Deep Bayesian Learning

Václav Šmídl,

Winter school of machine learning,
Czech Technical University
vasek.smidl@gmail.com

January 22, 2020

Overview

Extract from *Hierarchical Bayesian Models*, FJFI summer

- Lecture 1: How to be a Bayesian
- Lecture 2: Approximations and computational tools
- Lecture 3: Application to Deep Active Learning

Overview

Extract from *Hierarchical Bayesian Models*, FJFI summer

Lecture 1: How to be a Bayesian

Lecture 2: Approximations and computational tools

Lecture 3: Application to Deep Active Learning

Lecture 3:

Variational autoencoder

- Density estimation
- Generative model

Bayesian NN

- Uncertainty, prediction
- Active Learning
- Sampling methods

Density estimation

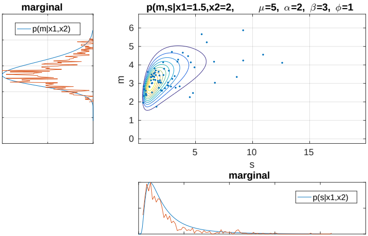
Inverse task to sampling

- We are given set of samples $X = \{x_i\}_{i=1}^n$,
- We want to find a generating distribution $p(x)$.
  - mean, variance

Density estimation

Inverse task to sampling

- We are given set of samples $X = \{x_i\}_{i=1}^{n}$,
- We want to find a generating distribution $p(x)$.
  - mean, variance
  - complex distributions
  - high dimensional distribution
- Application in anomaly detection (out of sample)
- Classical methods
  - one class SVM
  - kernel density estimator
  - mixture of Gaussians



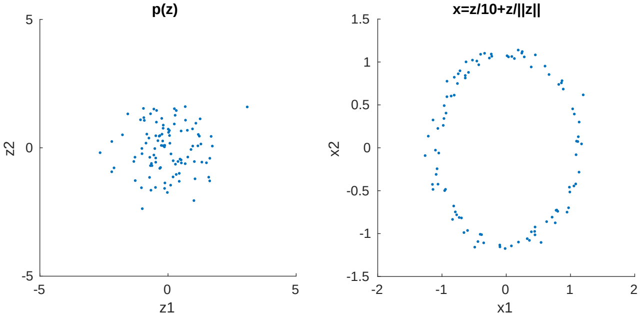
Variational Autoencoder

Generative model

$$x_i = f(z_i) + e_i, \qquad z_i \sim \mathcal{N}(0, I),$$
$$e_i \sim \mathcal{N}(0, \sigma I),$$

where z is the latent variable, and e is noise.

- The power comes from the $f()$
- For $f(z) = z/10 + z/||z||$



- Given samples $X = \{x\}_{i=1}^{n}$, find $f()$.

VAE: optimization problem

Choose parametric form $f_\theta(z)$

$$p(x|z) = \mathcal{N}(f_\theta(z_i), \sigma I), \qquad p(z) = \mathcal{N}(0, I),$$

We seek θ^*

$$\theta^* = \arg\max_\theta \prod_i p_\theta(x_i), \qquad p_\theta(x) = \int p(x|z, \theta)p(z)dz,$$

How to match data to an unknown solution of the integral? Mix. of Dirac?

VAE: optimization problem

Choose parametric form $f_\theta(z)$

$$p(x|z) = \mathcal{N}(f_\theta(z_i), \sigma I), \qquad p(z) = \mathcal{N}(0, I),$$

We seek θ^*

$$\theta^* = \arg\max_\theta \prod_i p_\theta(x_i), \qquad p_\theta(x) = \int p(x|z,\theta)p(z)dz,$$

How to match data to an unknown solution of the integral? Mix. of Dirac?

Bayes rule

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

ELBO for approximating distribution $q(x|z)$:

$\mathrm{KL}(q_\psi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q(z|x)}\left(\log q\psi(z|x) - \log p_\theta(x|z) - \log p(z)\right) + \log p_\theta(x)$

Key idea: $q_\psi(z|x)$ is flexible to approach $p_\theta(z|x)$ => (KL ≈ 0)

$$\theta^*, \psi^* = \arg\max_{\theta,\psi} \mathrm{E}_{q(z|x)}\left(-\log q_\psi(z|x) + \log p_\theta(x|z) + \log p(z)\right)$$

Autoencoding Variational Bayes

Key idea: $q(z|x,\psi)$ is flexible to approach $p(z|x)$ => (KL->0)

$$\begin{aligned}\theta^*,\psi^* &= \arg\max_{\theta,\psi} \mathrm{E}_{q(z|x)}\left(-\log q_\psi(z|x) + \log p_\theta(x|z) + \log p(z)\right)\\ &= \arg\max_{\theta,\psi} \mathrm{E}_{q(z|x)}\left(\log p_\theta(x|z)\right) - \mathrm{KL}(q_\psi(z|x)||p(z))\end{aligned}$$

- $f_\theta(z)$ is a NN with parameters θ,
- $q_\psi(z|x) = \mathcal{N}(\mu_\psi(x), \mathrm{diag}(\sigma^2_\psi(x)))$ where $\mu_\psi(x)$ and $\sigma_\psi(x)$ are NN.
  - $\mathrm{KL}(q_\psi(z|x)||p(z))$ has analytical form!

$$\mathrm{KL} = \frac{1}{2}\left\{\sigma^2_\psi(x) + \mu_\psi(x)^2 - 1 - 2\log\sigma_\psi(x)\right\}$$

  - log-likelihood

$$\log p_\theta(x|z) = -\frac{1}{2\sigma}(x - f_\theta(z))^2$$

Reparametrization trick

Minimization:

$$\theta^*, \psi^* = \arg\min_{\theta,\psi} \mathsf{E}_{q(z|x)} \left(\frac{1}{\sigma}(x - f_\theta(z))^2 \right) + \sigma_\psi^2(x) + \mu_\psi(x)^2 - 1 - 2\log\sigma_\psi(x),$$

we need to compute the expectation.

Reparametrization trick

Minimization:

$$\theta^*, \psi^* = \arg\min_{\theta,\psi} \mathrm{E}_{q(z|x)} \left(\frac{1}{\sigma}(x - f_\theta(z))^2 \right) + \sigma_\psi^2(x) + \mu_\psi(x)^2 - 1 - 2\log\sigma_\psi(x),$$

we need to compute the expectation.

Reparametrization trick:

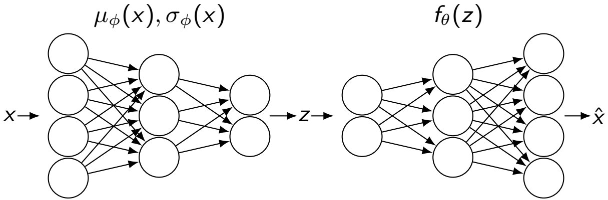
$$q_\psi(z|x) \approx \frac{1}{n}\sum_{i=1}^{n} \delta(z - z^{(i)}),$$

$$z_i = \mu_\psi(x) + \sigma_\psi(x) \circ e_i, \qquad e_i \sim \mathcal{N}(0,1)$$

Final cost (for NN training)

$$\theta^*, \psi^* = \arg\min_{\theta,\psi} \sum_{i=1}^{n} \left(\frac{1}{\sigma}(x_i - f_\theta(\mu_\psi(x_i) + \sigma_\psi(x_i) \circ e_i))^2 \right) + \sum_{i=1}^{n} \left(\sigma_\psi^2(x_i) + \mu_\psi(x_i)^2 - 1 - 2\log\sigma_\psi(x) \right),$$

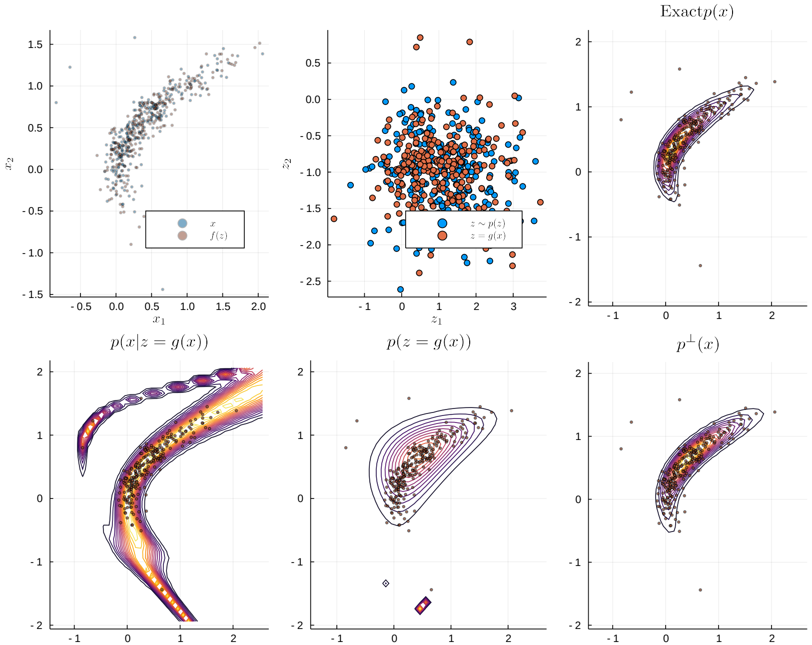
Variational Autoencoder



- Extension of classical Autoencoders
  - "just" another regularization of AE
- Useful for generation of artificial samples: $z = \text{randn}$
- Allows dimensionality reduction
- Special case: Probabilistic PCA Principal Component analysis,

$$x = Az + e, \qquad\qquad z \sim \mathcal{N}(0, I),$$

Examples of use



Drawbacks and extensions

1. Joint density estimation and dimensionality reduction

   2-stage VAE: Decomposition of 2 VAE: $x \to z \to x$ and $z \to u \to z$.

2. Transformation of Gaussian problematic for multivariate densities.

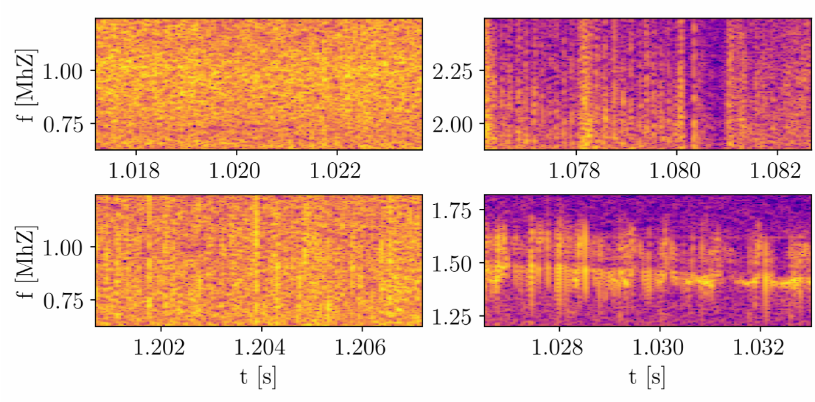
   Wasserstein AE: We may transform multi-modal prior $p(z)$. KL intractable. In practice, common approximation is

   $$\mathrm{MMD}(p, q) = ||\mathrm{E}_{x\sim p}(\varphi(x)) - \mathrm{E}_{y\sim q}(\varphi(y)))||_{\mathcal{H}}.$$

   works with any distribution, as long as we can sample from it.

3. Combinations: VAE-GAN loss, Stein, Relevance VAE,

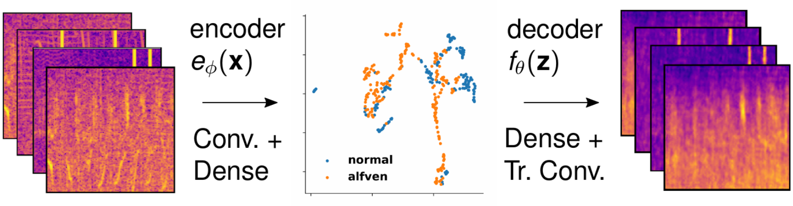
Detection of Alfven Eigenmodes



Data:

- 10^6 unlabeled spectrograms
- 400 labels

Detection of Alfven Eigenmodes



$\mathbf{x} \in \mathbb{R}^{128\times128\times1}$ $\mathbf{z} \in \mathbb{R}^d$ $\hat{\mathbf{x}} \in \mathbb{R}^{128\times128\times1}$

opt. criteria	classifier	AUC	prec@50
MSE	kNN	0.80±0.07	0.88±0.10
KLD	kNN	0.80±0.08	0.85±0.11
MMD	kNN	**0.91±0.06**	**0.94±0.05**
GAN	kNN	0.83±0.07	0.87±0.10
MMD + GAN	kNN	0.86 ± 0.07	0.91±0.10
MSE	GMM	0.75±0.06	0.80±0.10
KLD	GMM	0.74±0.06	0.83±0.11
MMD	GMM	0.66±0.12	0.72±0.12
GAN	GMM	0.74±0.06	0.82±0.11
MMD + GAN	GMM	0.76±0.06	0.84±0.10

Bayesian Neural Networks

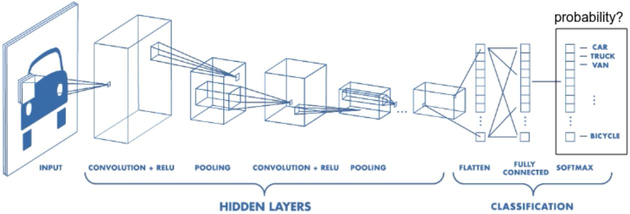
Consider regression problem

$$y = f_\theta(x) + e,$$

for known x and y. (If $y \in \{0, 1\}$ it is classification).

- Classical SGD learns $\hat{\theta}$
- Is it worrying?
  - not if we have enough "good" data,
  - not if data are i.i.d,
  - we need labels!
- Bayesian approach
  - makes use of all your data,
  - allows active learning

Deep classification: trivialized



Objective, cost function (cross-entropy):

$$\mathcal{L}(y,x) = -\sum_{i=1}^{n} y_i f(x_i,\theta) + (1-y_i)\log(1-f(x_i,\theta)),$$

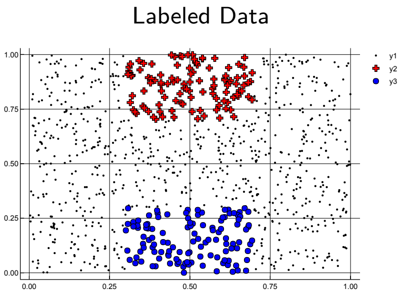
Training with (stochastic) gradient

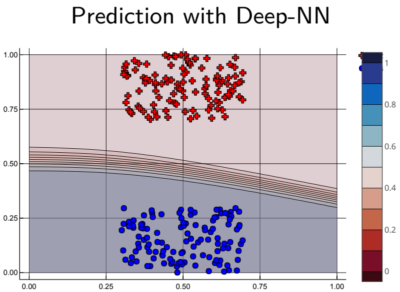
$$\hat{\theta}^{\text{new}} = \hat{\theta}^{\text{old}} - \eta\nabla_\theta \mathcal{L}(),$$

$$\nabla_\theta \mathcal{L}() = \nabla_\theta \sum_{i=1}^{n} l(y_i,x_i) \approx \sum_{j=1}^{J} \nabla_\theta l(y_j,x_j),$$

where j are i.i.d. samples from $\{1,\dots n\}$.

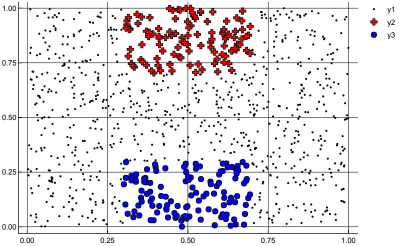
Deep classification: toy data



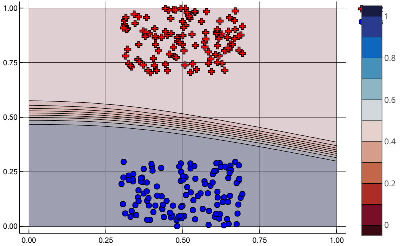


Deep classification: toy data

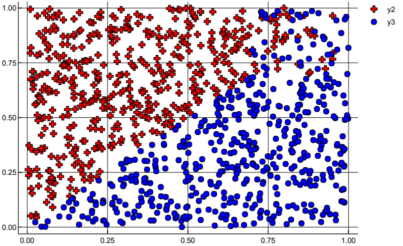
Labeled Data



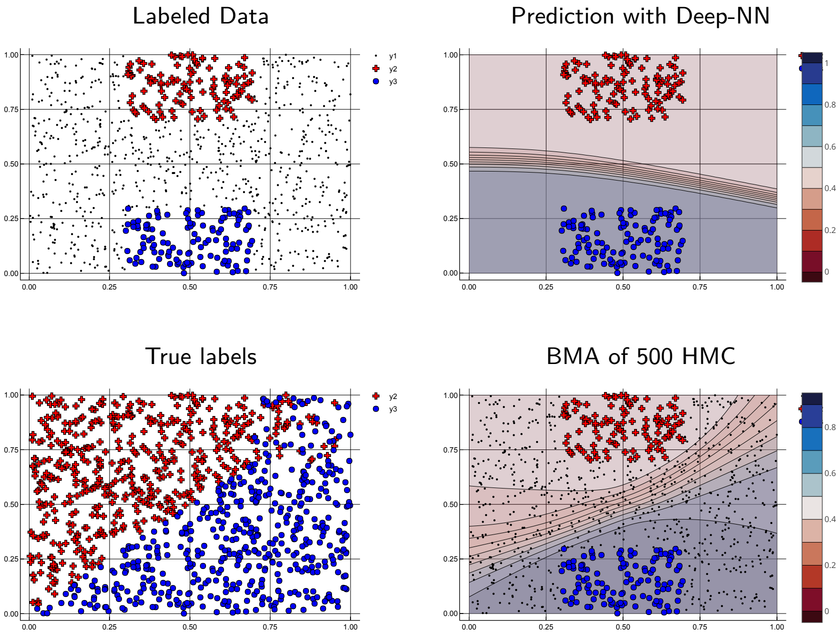
Prediction with Deep-NN



True labels



Deep classification: toy data



Active Learning

Knowledge: data seen so far $X^{\text{seen}}, y^{\text{seen}}$

Unknown: labels y^{new} at points X^{new}, network weights

Decision: select "most-interesting" point x^* in X^{new} for an evaluator (oracle) to obtain y^*.

Active Learning

Knowledge: data seen so far $X^{\text{seen}}, y^{\text{seen}}$

Unknown: labels y^{new} at points X^{new}, network weights

Decision: select "most-interesting" point x^* in X^{new} for an evaluator (oracle) to obtain y^*.

Expected utility:

$$x^* = \arg\max_{x \in X^{\text{new}}} E\left\{U(x,y) | X^{\text{seen}}, y^{\text{seen}}\right\}$$
$$= \arg\max_{x \in X^{\text{new}}} \sum_{j=1}^{0} U(x,y) p(y=j | X^{\text{seen}}, y^{\text{seen}})$$

where U is utility function and $p(y=j|\cdot)$ is a posterior *predictive* probability.

Active Learning

Knowledge: data seen so far $X^{\text{seen}}, y^{\text{seen}}$

Unknown: labels y^{new} at points X^{new}, network weights

Decision: select "most-interesting" point x^* in X^{new} for an evaluator (oracle) to obtain y^*.

Expected utility:

$$
\begin{aligned}
x^* &= \arg\max_{x \in X^{\text{new}}} E\left\{U(x,y) | X^{\text{seen}}, y^{\text{seen}}\right\} \\
&= \arg\max_{x \in X^{\text{new}}} \sum_{j=1}^{0} U(x,y) p(y=j | X^{\text{seen}}, y^{\text{seen}})
\end{aligned}
$$

where U is utility function and $p(y=j|\cdot)$ is a posterior *predictive* probability.

Utility: to learn about the true problem as much as possible

Expected Utility as Acquisition function

Utility: to learn about the true problem as much as possible

- mutual information between $D^{\text{new}} = \{y^{\text{new}}, x^{\text{new}}\}$ and $D = \{X^{\text{seen}}, y^{\text{seen}}\}$

$$I(D^{\text{new}}, D) = \mathrm{KL}\left(p(D, D^{\text{new}}) || p(D)p(D^{\text{new}})\right).$$

Expected Utility as Acquisition function

Utility: to learn about the true problem as much as possible

- mutual information between $D^{\text{new}} = \{y^{\text{new}}, x^{\text{new}}\}$ and $D = \{X^{\text{seen}}, y^{\text{seen}}\}$

$$I(D^{\text{new}}, D) = \text{KL}\left(p(D, D^{\text{new}}) || p(D)p(D^{\text{new}})\right).$$

- proxy:
  - variance $U(x, y) = (y(x) - \overline{y}(x))$,
  - entropy $U(x, y) = \log(y(x))$

Acquisition function:

$$x^* = \arg\max_{x \in X^{\text{new}}} a(x, y)$$

$$a(x, y) = \sum_{j=1}^{0} U(x, y)p(y = j | X^{\text{seen}}, y^{\text{seen}})$$

Benchmark: Active Learning

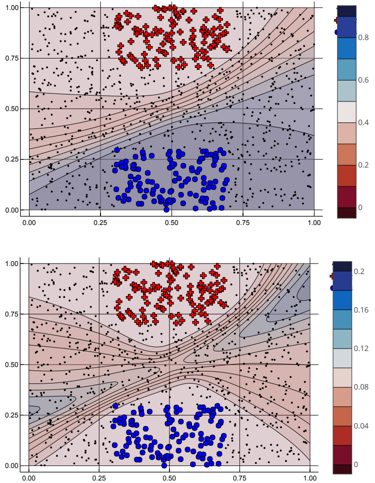
1. Fit parametric model including hyperparameters

$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.



Benchmark: Active Learning

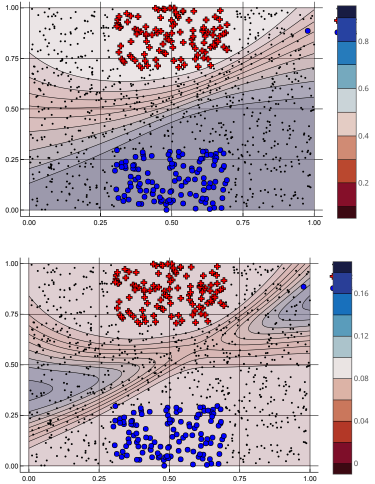
1. Fit parametric model including hyperparameters

$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.



Benchmark: Active Learning

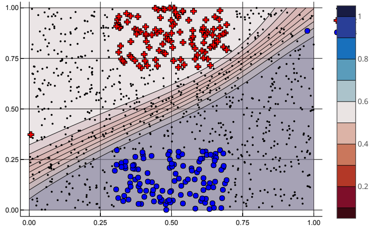
1. Fit parametric model including hyperparameters

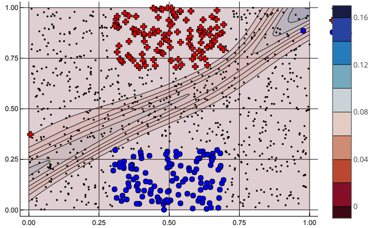
$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.





Benchmark: Active Learning

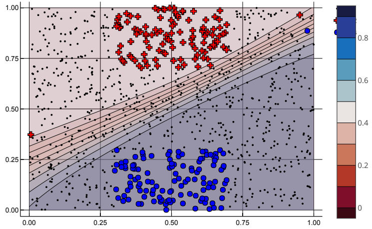
1. Fit parametric model including hyperparameters

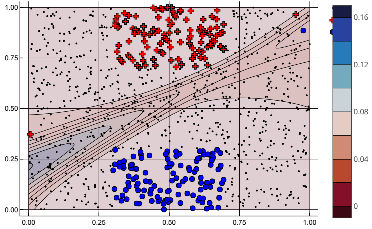
$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.





Benchmark: Active Learning

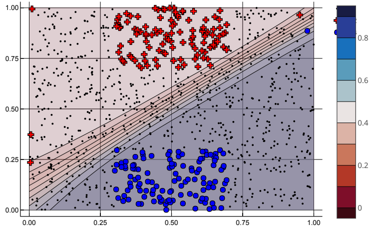
1. Fit parametric model including hyperparameters

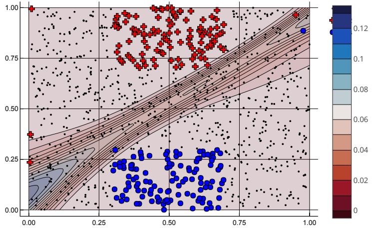
$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.





Benchmark: Active Learning

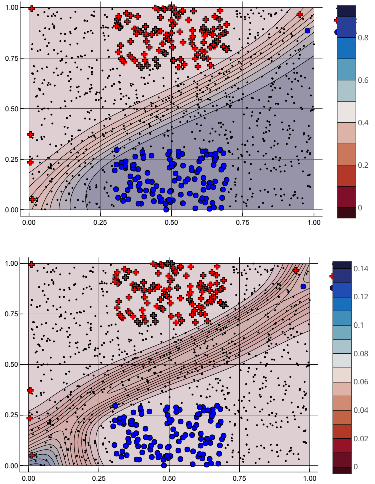
1. Fit parametric model including hyperparameters

$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.



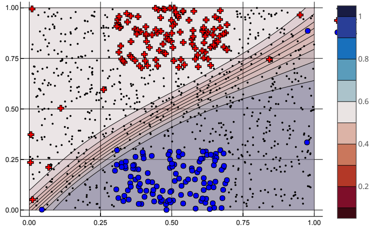
1. Fit parametric model including hyperparameters

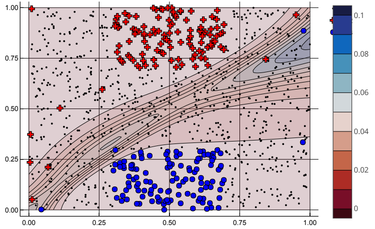
$$p(y|X^{\text{known}}, y^{\text{known}})$$

2. Find point with maximum probability of improvement,

$$x^* = \arg\max_{x \in X} E\left(y^2 - E(y^2)\right)$$

3. Evaluate y^* and add x^* to X^{known} and y^* to y^{known}
4. GOTO 1.





Probability is crucial!

HMC [Neal, 1993]: golden standard. Beautiful. Expensive!

SGLD [Welling, Teh, 2011]: extension of SGD

$$\hat{\theta}^{\text{new}} = \hat{\theta}^{\text{old}} - \eta \nabla_\theta \mathcal{L}() + \sqrt{\eta} e, \;\; e \sim \mathcal{N}(0, 1),$$

is a valid MCMC algorithm with Langevin kernel.

- Acceptance rate->1 with decreasing η.
- Hard to tune.

SGD is Baysian [Mandt et.al., 2017]: SGD is discretization of stochastic Ornstein-Uhlenbeck process:

$$d\theta(t) = -\epsilon g(\theta) dt + \frac{\epsilon}{\sqrt{S}} B dW(t)$$

Using properties of OU, calibration of SGD is, $C = B^\top B$,

$$\hat{\theta}^{\text{new}} = \hat{\theta}^{\text{old}} - H \nabla_\theta \mathcal{L}(),$$

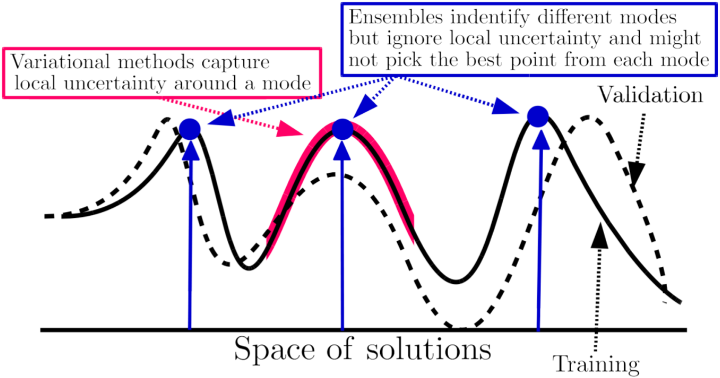
$$H \approx \frac{S}{N} C,$$

$$C_t = \rho C_{t-1} + (1 - \rho) \text{var}(\nabla_\theta \mathcal{L}())$$

Many others: MC dropout...

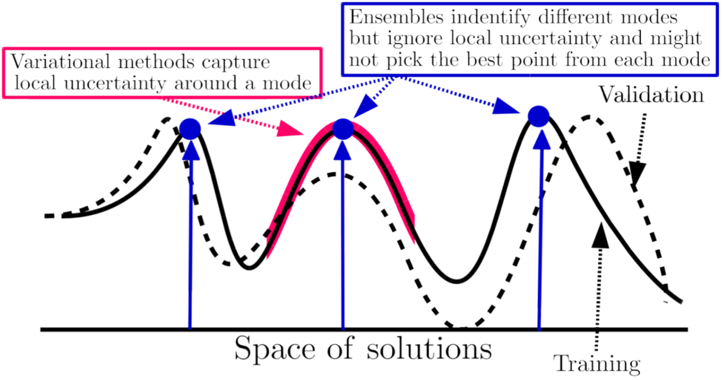
How well Deep Bayes methods work?

- Not as good as we would like. Ensembles are often better.
- Loss of Landscape [Fort et. al., 2019]



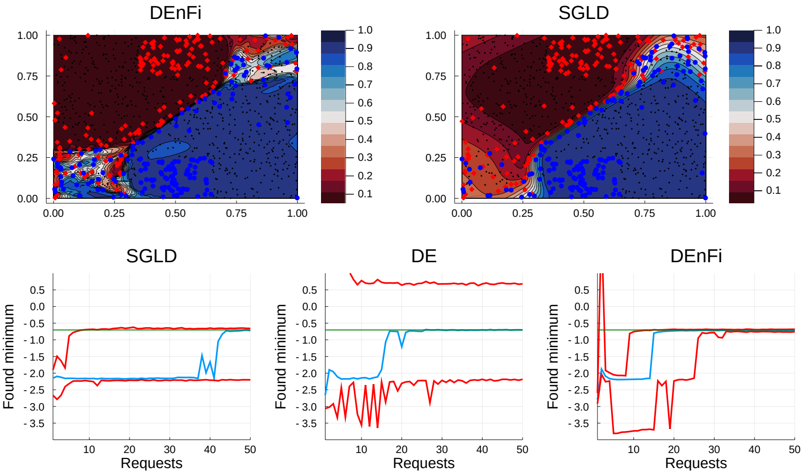
How well Deep Bayes methods work?

- Not as good as we would like. Ensembles are often better.
- Loss of Landscape [Fort et. al., 2019]

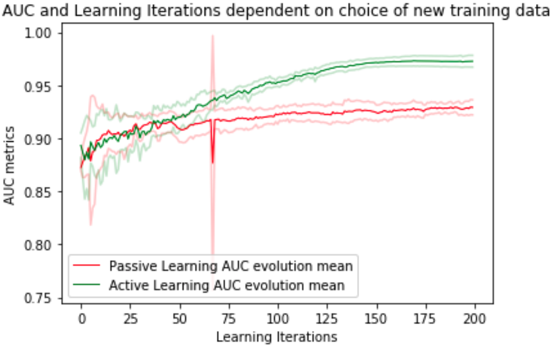


Deep Ensemble Filter [Ulrych, Smidl, 2020?] deep ensemble with inflation and localization steps.

DEnFi: Active Learning



Real-world data: Active text classification



Take home message

- Bayesian methods are useful when data are not complete or not i.i.d.
  - few data samples,
  - active learning
  - robust decision are required
- This happens in deep learning
- Plenty of work to be done
  - approximate inference
  - conjecture: redundancy in deep learning can be exploited to obtain Bayesian inference