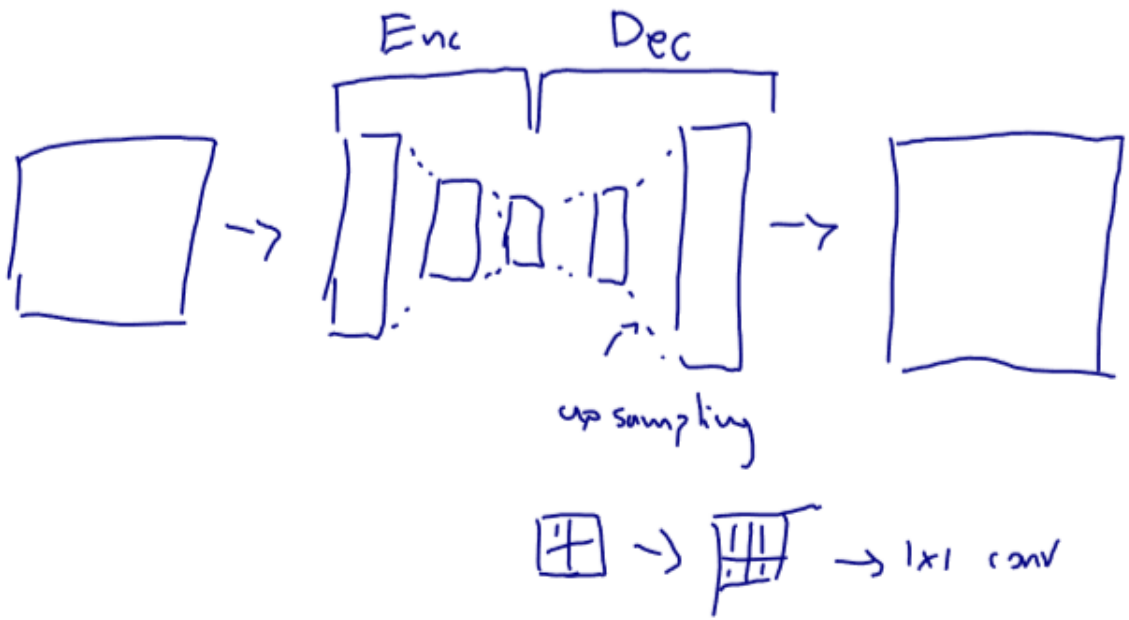
- one of the SOTA NNs for semantic segmentation is DeepLab
- uses atrous (dilated) convolutions (has 'holes')
  - widen field of view over the input feature maps without increasing parameters or pooling



- uses encoder-decoder architectures
  - upsampling replicates pixels then applies eg 1x1 conv which doesn't reduce dim
- can be applied to any base network ('backbone') as long as it is fully convolutional (ie no fully connected layers)

DeepLab

- one of the SOTA NNs for semantic segmentation is DeepLab
- uses atrous (dilated) convolutions (has 'holes')
  - widen field of view over the input feature maps without increasing parameters or pooling
- uses encoder-decoder architectures
  - upsampling replicates pixels then applies eg 1x1 conv which doesn't reduce dim



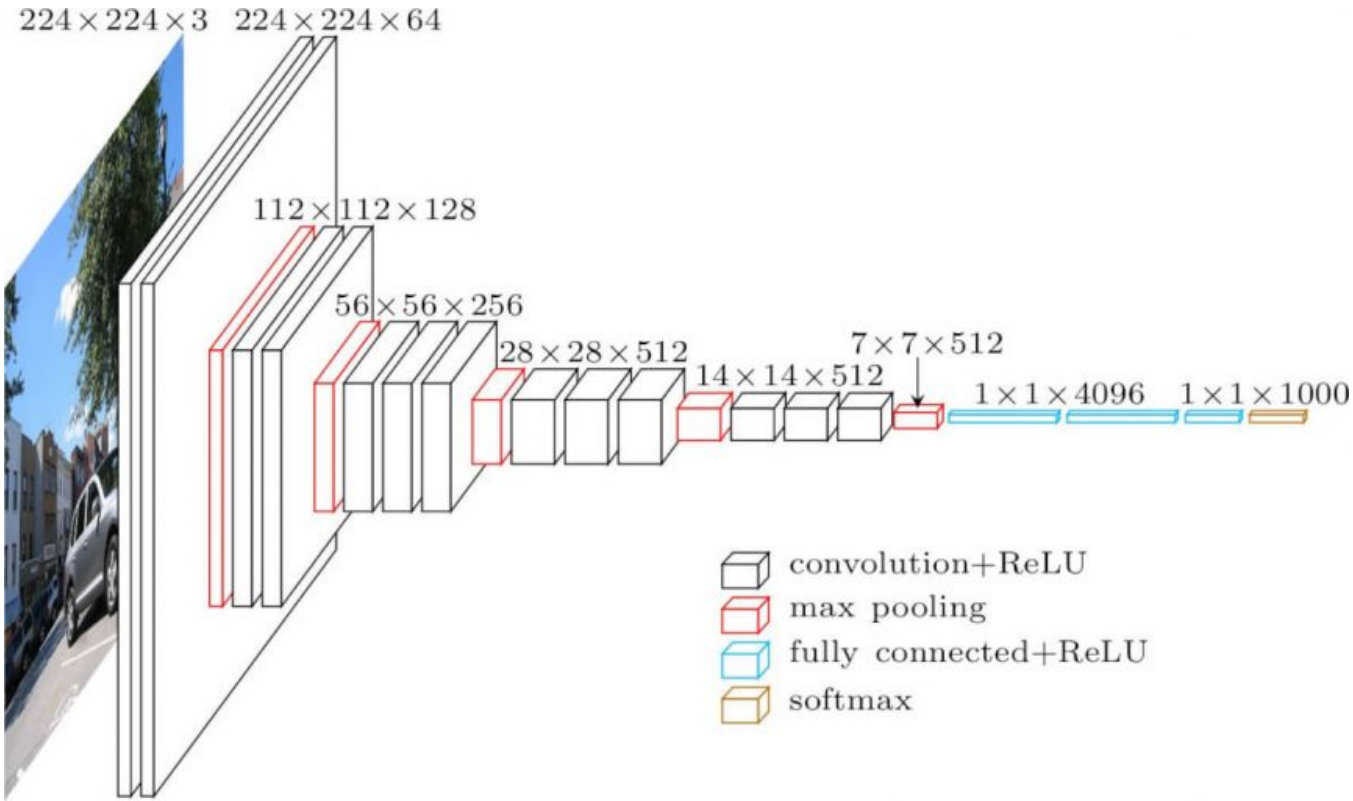
DeepLab

UNIVERSITY OF OXFORD

- one of the SOTA NNs for semantic segmentation is DeepLab
- uses atrous (dilated) convolutions (has 'holes')
  - widen field of view over the input feature maps without increasing parameters or pooling
- uses encoder-decoder architectures
  - upsampling replicates pixels then applies eg 1x1 conv which doesn't reduce dim
- can be applied to any base network ('backbone') as long as it is fully convolutional (ie no fully connected layers)

Backbone

- Popular deep CNNs backbones
  - VGG-16
  - ResNet101
  - Xception



neurohive.io/

- ResNet

Backbone

- Popular deep CNNs backbones
  - VGG-16
  - ResNet101
  - Xception
- ResNet
  - layer def
  - solves the issue of "diminishing gradient" in deep nets (bounding eigenvalues from below)
  - can use hundreds of layers - seems to improve results the more layers you use

$$f^{w'}(x) = (w' + I_2)\,x = w'x + x \quad (2\times 2)$$

$$y(x) = w\, f^{w'}(f^{w^2}(f^{w^3}(x)))$$

$$= w(w'+I)(w^2+I)(w^3+I)x$$

$$= w w' w^2 w^3 x + w w^2 w^3 x + w w^3 x + \dots$$

Backbone

- Popular deep CNNs backbones
  - VGG-16
  - ResNet101
  - Xception
- ResNet
  - layer def
  - solves the issue of "diminishing gradient" in deep nets (bounding eigenvalues from below)
  - can use hundreds of layers - seems to improve results the more layers you use

$$L = (y - y(x))^2$$

$$\frac{\partial L}{\partial w} = \underbrace{\frac{\partial L}{\partial y(x)}}_{-2(y-y(x))} \underbrace{\frac{\partial y(x)}{\partial w}}_{?}$$

Deep Net:

$$y(x) = w f^{w'}(f^{w^2}(f^{w^3}(x))) = w w' w^2 w^3 x$$

$$\partial y(x)/\partial w^3 = w w' w^2 x$$

ResNet:

$$\partial y(x)/\partial w^3 = w w' w^2 + w w^2 x + w w' x + w x$$

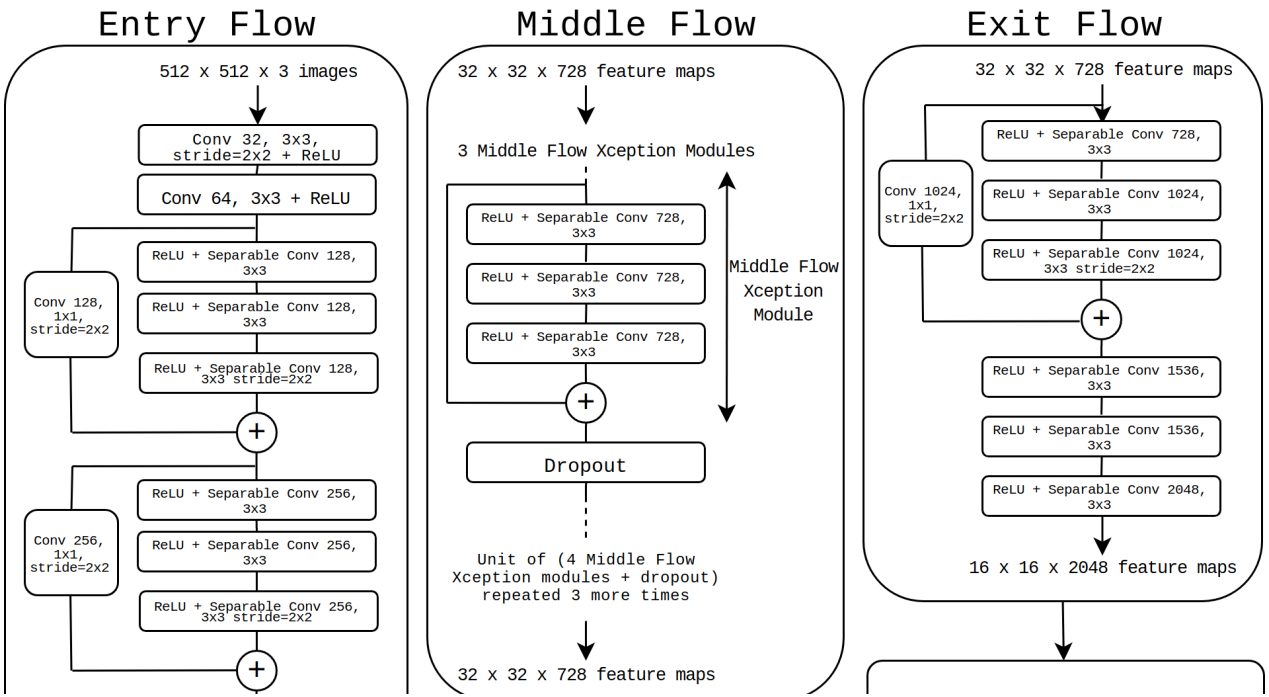
$$= w(w' w^2 + w^2 + w' + I) x$$

Xception

UNIVERSITY OF OXFORD

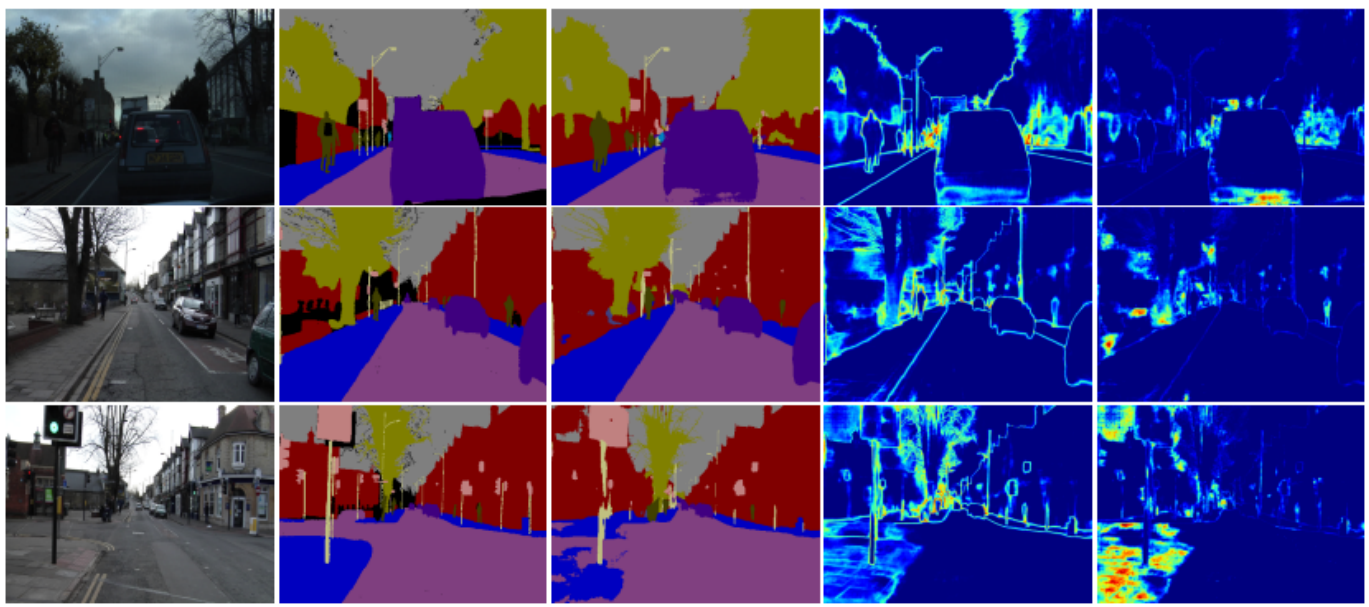
We use Xception

- architecture has simplicity of VGG with multiple convolution layers stacked on top of one another
- Xception modules use skip connections similar to ResNet but between blocks
- works well empirically



Uncertainties in segmentation

- we have a classification problem with H by W softmax outputs (categorical variable for each pixel)
- model loss: sum of cross entropy (log likelihoods) for each pixel
- can use standard tools for uncertainty in classification (per pixel)
- and look at epistemic and aleatoric uncertainty maps



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Figure 1. Qualitative results of our method on the CamVid semantic segmentation dataset, with three example input images. Our

More applications

UNIVERSITY OF OXFORD

www.nature.com/scientificreports

OPEN

Leveraging uncertainty information from deep neural networks for disease detection

24 July 2017
1 December 2017
online: 19 December 2017

Christian Leibig[1], Vaneeda Allken[1], Murat Seçkin Ayhan[1], Philipp Berens[1,2] & Siegfried Wahl[1,3]

Deep learning (DL) has revolutionized the field of computer vision and image processing. In medical imaging, algorithmic solutions based on DL have been shown to achieve high performance on tasks that previously required medical experts. However, DL-based solutions for disease detection have been proposed without methods to quantify and control their uncertainty in a decision. In contrast, a physician knows whether she is uncertain about a case and will consult more experienced colleagues if

More applications

www.nature.com/scientificreports

SC

Uncertainty-Aware Reinforcement Learning for Collision Avoidance

Gregory Kahn*, Adam Villaflor*, Vitchyr Pong*, Pieter Abbeel*†, Sergey Levine*
*Berkeley AI Research (BAIR), University of California, Berkeley
†OpenAI

: 24 July 2017
: 1 December 2017
online: 19 Decem

O ***Abstract*—Reinforcement learning can enable complex, adap-ive behavior to be learned automatically for autonomous robotic platforms. However, practical deployment of reinforcement learn-ng methods must contend with the fact that the training process tself can be unsafe for the robot. In this paper, we consider he specific case of a mobile robot learning to navigate an a priori unknown environment while avoiding collisions. In order o learn collision avoidance, the robot must experience collisions t training time. However, high-speed collisions, even at training ime, could damage the robot. A successful learning method nust therefore proceed cautiously, experiencing only low-speed ollisions until it gains confidence. To this end, we present an ncertainty-aware model-based learning algorithm that estimates he probability of collision together with a statistical estimate f uncertainty. By formulating an uncertainty-dependent cost unction, we show that the algorithm naturally chooses to proceed autiously in unfamiliar environments, and increases the velocity f the robot in settings where it has high confidence. Our redictive model is based on bootstrapped neural networks**

Fig. 1: **Uncertainty-aware collision prediction model for collision av ance**: A quadrotor and an RC car are tasked with navigating in an unkn environment. How should the robots navigate while avoiding collisions? propose a model-based reinforcement learning approach in which the r learns a collision prediction model by experiencing collisions at low sp which is unlikely to damage the vehicle. We formulate a velocity-depen collision cost that uses collision prediction estimates and their associa uncertainties to enable the robot to only experience safe collisions du training while still approaching the desired task performance.

catastrophic) collisions during training. The robot can overco this quandary by first experiencing gentle collisions in orde

More applications

UNIVERSITY OF OXFORD

www.nature.com/scientificreports

Uncertainty Aware Reinforcement Learning for

SC

O

Abstract—Rei
ive behavior to l
platforms. Howe
ng methods mus
tself can be un
he specific case
priori unknown
o learn collision
t training time.
ime, could dan
nust therefore p
ollisions until it
ncertainty-awar
he probability
f uncertainty.
unction, we shov
autiously in unf
f the robot in
predictive mode

24 July 2017
1 December 2017
online: 19 Decem

Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks

Michael Kampffmeyer*, Arnt-Børre Salberg† and Robert Jenssen*

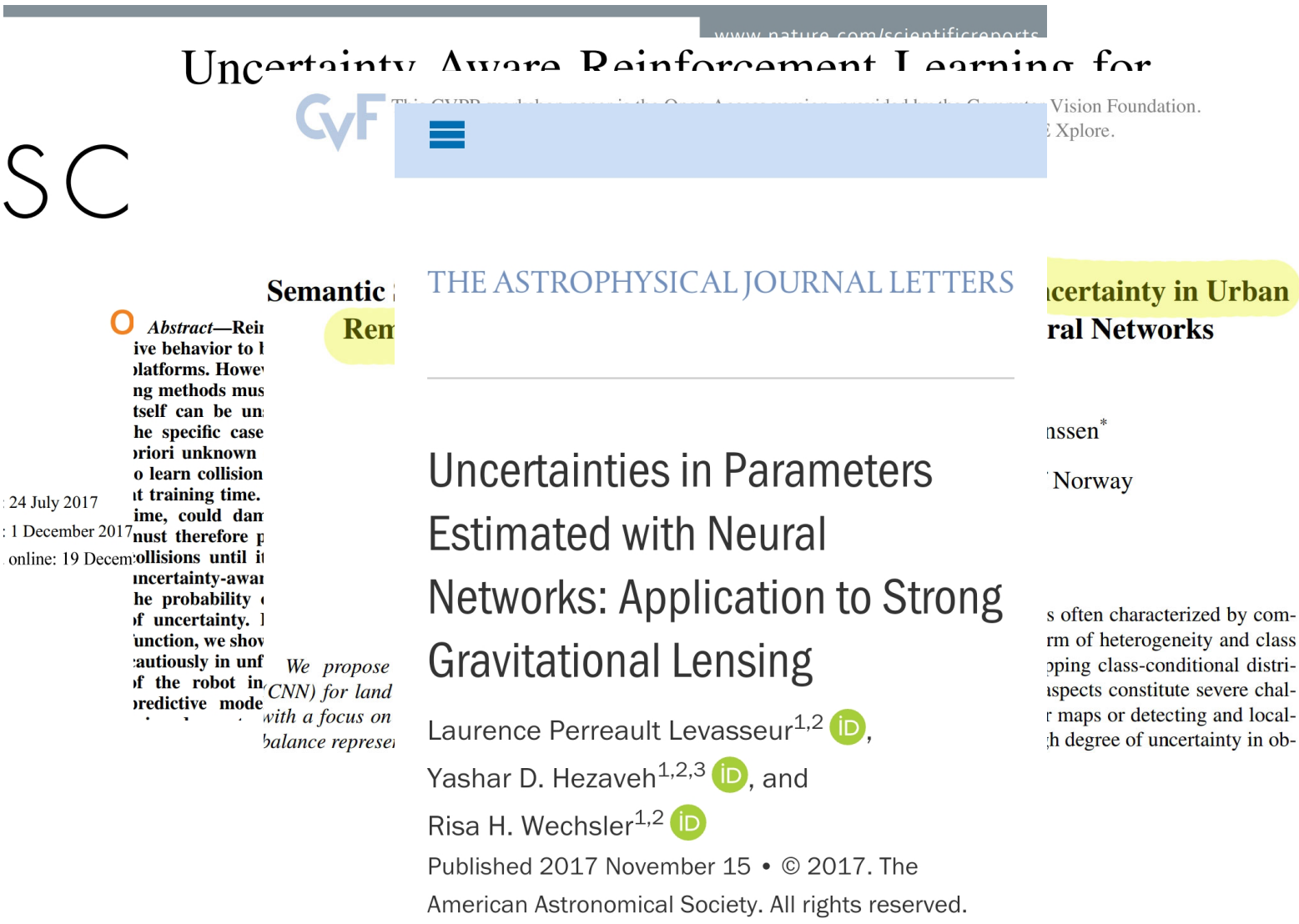
*Machine Learning @ UiT Lab, UiT–The Arctic University of Norway
†Norwegian Computing Center

Abstract

We propose a deep Convolutional Neural Network (CNN) for land cover mapping in remote sensing images, with a focus on urban areas. In remote sensing, class imbalance represents often a problem for tasks like land cover

Remote sensing imagery is often characterized by complex data properties in the form of heterogeneity and class imbalance, as well as overlapping class-conditional distributions [6]. Together, these aspects constitute severe challenges for creating land cover maps or detecting and localizing objects, producing a high degree of uncertainty in ob-

More applications



www.nature.com/scientificreports

Uncertainty Aware Reinforcement Learning for

SC

Semantic

Rem

Abstract—Rei
ive behavior to l
platforms. Howe
ng methods mus
tself can be un
he specific case
priori unknown
o learn collision
t training time.
ime, could dan
nust therefore p
ollisions until it
ncertainty-awar
he probability
f uncertainty.
unction, we show
autiously in unf
f the robot in
predictive mode

24 July 2017
1 December 2017
online: 19 Decem

We propose
CNN) for land
with a focus on
balance represe

THE ASTROPHYSICAL JOURNAL LETTERS

Uncertainties in Parameters Estimated with Neural Networks: Application to Strong Gravitational Lensing

Laurence Perreault Levasseur[1,2], Yashar D. Hezaveh[1,2,3], and Risa H. Wechsler[1,2]

Published 2017 November 15 • © 2017. The American Astronomical Society. All rights reserved.

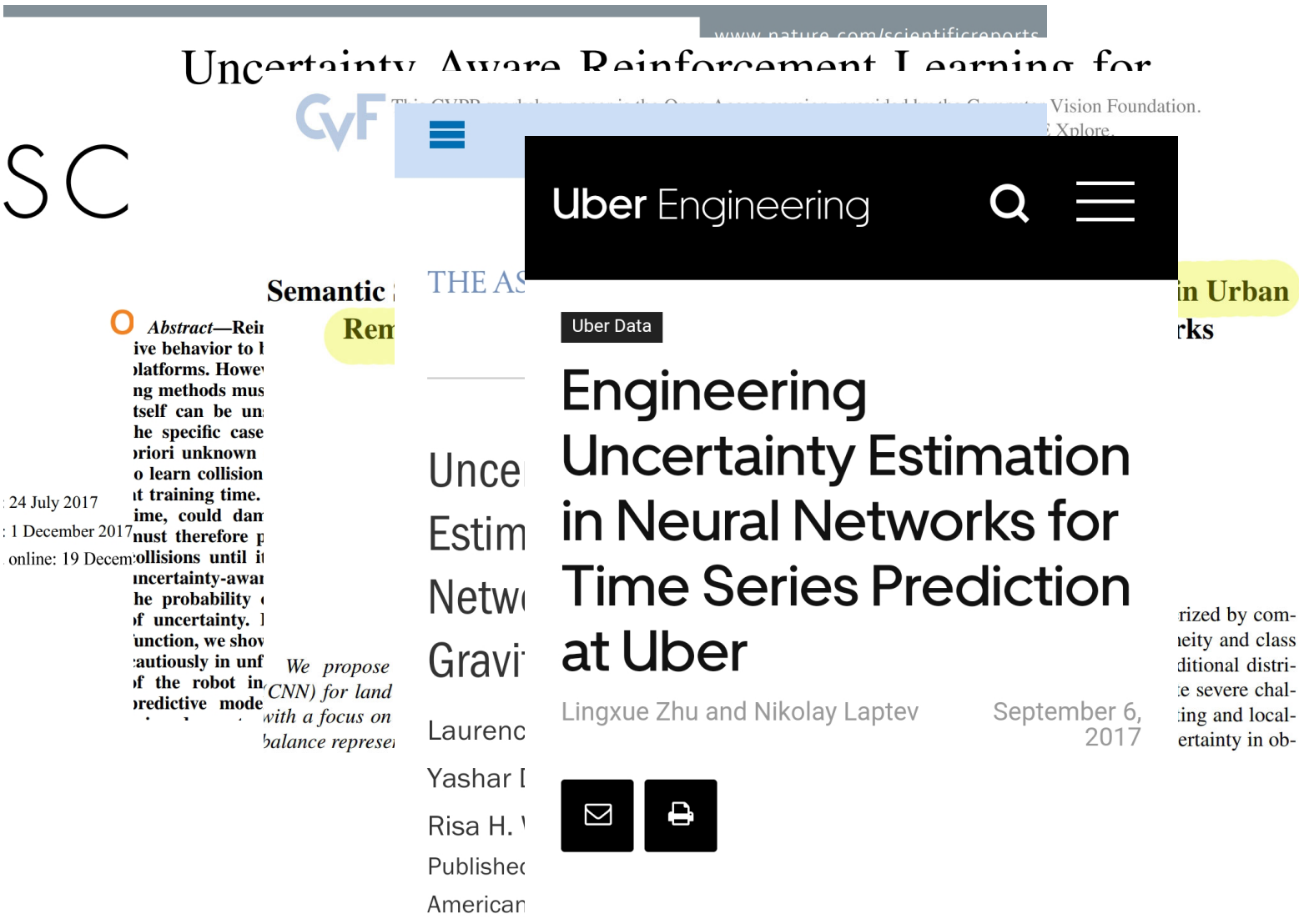
ncertainty in Urban
ral Networks

nssen*

Norway

s often characterized by com-
rm of heterogeneity and class
pping class-conditional distri-
aspects constitute severe chal-
r maps or detecting and local-
h degree of uncertainty in ob-

More applications



What you should be able to do now

UNIVERSITY OF OXFORD

- use uncertainty in regression **correctly**
- perform predictions in simple probabilistic models **efficiently**
- use Bayesian modelling in complex ML models (eg classification)
- use uncertainty (both epistemic and aleatoric) in real world models
- extend VI **correctly** to complex models
  - try to extend to new likelihoods like Laplace
  - try to extend to multiple outputs: categorical and continuous outputs
- do deep learning with small amounts of data
  - do try this at home!
- evaluate whether your uncertainty makes sense
- (somewhat) understand how huge deep vision systems work