딥러닝 에스프레소

딥러닝을 위한 베이지안 통계 Day1

Introduction, Probability Theory

2022

멀티캠퍼스

박수철

학습 목표

학습 목표

- Bayesian model을 통해 결정에 대한 불확실성(uncertainty)를 구할 수 있다.
- Bayesian model의 핵심이 되는 prior, likelihood, posterior의 개념을 이해할 수 있다.
- Bayesian model를 deep learning에 적용하기 위해서 필요한 variational inference의 개념을 이해할 수 있다.
- Bayesian을 접목한 deep learning model들을 이해하고 활용할 수 있다.

다루는 내용

다루는 내용

- Bayesian Linear Regression

  Linear Regression의 Bayesian 버전입니다. Bayesian이라고 하면 model의 parameter를 고정된 값이 아닌 random variable로 여기는 것입니다. 이를 통해 predictive distribution을 얻고 예측에 대한 불확실성(uncertainty)를 표현할 수 있습니다.

- GMM (Gaussian Mixture Models)

  데이터를 여러 Gaussian distribution들로 clustering 합니다. 데이터 마다 주어진 cluster component를 latent variable이라 여기면 visible data로부터 latent data를 inference하는 문제로 생각할 수 있습니다. Gaussian mixture로 주어진 모델이 데이터를 잘 표현하도록 marginal likelihood를 최대화하는 방식으로 트레이닝이 이루어지며 이를 위해 EM 알고리즘이 등장합니다.

- Probabilistic PCA

  Continuous latent space 상의 prior에서 뽑은 샘플이 data space로 linear transform을 통해 맵핑되고 이를 mean으로 구성된 Gaussian distribution이 data distribution을 나타내도록 모델링하는 generative model입니다. GMM과 같이 marginal likelihood를 최대화하는 방식으로 트레이닝 되고, close form이나 EM 알고리즘을 통해 solution을 얻을 수 있습니다.

- Auto-Encoders

  Deep Learning의 대표적인 모델로 dimension reduction, feature extraction을 수행합니다. Encoder, Decoder 구조를 통해 데이터를 압축하고 다시 복원합니다. 이 과정에서 얻은 latent variable을 데이터의 중요한 feature가 담긴 정보로 여길 수 있습니다.

다루는 내용

- VAE (Variational Auto-Encoders)

  Deep learning에서 generative model로써 중요한 위치를 차지하고 있는 모델이며 control 가능한 latent variable을 얻는데 목적이 있습니다. Auto-Encoders와 같이 Encoder, Decoder 구조를 가지고 있으며, 이에 더해 latent variable의 분포를 Gaussian 등으로 제약시키는 특징을 가지고 있습니다. GMM, Probabilistic PCA와 같이 marginal likelihood를 최대화 하는 것을 목적으로 하지만 계산의 intractability 때문에 variational inference가 적용되고 ELBO를 최대화하는 것으로 대체됩니다.

- BBB (Bayes by Backprop.)

  Bayesian을 deep learning에 접목시켜 regression과 classification 문제를 해결합니다. VAE가 latent variable의 posterior를 얻기 위해 variational inference를 사용하였다면 BBB에서는 모델의 weight의 posterior를 얻기 위해 variational inference를 사용합니다. Reparameterization trick 등 VAE와 기본적으로 접근 방식이 비슷합니다.

- MC Dropout (Dropout as a Bayesian Approximation)

  Neural networks에 dropout을 포함시켜 Bayesian neural notworks처럼 작동하도록 합니다. Weight의 posterior를 구하기 위해 variational inference를 사용하는 번거로움을 해결할 수 있습니다. Bayesian neural networks와 마찬가지로 결과에 대한 uncertainty를 측정할 수 있습니다.

- GAN (Generative Adversarial Networks)

  VAE와 함께 deep learning에서 대표적인 generative models 중 하나입니다. Generator와 discriminator를 서로 경쟁 관계에 놓아 좋은 결과를 얻을 수 있도록 트레이닝 합니다. VAE와 달리 매우 선명한 결과를 생성하는 것이 특징입니다. Bayesian 모델은 아니지만 data distribution과 model distributuion, KL-divergence 등의 개념이 적용됩니다.

다루는 내용

	Supervised		Unsupervised	
	Regression	Classification	Clustering	Dimension Reduction
Machine Learning	✓Linear Regression ✓Bayesian Linear Regression	Logistic Regression Bayesian Logistic Regression	✓K-means ✓GMM Bayesian GMM	✓PCA ✓Probabilistic PCA Bayesian PCA
Deep Learning	✓Non-Linear Regression by Neural Networks ✓Bayesian Regression by Neural Networks	✓Non-Linear Classification By Neural Networks ✓Bayesian Classification by Neural Networks	VQ-VAE	✓VAE

선행 지식

선행 지식

- Deep learning 기초

  CNN, RNN, Fully-connected Layers

  Backpropagation

  Stochastic Gradient Descent

- Deep learning frameworks

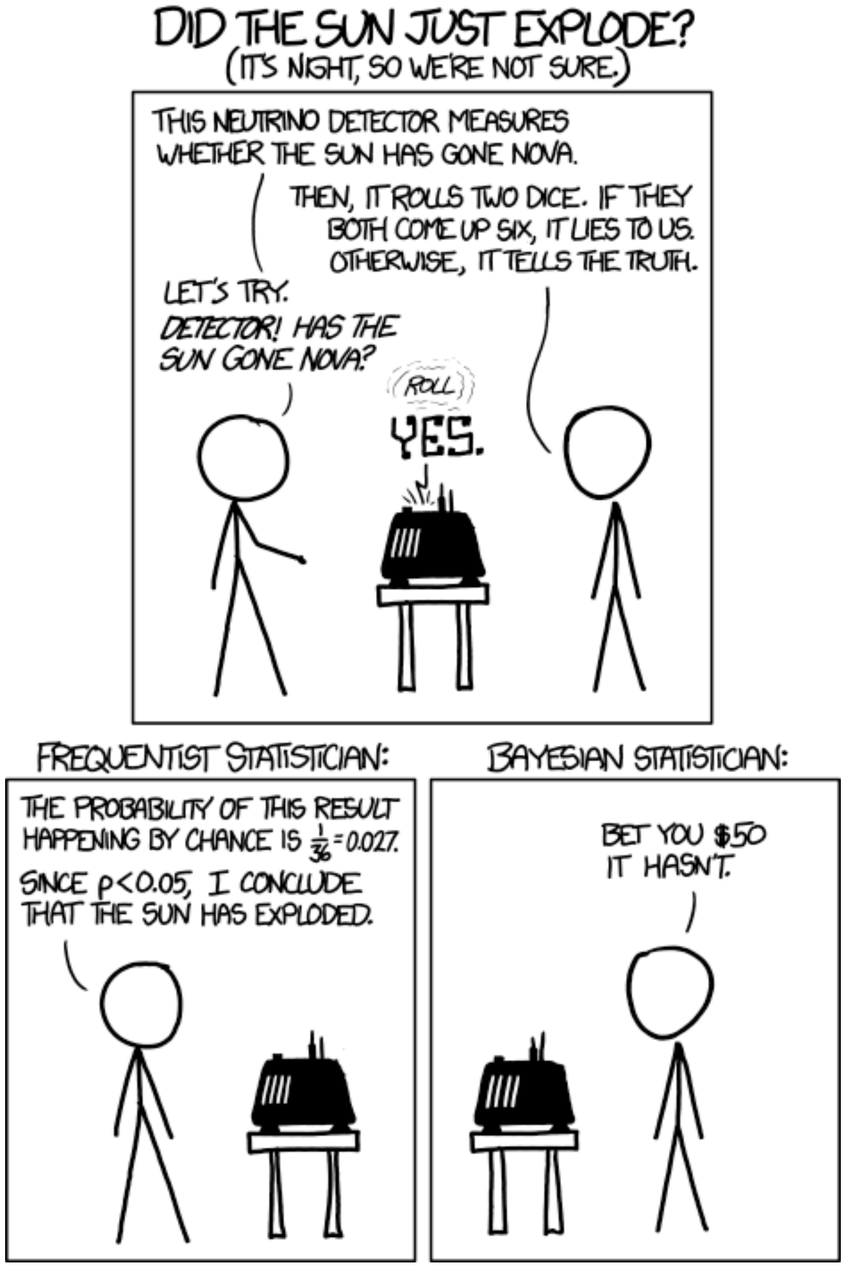
  Tensorflow, Pytorch, Keras

- 기초 수학 과목

  Calculus, Linear Algebra, Probability and Statistics

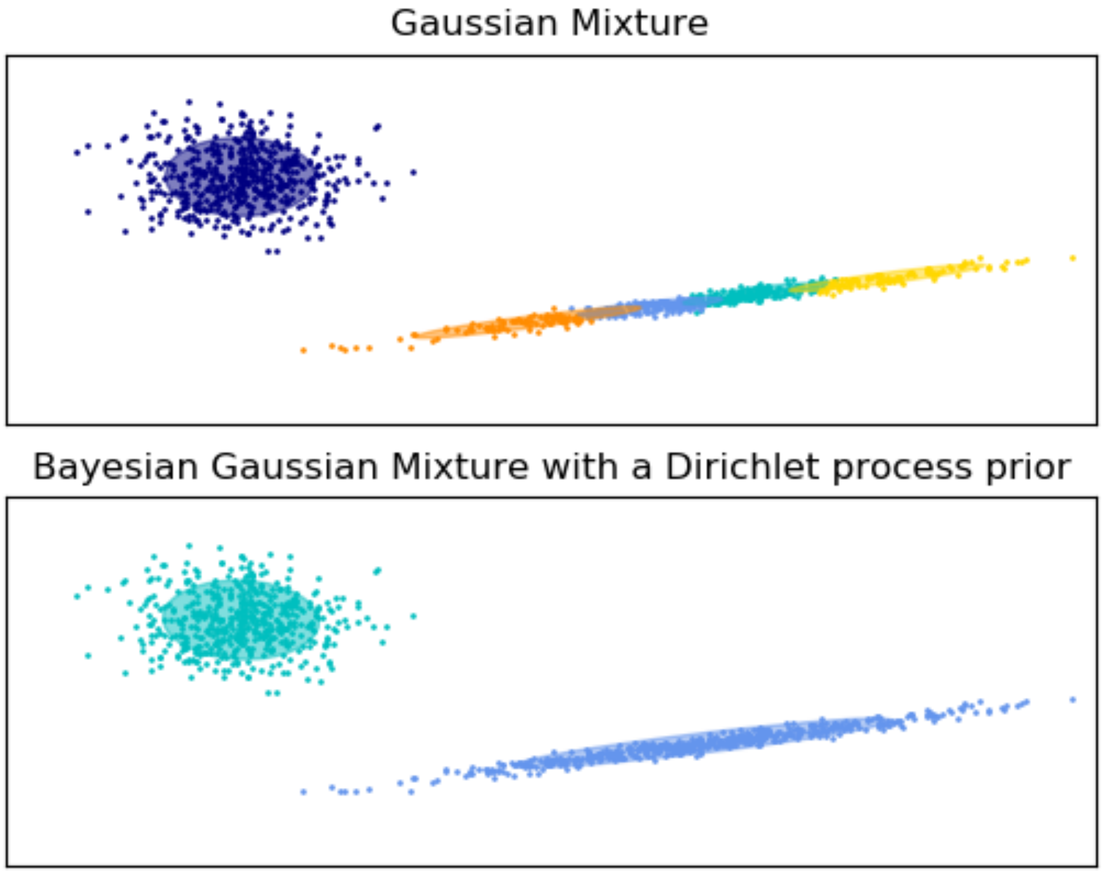
Prior

Prior



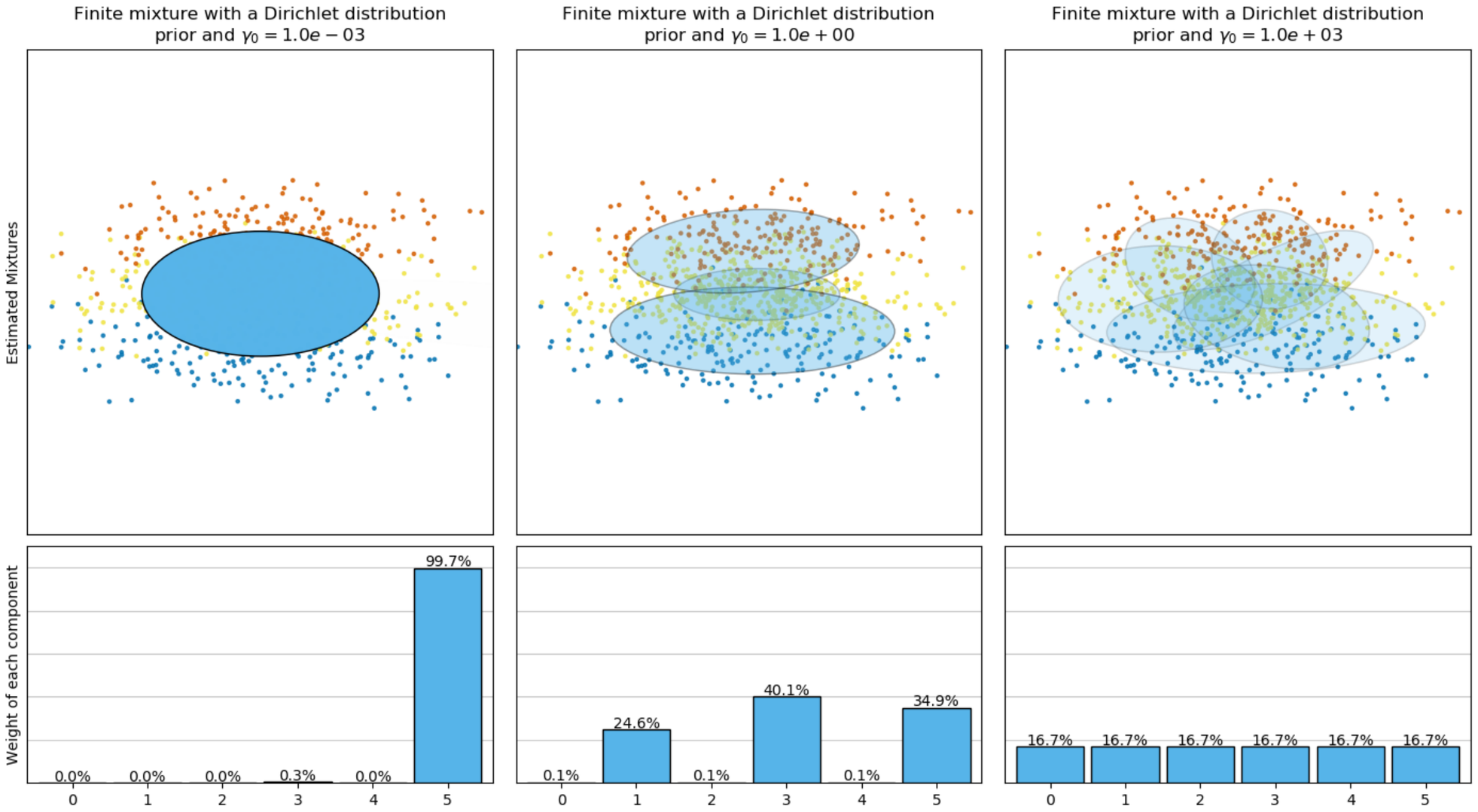
https://www.explainxkcd.com/wiki/index.php/1132:_Frequentists_vs._Bayesians

Prior GMM vs. Variational GMM



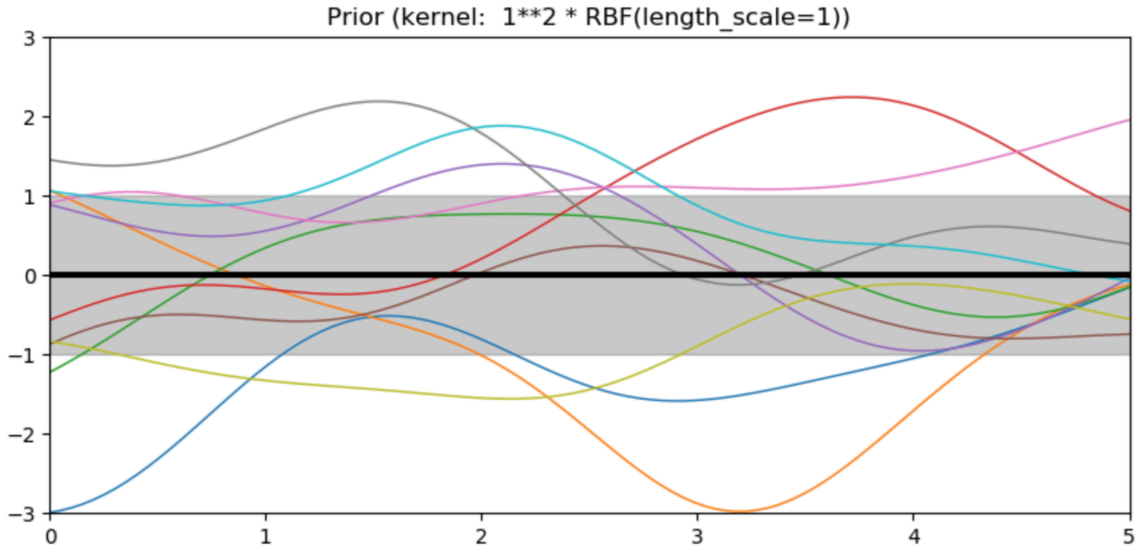
https://scikit-learn.org/stable/modules/mixture.html

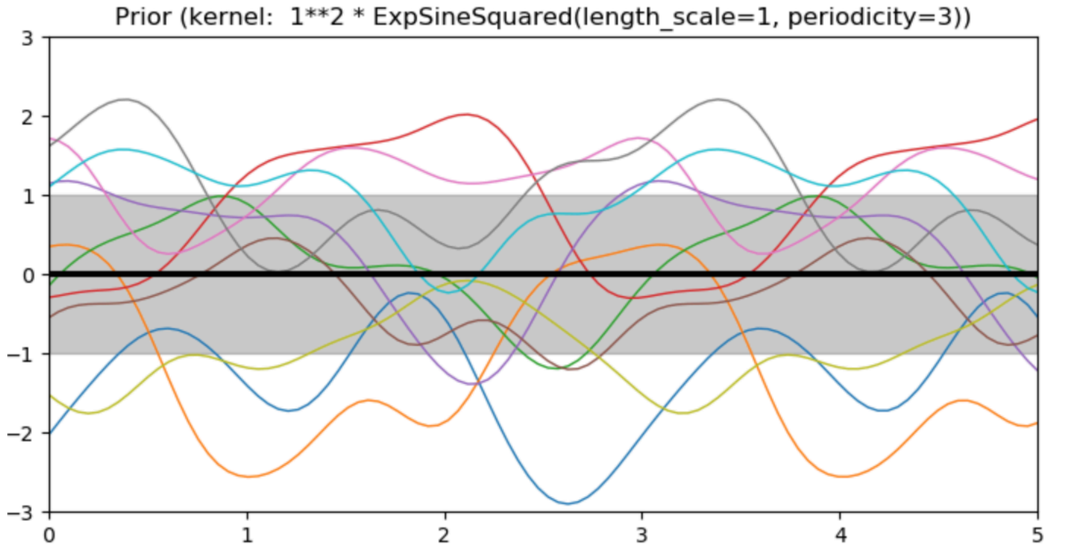
Prior GMM vs. Variational GMM

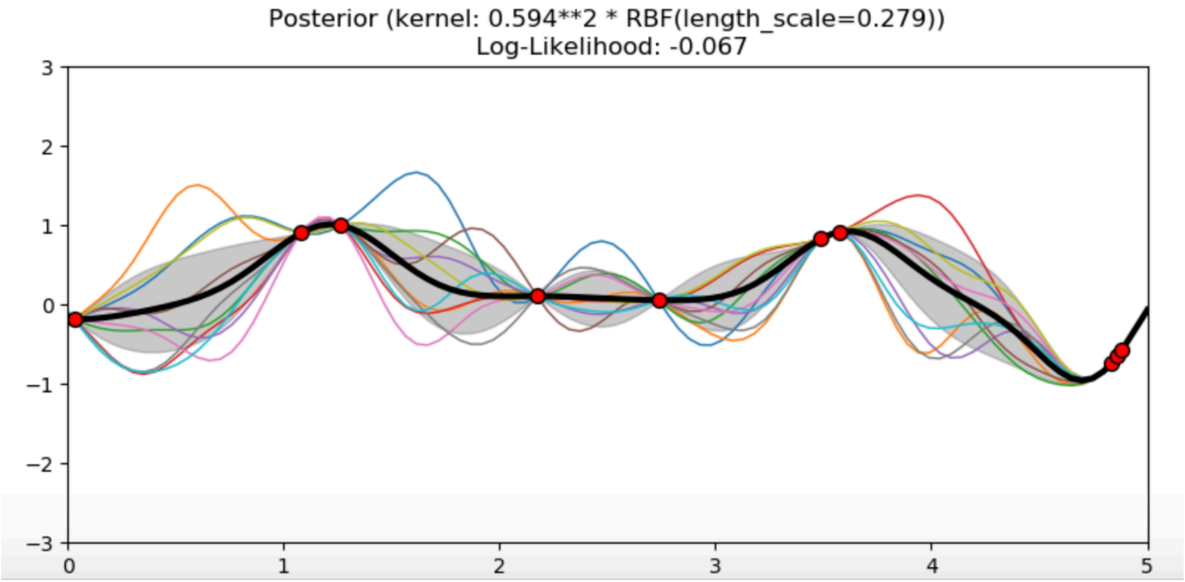


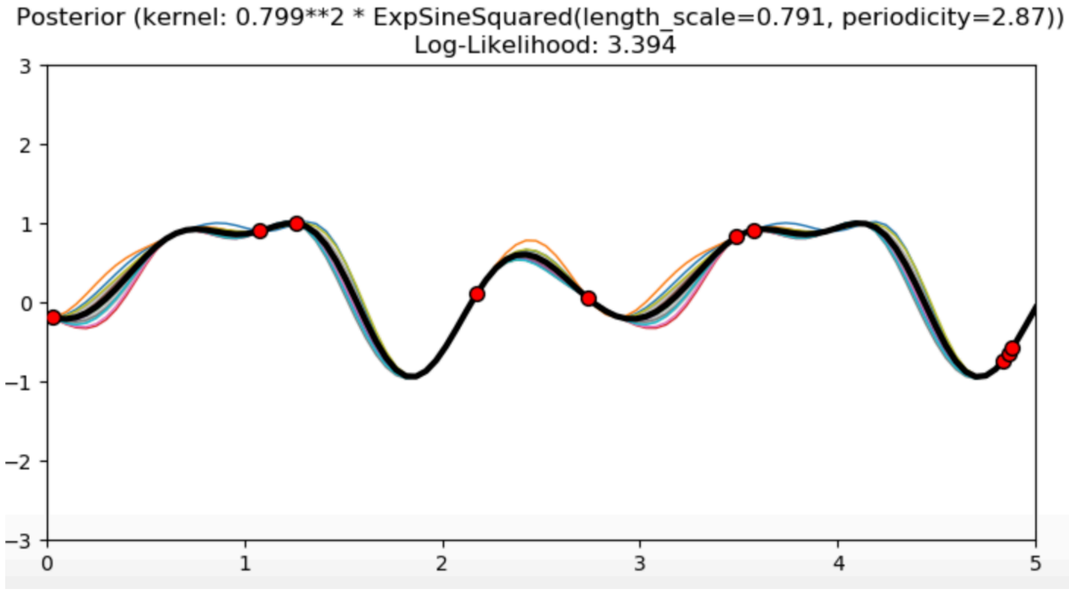
https://scikit-learn.org/stable/auto_examples/mixture/plot_concentration_prior.html

Prior Gaussian Process



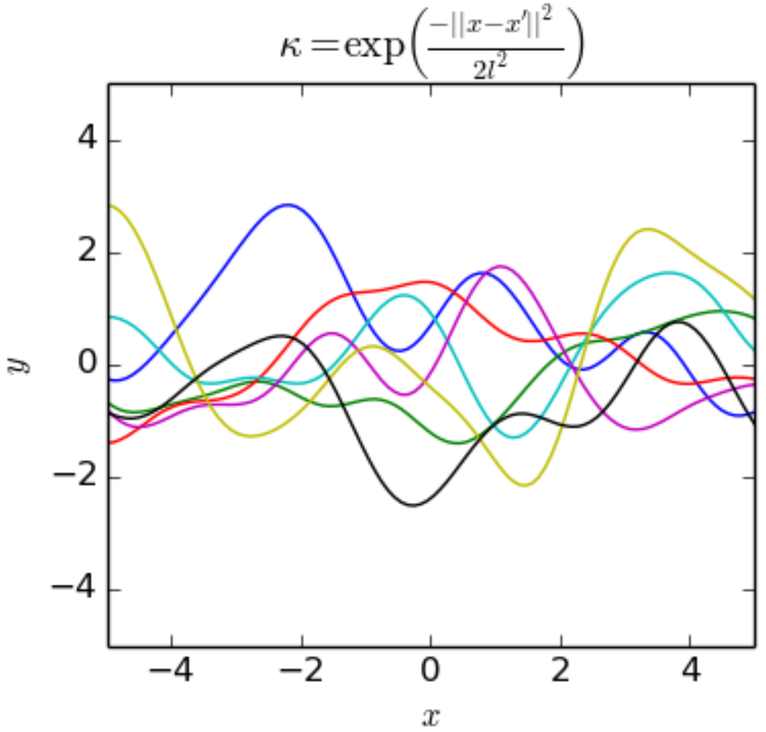


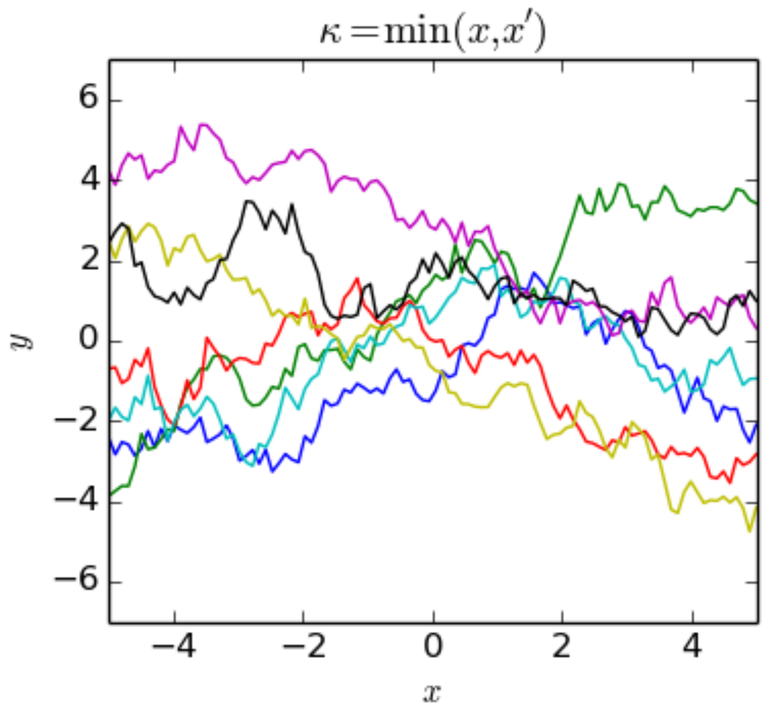


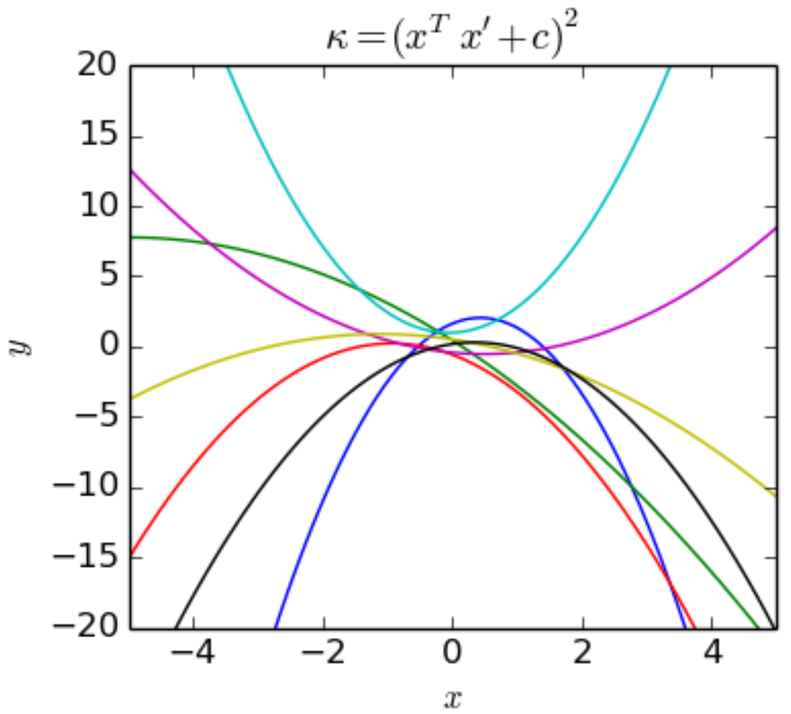


https://scikit-learn.org/stable/modules/gaussian_process.html

Prior Gaussian Process

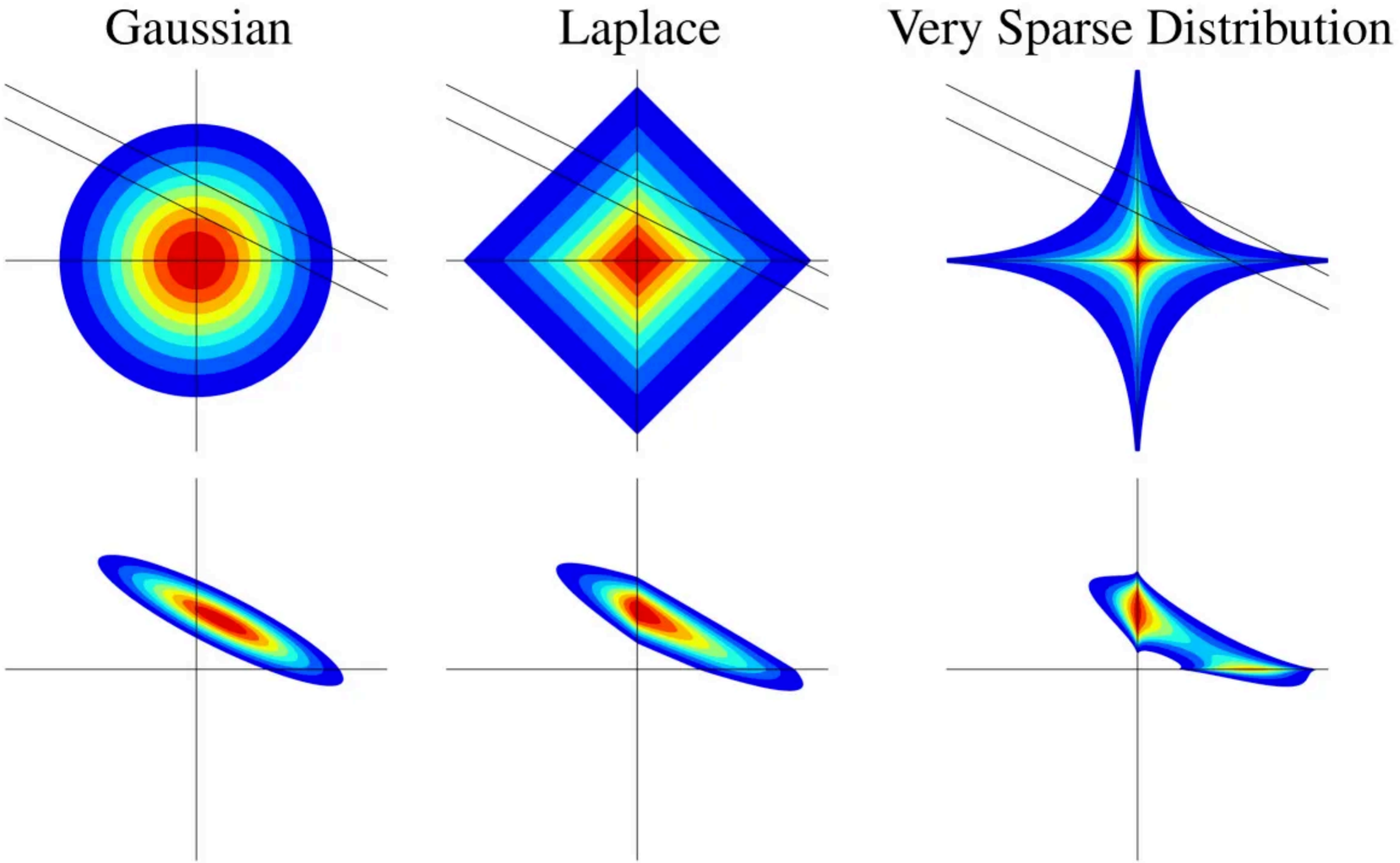






https://en.wikipedia.org/wiki/Gaussian_process#/media/File:Gaussian_process_draws_from_prior_distribution.png

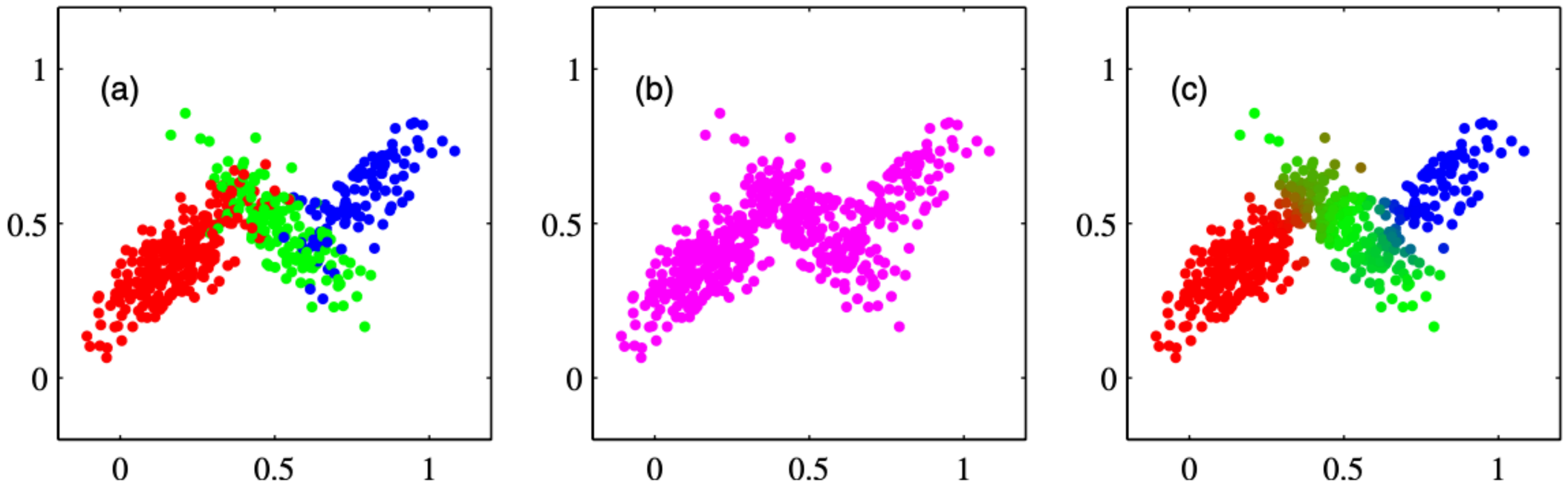
Prior Gaussian, Laplace, Sparse Distribution



https://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-1-51#Fig1

Hidden Variables

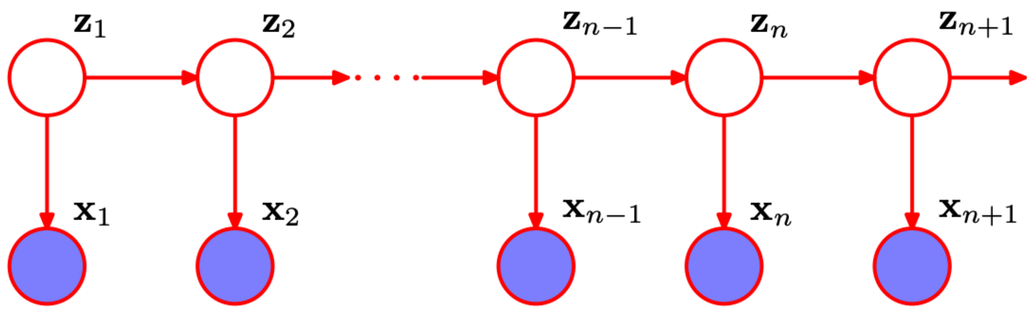
Hidden Variables Gaussian Mixture Models



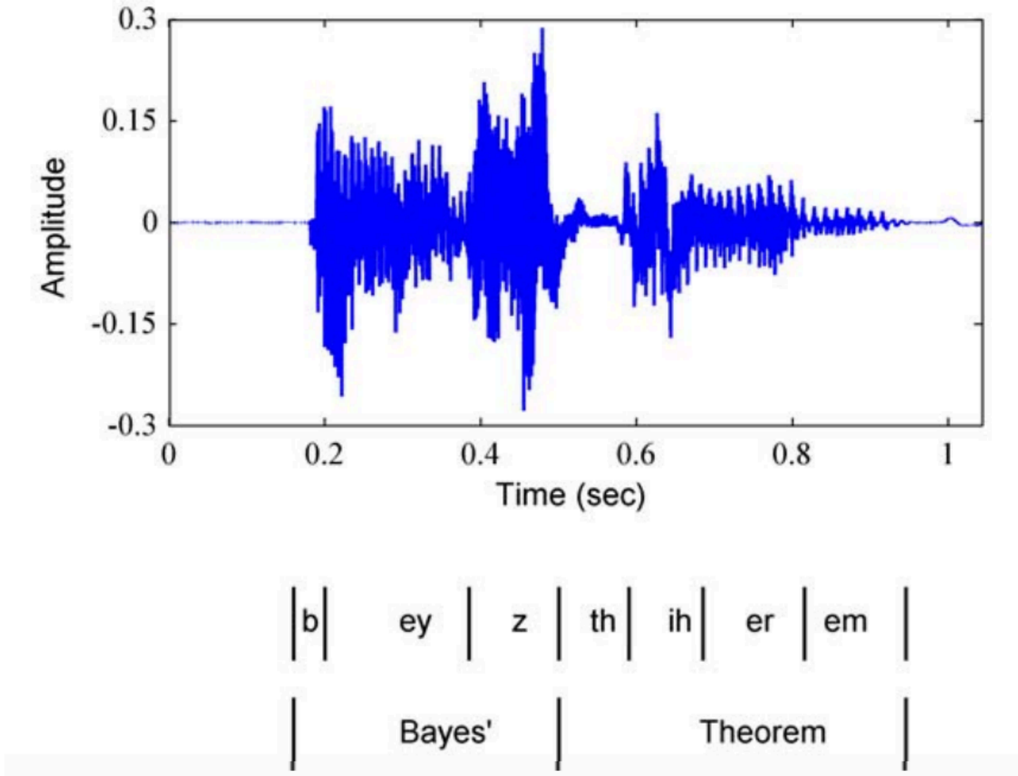
Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433

Hidden Variables Hidden Markov Chains

Directed graphs representation of the HMM



Speech recognition using the HMM



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.606, 609

Hidden Variables Variational Auto-Encoders

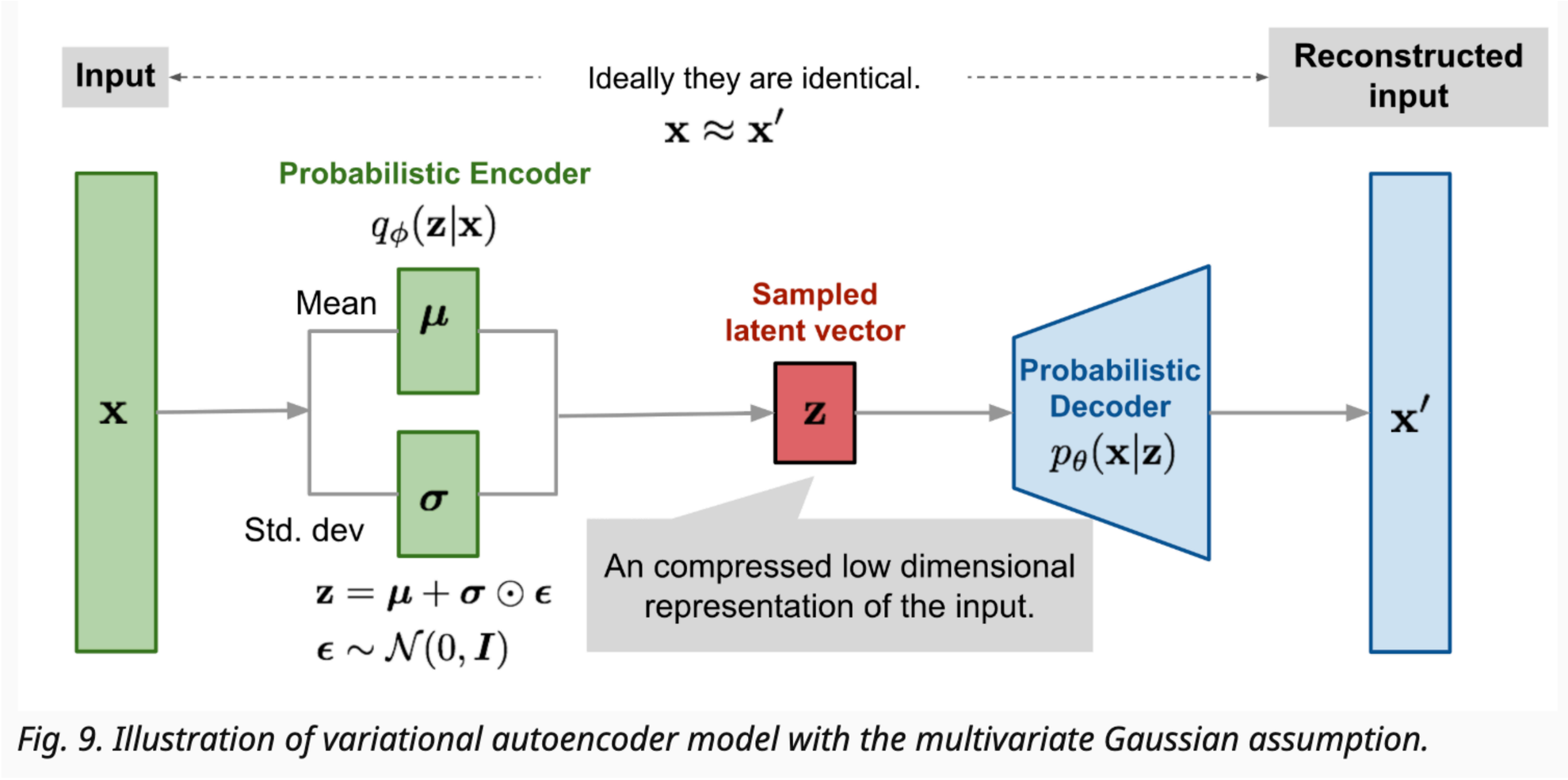
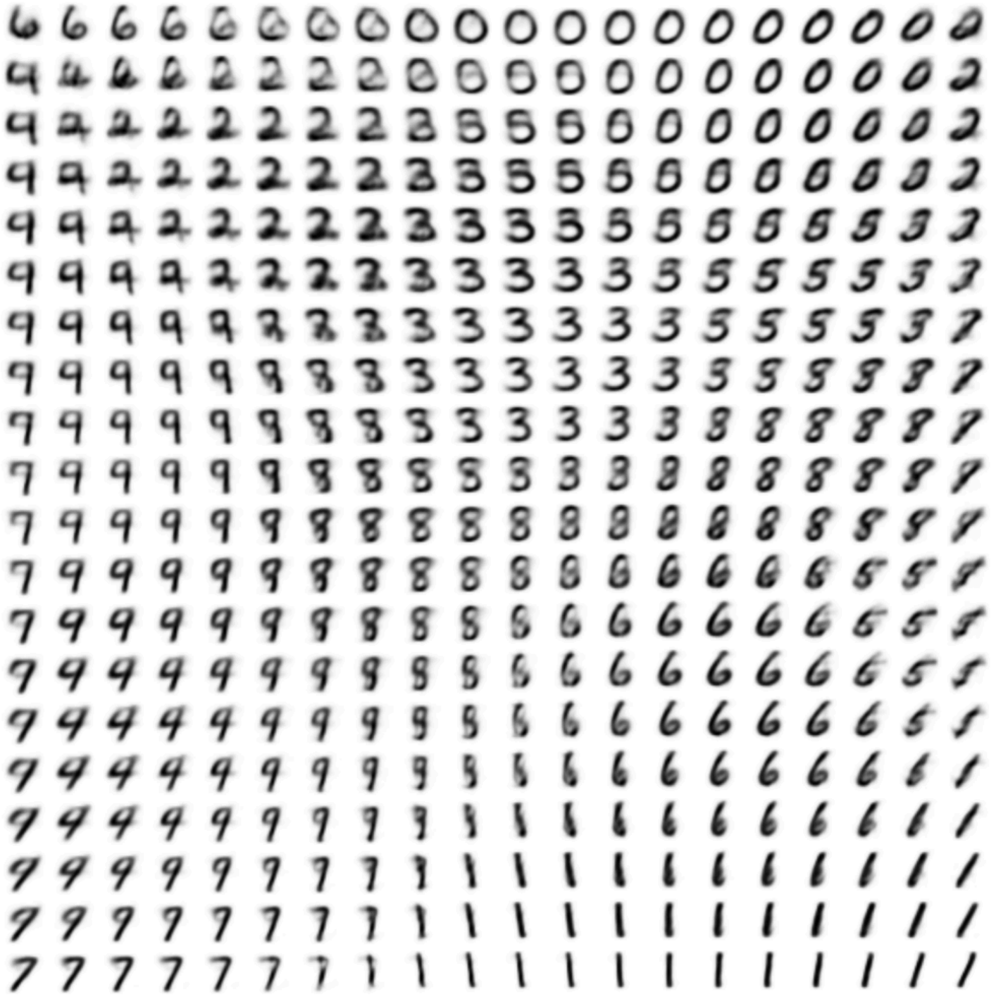


Fig. 9. Illustration of variational autoencoder model with the multivariate Gaussian assumption.

https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Hidden Variables Variational Auto-Encoders

(a) Learned Frey Face manifold



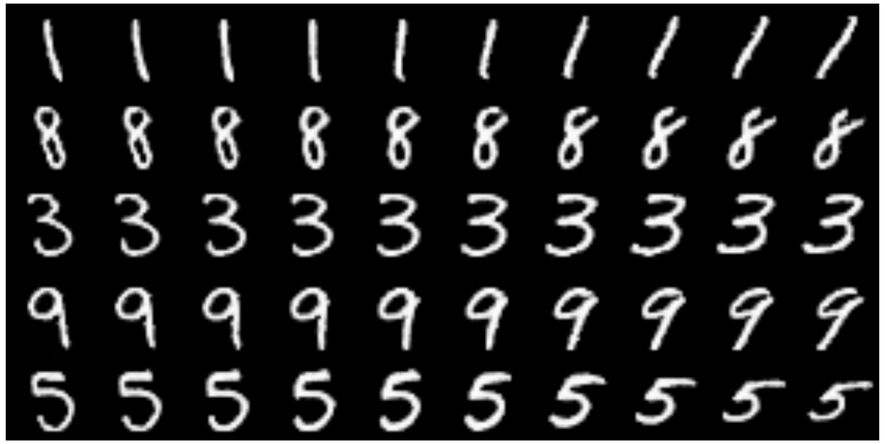
(b) Learned MNIST manifold

Diederik P. Kinema, Max Welling. Auto-Encoder Variational Bayes.

Hidden Variables InfoGAN

(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Hidden Variables InfoGAN

(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Hidden Variables InfoGAN

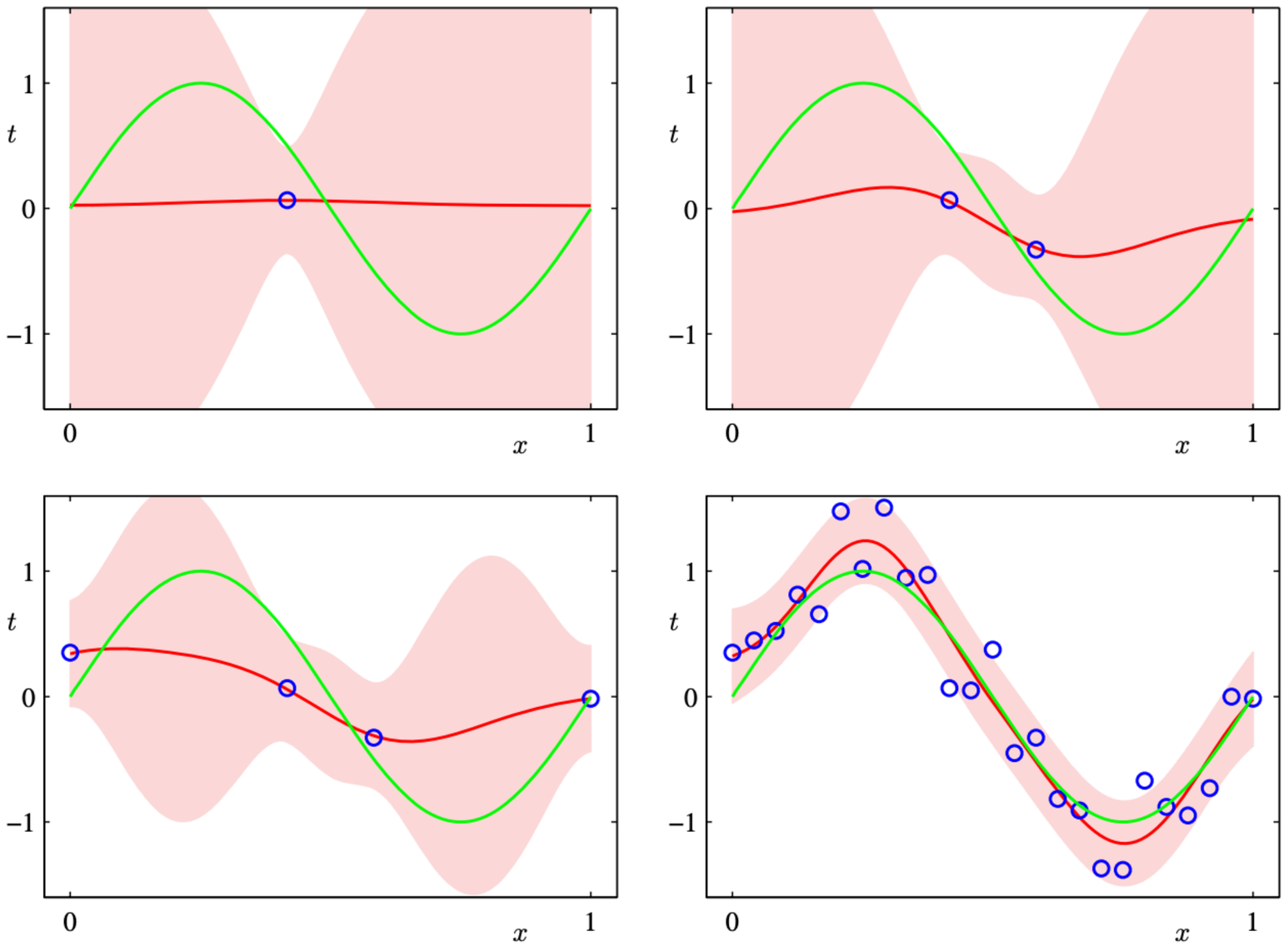
(a) Rotation

(b) Width

X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

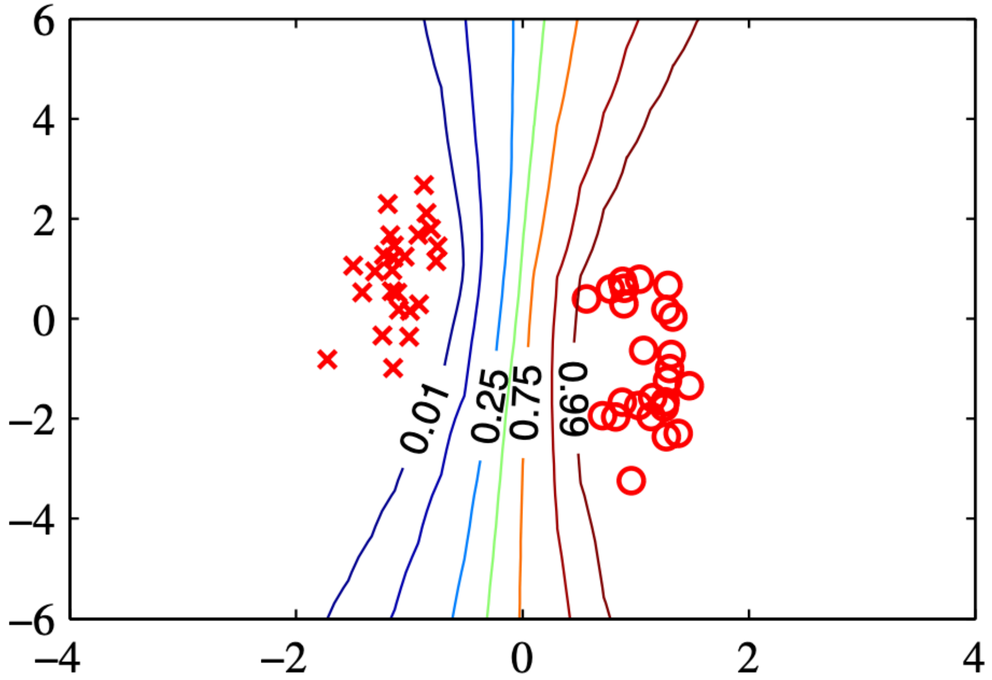
Predictive Distribution

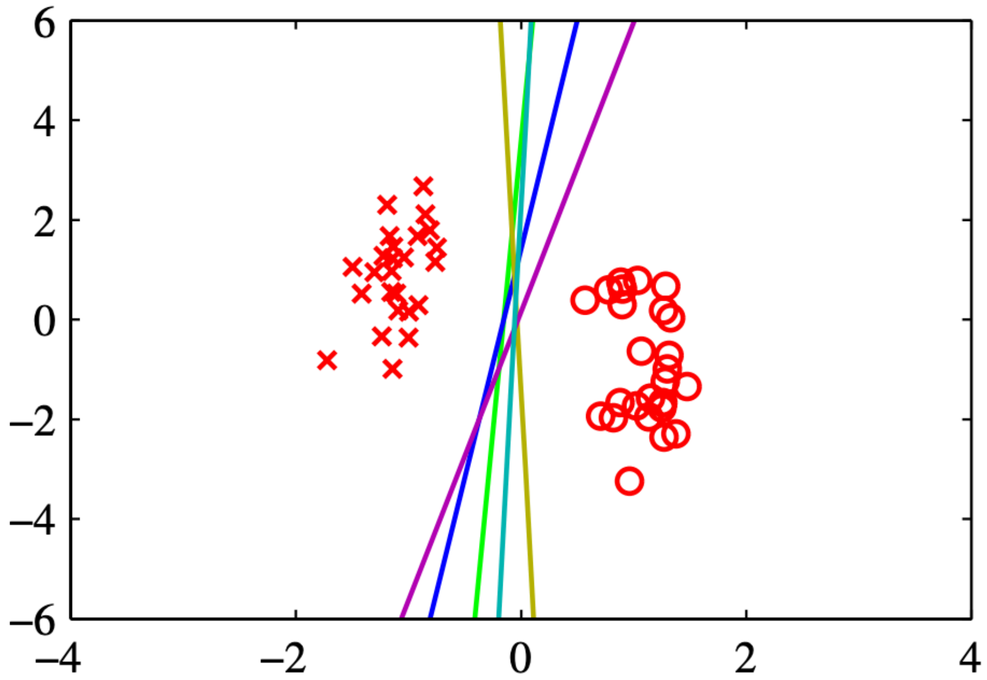
Predictive Distribution Bayesian Linear Regression



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.157

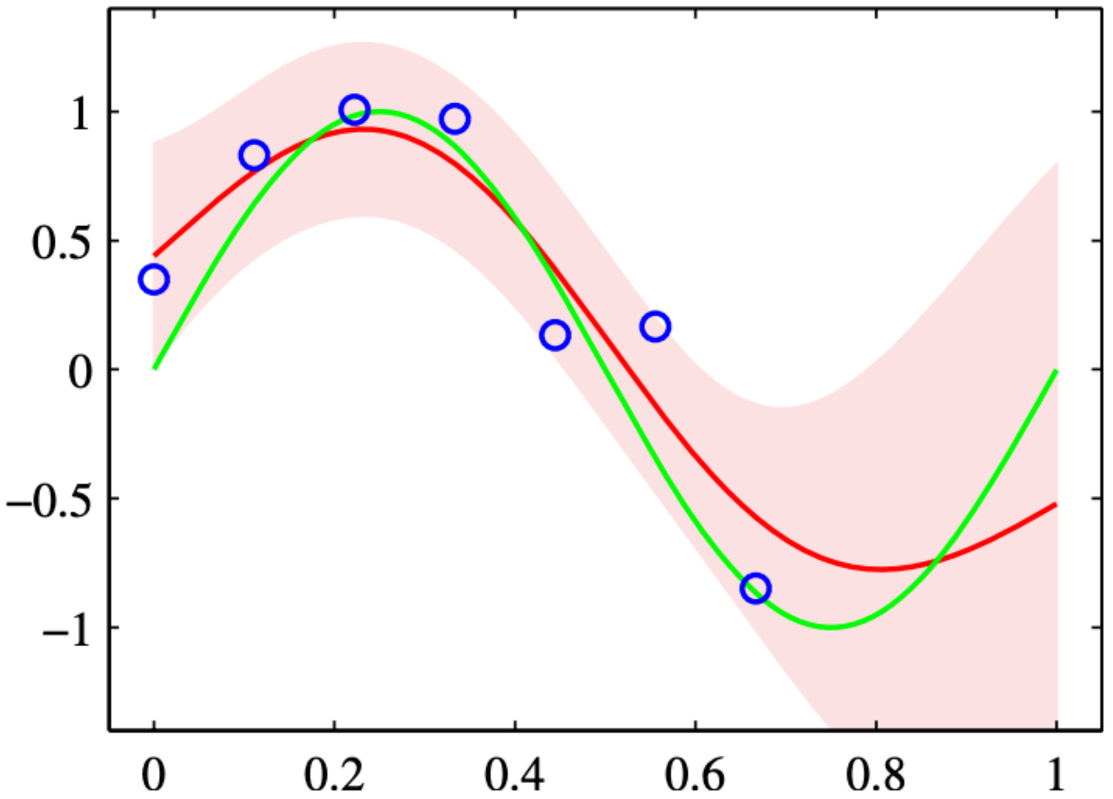
Predictive Distribution Bayesian Logistic Regression





Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.502

Predictive Distribution Gaussian Process



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.310

Predictive Distribution Bayes By Backdrop.

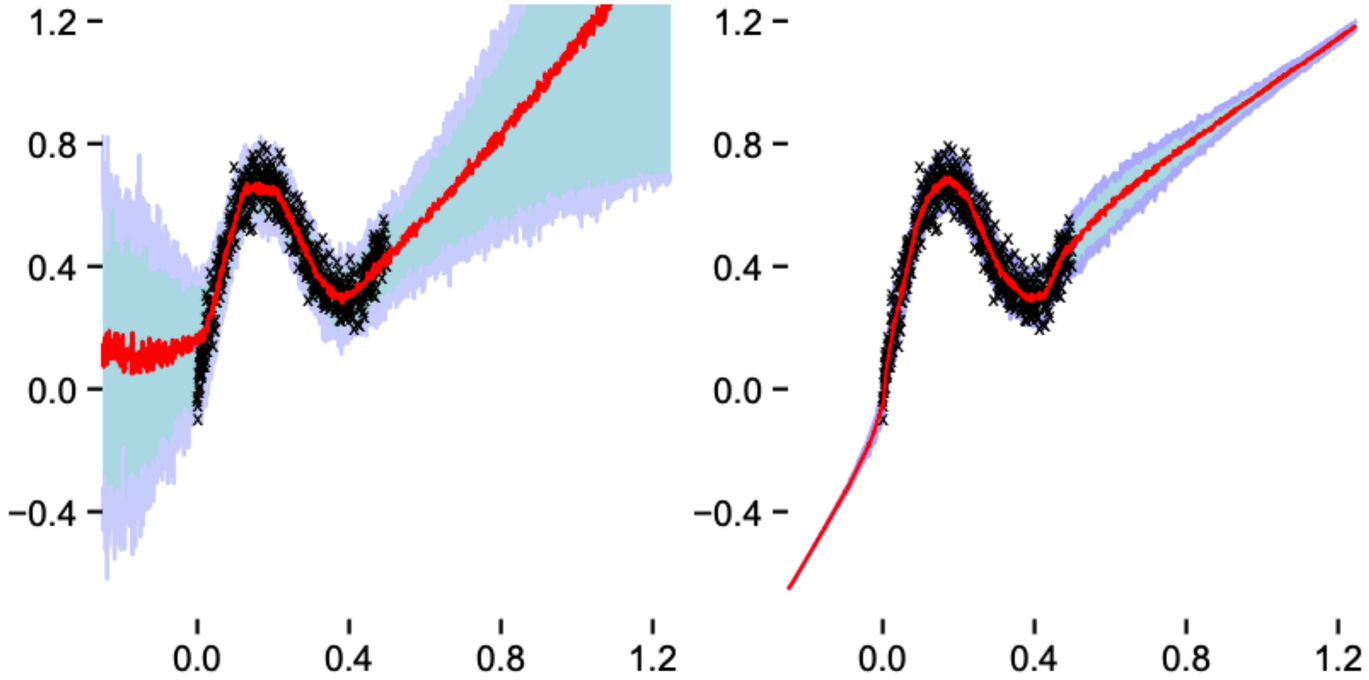


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Predictive Distribution

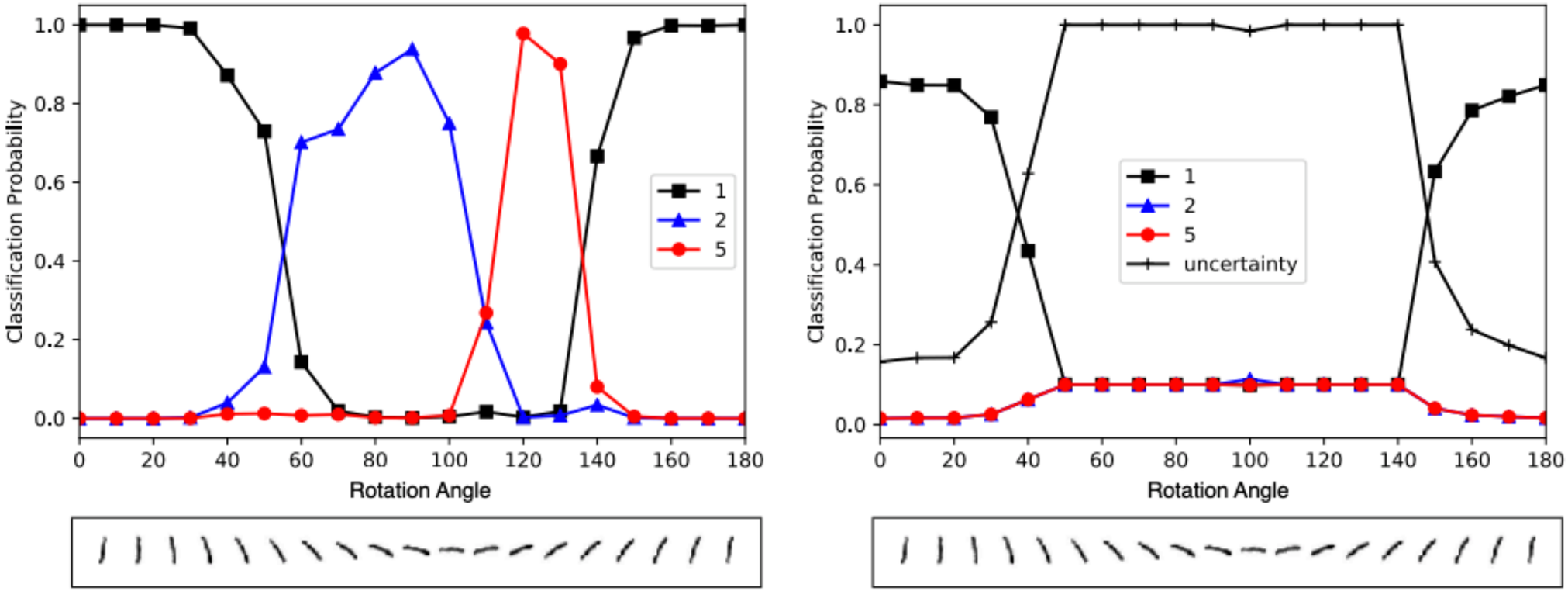
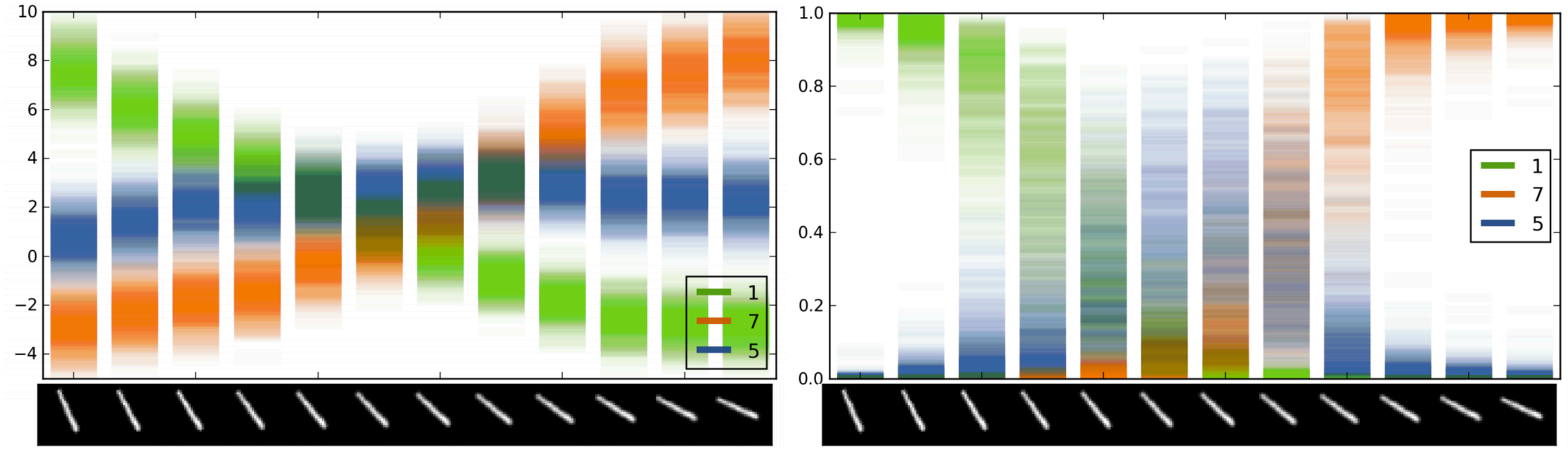


Figure 1: Classification of the rotated digit 1 (at bottom) at different angles between 0 and 180 degrees. **Left:** The classification probability is calculated using the *softmax* function. **Right:** The classification probability and uncertainty are calculated using the proposed method.

Murat Sensoy et al., Evidential Deep Learning to Quantify Classification Uncertainty

Predictive Distribution



(a) Softmax *input* scatter

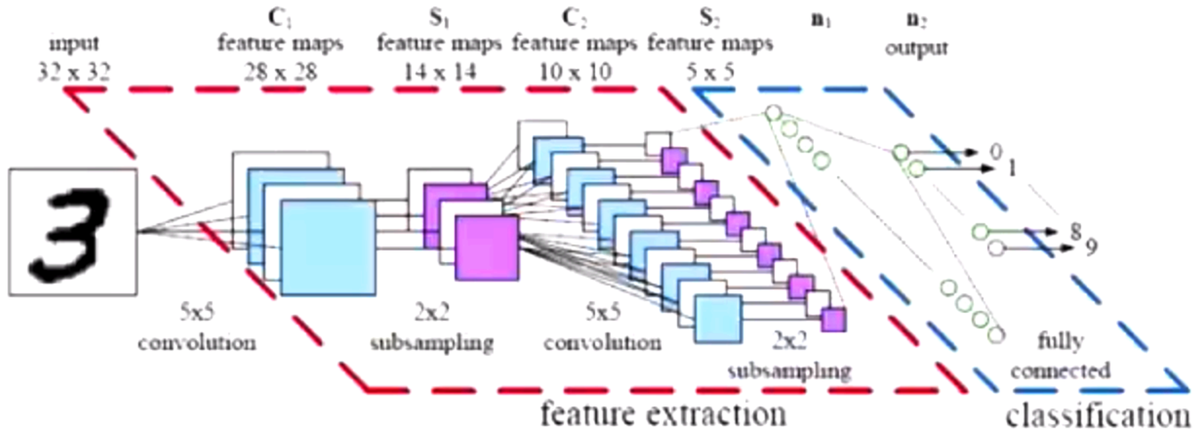
(b) Softmax *output* scatter

Figure 4. **A scatter of 100 forward passes of the softmax input and output for dropout LeNet.** On the X axis is a rotated image of the digit 1. The input is classified as digit 5 for images 6-7, even though model uncertainty is extremly large (best viewed in colour).

Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Bayesian Deep-Learning

Pros and Conds of Deep Learning



A framework for constructing flexible models

+ Rich non-linear models for classification and sequence prediction.

+ Scalable learning using stochastic approximations and conceptually simple.

+ Easily composable with other gradient-based methods

- Only point estimates

- Hard to score models, do model selection and complexity penalisation.

Shakir Mohamed. Bayesian Agents: Bayesian Reasoning and Deep Learning in Agent-based Systems @ Bayesian Deep Learning Workshop NIPS 2016

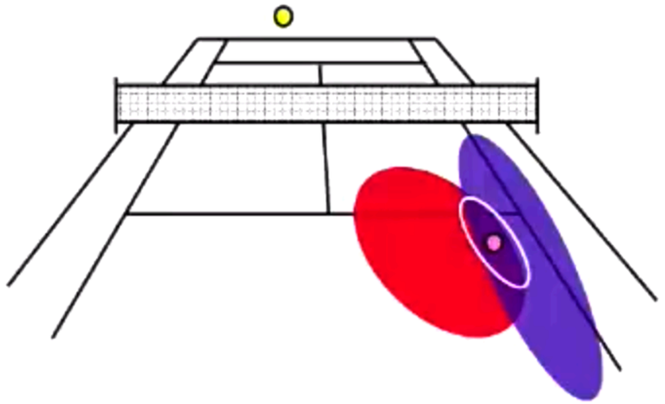
Limitation of Deep Learning

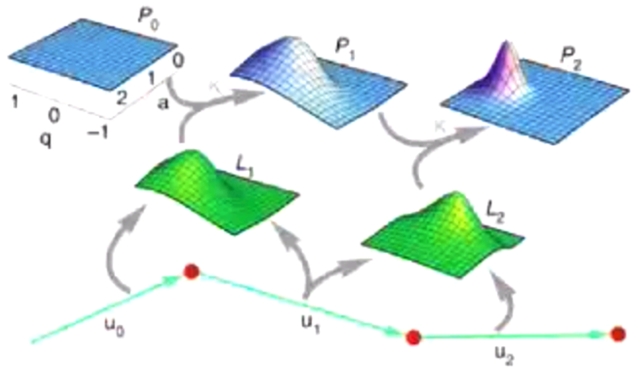
Neural networks and deep learning systems give amazing performance on many benchmark tasks, but they are generally:

- very data hungry (e.g. often millions of examples)
- very compute-intensive to train and deploy (cloud GPU resources)
- poor at representing uncertainty
- easily fooled by adversarial examples
- finicky to optimise: non-convex + choice of architecture, learning procedure, initialisation, etc, require expert knowledge and experimentation
- uninterpretable black-boxes, lacking in trasparency, difficult to trust

Zoubin Ghahramani. History of Bayesian Neural Networks (Keynote talk) @ Bayesian Deep Learning Workshop NIPS 2016

Pros and Conds of Bayesian Reasoning





A framework for inference and decision making

+ Unified framework for model building, inference, prediction and decision making

+ Explicit accounting for uncertainty and variability of outcomes

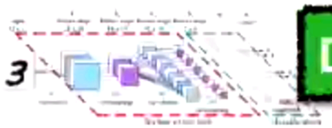
+ Robust to overfitting; tools for model selection and composition.

- Mainly conjugate and linear models

- Intractable inference leading to expensive computation or long simulation times.

Shakir Mohamed. Bayesian Agents: Bayesian Reasoning and Deep Learning in Agent-based Systems
@ Bayesian Deep Learning Workshop NIPS 2016

Bayesian Deep Learning



Deep Learning

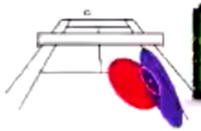
+ Rich non-linear models for classification and sequence prediction.

+ Scalable learning using stochastic approximation and conceptually simple.

+ Easily composable with other gradient-based methods

- Only point estimates

- Hard to score models, do selection and complexity penalisation.



Bayesian Reasoning

- Many conjugate and linear models

- Potentially intractable inference, computationally expensive or long simulation time.

+ Unified framework for model building, inference, prediction and decision making

+ Explicit accounting for uncertainty and variability of outcomes

+ Robust to overfitting; tools for model selection and composition.

Natural to marry these approaches.

Shakir Mohamed. Bayesian Agents: Bayesian Reasoning and Deep Learning in Agent-based Systems @ Bayesian Deep Learning Workshop NIPS 2016

Bayesian Deep Learning

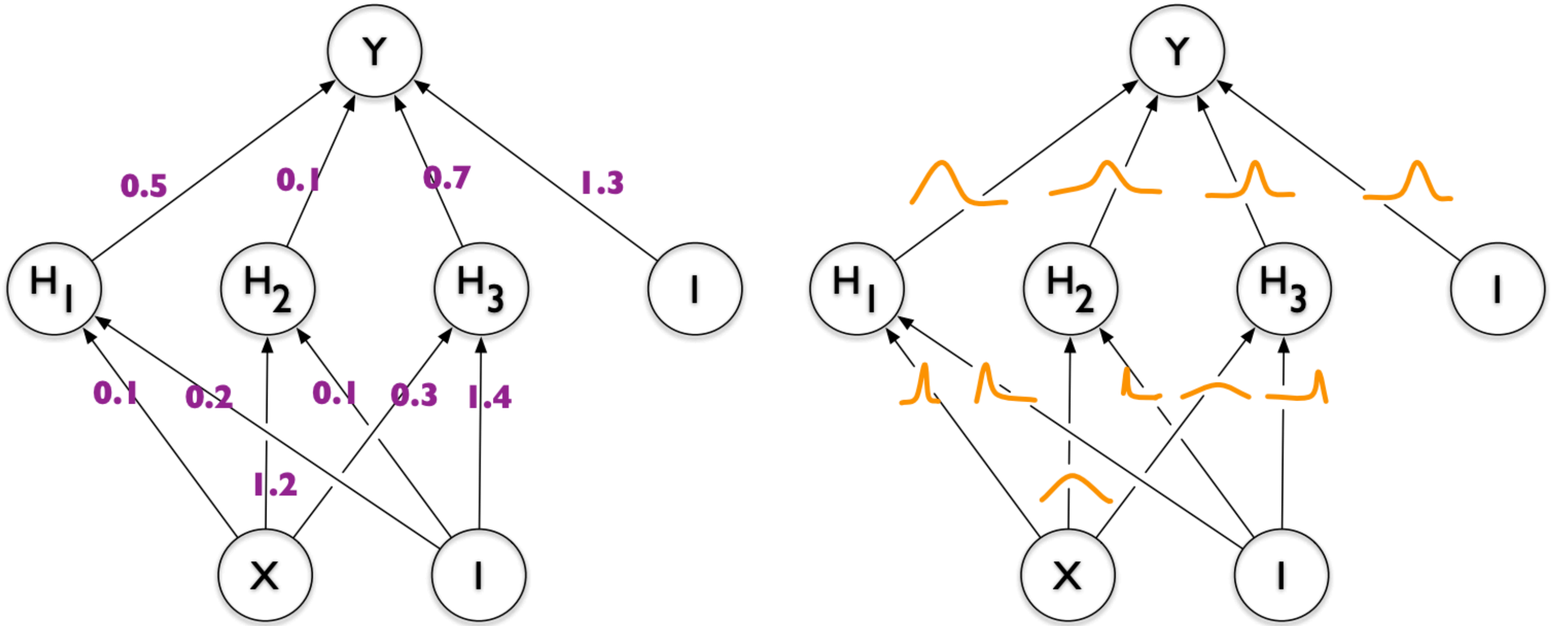
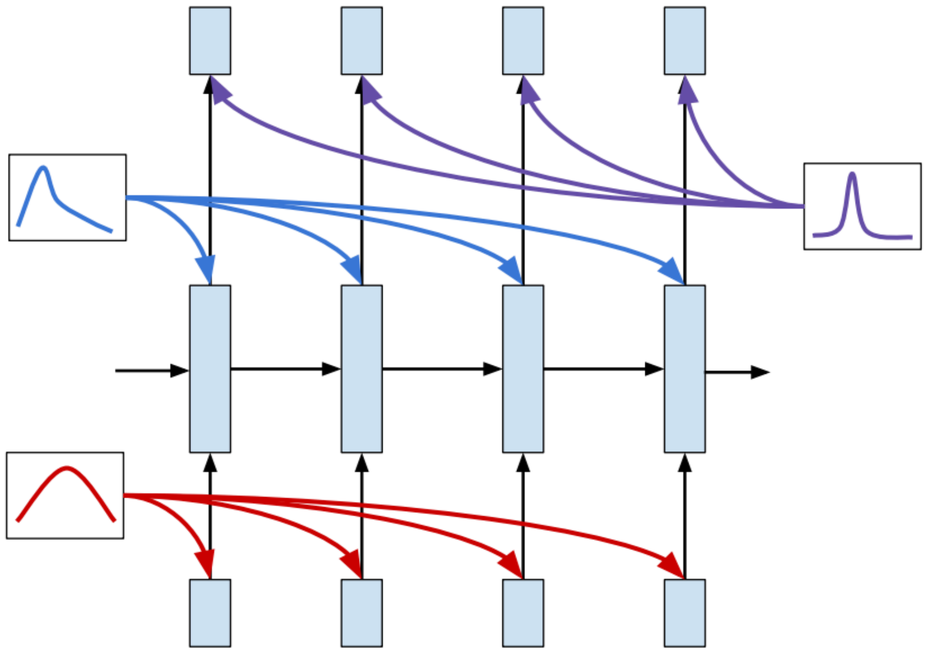


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayesian RNN



Algorithm: Bayes by Backprop for RNNs

Sample $\epsilon \sim \mathcal{N}(0, I)$, $\epsilon \in \mathbb{R}^d$, and set network parameters to $\theta = \mu + \sigma\epsilon$.
Sample a minibatch of truncated sequences (x, y).
Do forward and backward propagation as normal, and let g be the gradient w.r.t θ.
Let $g_\theta^{KL}, g_\mu^{KL}, g_\sigma^{KL}$ be the gradients of $\log \mathcal{N}(\theta|\mu, \sigma^2) - \log p(\theta)$ w.r.t. θ, μ and σ respectively.
Update μ using the gradient $\frac{g + \frac{1}{C} g_\theta^{KL}}{B} + \frac{g_\mu^{KL}}{BC}$.
Update σ using the gradient $\left(\frac{g + \frac{1}{C} g_\theta^{KL}}{B}\right)\epsilon + \frac{g_\sigma^{KL}}{BC}$.

Figure 1: Illustration (left) and Algorithm (right) of Bayes by Backprop applied to an RNN.

Meire Fortunato et al., Bayesian Recurrent Neural Networks

Bayesian RNN

Table 1: Word-level perplexity on the Penn Treebank language modelling task (lower is better), where DE indicates that Dynamic Evaluation was used.

Model (medium)	Val	Test	Val (DE)	Test (DE)
LSTM (Zaremba et al., 2014)	120.7	114.5	-	-
LSTM dropout (Zaremba et al., 2014)	86.2	82.1	79.7	77.1
Variational LSTM (tied weights) (Gal & Ghahramani, 2016)	81.8	79.7	-	-
Variational LSTM (tied weights, MS) (Gal & Ghahramani, 2016)	-	79.0	-	-
Bayesian RNN (BRNN)	78.8	75.5	73.4	70.7
BRNN w/ Posterior Sharpening	$\leq$ **77.8**	$\leq$ **74.8**	$\leq$ **72.6**	$\leq$ **69.8**

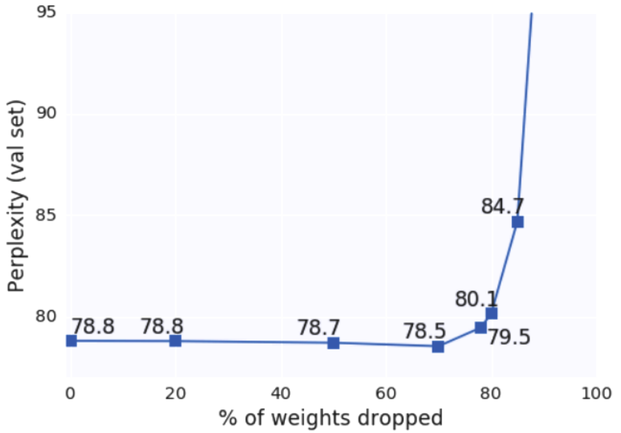


Figure 2: Weight pruning experiment. No significant loss on performance is observed until pruning more than 80% of weights.

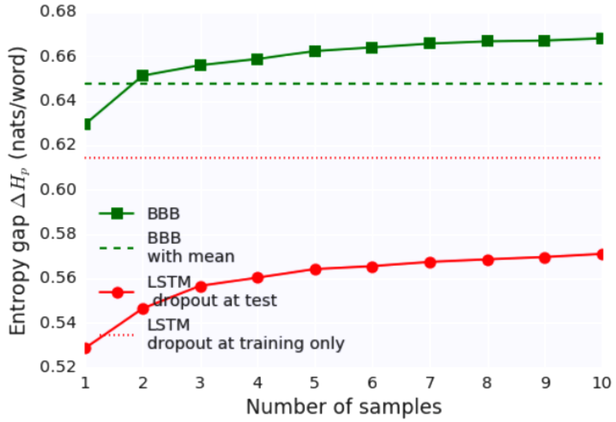


Figure 3: Entropy gap ΔH_p (Eq. (20)) between reversed and regular Penn Treebank test sets $\times$ number of samples.

Meire Fortunato et al., Bayesian Recurrent Neural Networks

Bayesian RNN

Figure 4: Image captioning results on MSCOCO development set.

We used the MSCOCO (Lin et al., 2014) data set and report perplexity, BLUE-4, and CIDER scores on compared to the Show and Tell model (Vinyals et al., 2016), which was the winning entry of the captioning challenge in 2015[3]. The results are:

Model	Perplexity	BLUE-4	CIDER
Show and Tell	8.3	28.8	89.8
Bayes RNN	**8.1**	**30.2**	**96.0**

Meire Fortunato et al., Bayesian Recurrent Neural Networks

Bayesian CNN

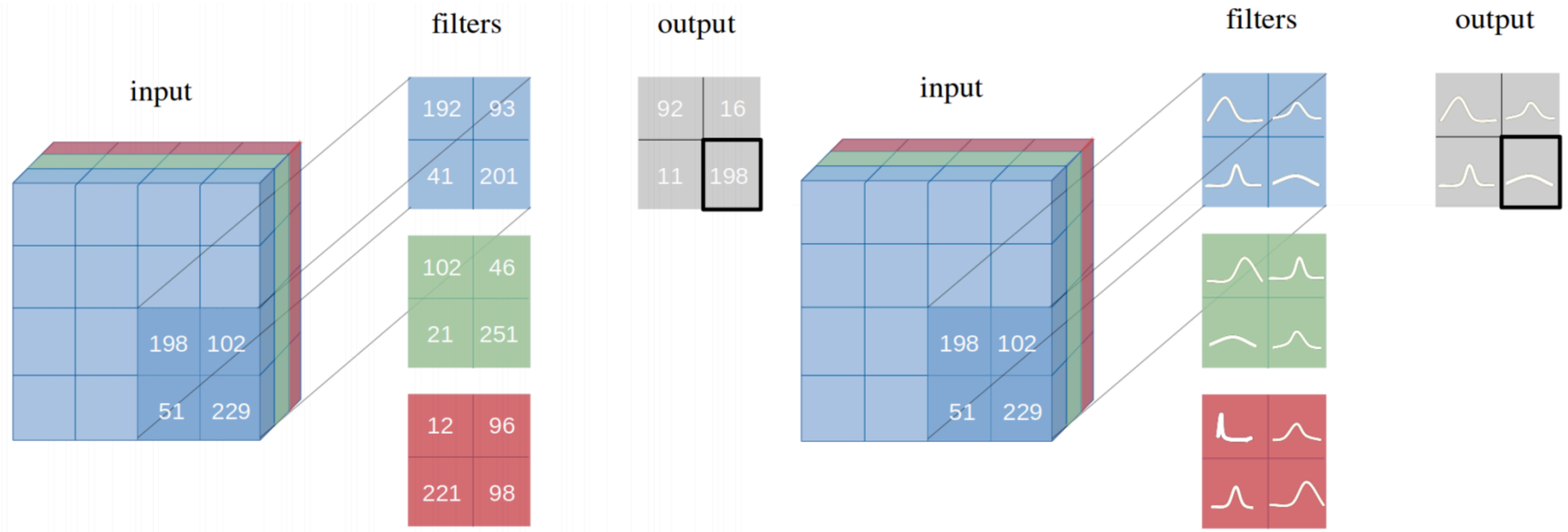


Figure 4: Input image with exemplary pixel values, filters, and corresponding output with point-estimates (top) and probability distributions (bottom) over weights.[55]

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

Bayesian CNN

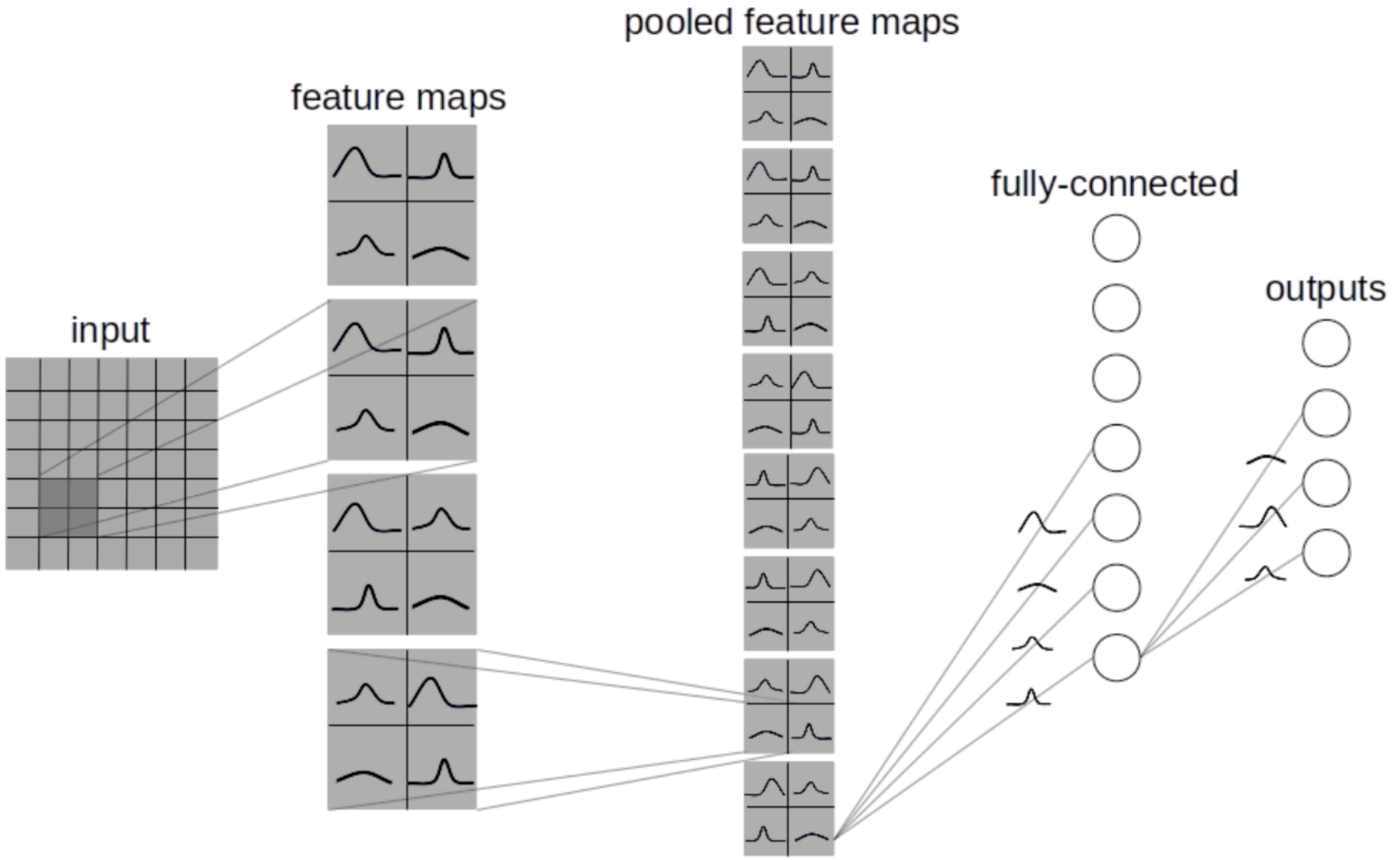


Figure 5: Fully Bayesian perspective of an exemplary CNN. Weights in filters of convolutional layers, and weights in fully-connected layers have the form of a probability distribution. [55]

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

Bayesian CNN

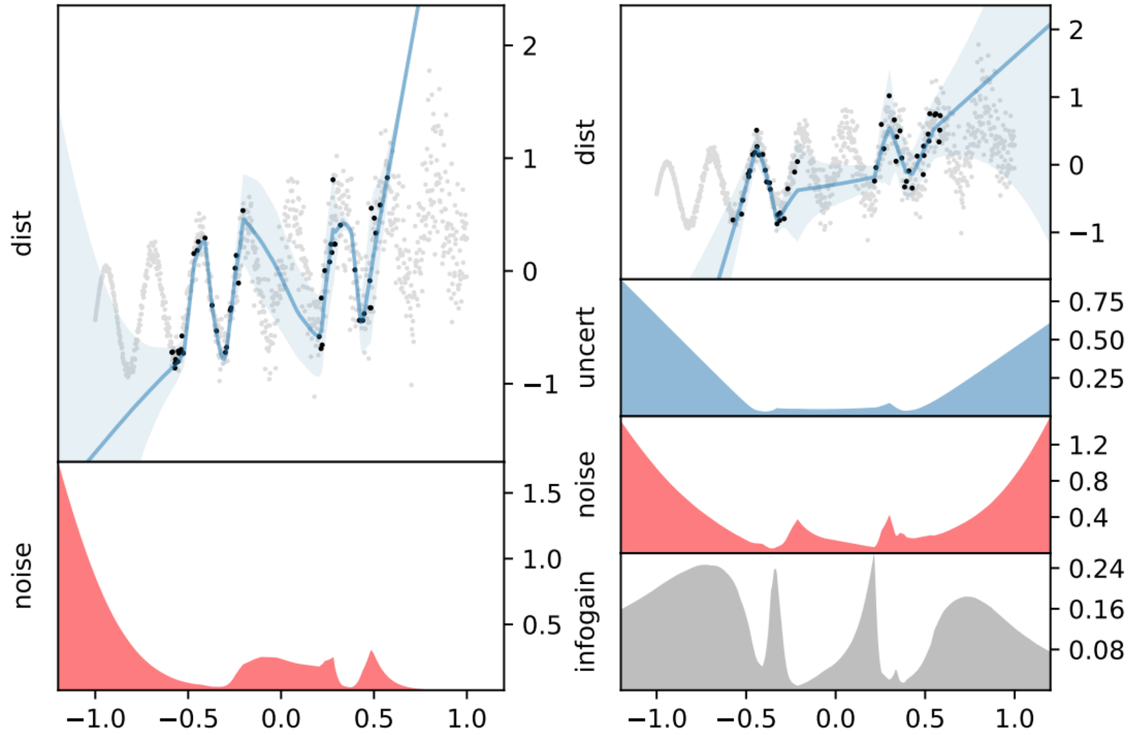


Figure 6: Predictive distributions is estimated for a low-dimensional active learning task. The predictive distributions are visualized as mean and two standard deviations shaded. ■ shows the epistemic uncertainty and ■ shows the aleatoric noise. Data points are shown in ■. **(Left)** A deterministic network conflates uncertainty as part of the noise and is over-confident outside of the data distribution. **(Right)** A variational Bayesian neural network with standard normal prior represents uncertainty and noise separately but is overconfident outside of the training distribution as defined by [22]

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

Bayesian CNN

- BayesCNN for Image Super Resolution

Figure 12: Sample image in Low Resolution image space taken randomly from BSD 300 [46] dataset.

Figure 13: Generated Super Resolution Image scaled to 40 percent to fit

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

When is the probabilistic approach essential?

Many aspects of learning and intelligence depend crucially on the careful probabilistic representation of *uncertainty*:

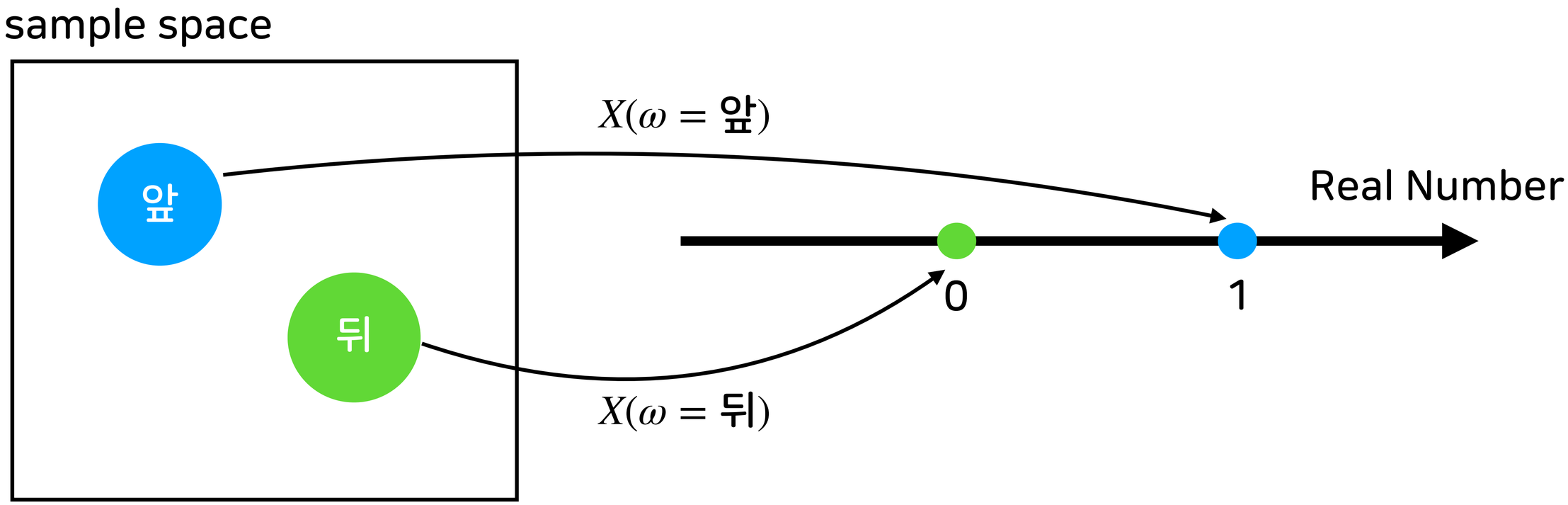
- Forecasting
- Decision making
- Learning from limited, noisy, and missing data
- Learning complex personalised models
- Data compression
- Automating scientific modelling, discovery, and experiment design

Zoubin Ghahramani. History of Bayesian Neural Networks (Keynote talk) @ Bayesian Deep Learning Workshop NIPS 2016

Random Variables & Probability Distriubtions

Random Variable

- 확률 변수 (Random Variable)는 무작위적으로 다른 값을 가질 수 있는 변수를 나타냅니다.
- 좀 더 엄밀하게는 sample space 내의 예측할 수 없는 각 사건들을 실수값에 대응시키는 함수로 생각할 수 있습니다.



Probability Distribution

- 확률 분포 (probability distribution)는 random variable이 가질 수 있는 값들의 가능성을 나타냅니다.

- Probability distribution은 discrete random variable에 대한 Probability Mass Function (PMF)와 continuous random variable에 대한 Probability Density Function(PDF) 두 종류가 있습니다.

Bernoulli-정의

- 베르누이 분포 (Bernoulli distribution)은 0 또는 1 두가지 값을 가지는 random variable의 확률 분포입니다.
- Random variable이 1 값을 가질 확률을 나타내는 μ를 parameter로 가집니다.
- Bernoulli distribution의 PMF는 다음과 같은 식으로 표현됩니다.
-

$$p(x|\mu) = Ber(x|\mu) = \begin{cases} \mu, & \text{if } x = 1 \\ 1-\mu, & \text{if } x = 0 \end{cases}$$

- 또는 다음과 같이 표현 할 수도 있습니다.
-

$$p(x|\mu) = Ber(x|\mu) = \mu^x(1-\mu)^{(1-x)}$$

Bernoulli-최적화

- Dataset $X = \{x_1, x_2, \ldots, x_N\}$이 주어진 경우, likelihood function을 다음과 같이 표현할 수 있습니다.

- $$p(X|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{(1-x_n)}$$

- Frequentist 관점에서 위 식을 최대화 하는 parameter인 μ를 구할 수 있습니다. 또는 단조 증가 함수인 $\log$함수를 likelihood에 적용하여 최대화 할 수도 있습니다.

- $$log\ p(X|\mu) = \sum_{n=1}^{N} log\ p(x_n|\mu) = \sum_{n=1}^{N} \{x_n log\ \mu + (1-x_n) log\ (1-\mu)\}$$

Bernoulli-최적화

- Data의 likelihood 또는 log likelihood를 최대화하는 최적화 방법을 Maximum Likelihood라고 합니다.
- Maximum Likelihood를 통해 얻은 파라메터 μ의 값은 다음과 같습니다.
-

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

- 이는 전체 데이터 중 1값을 갖는 데이터의 비율과 같습니다.

Binomial-정의

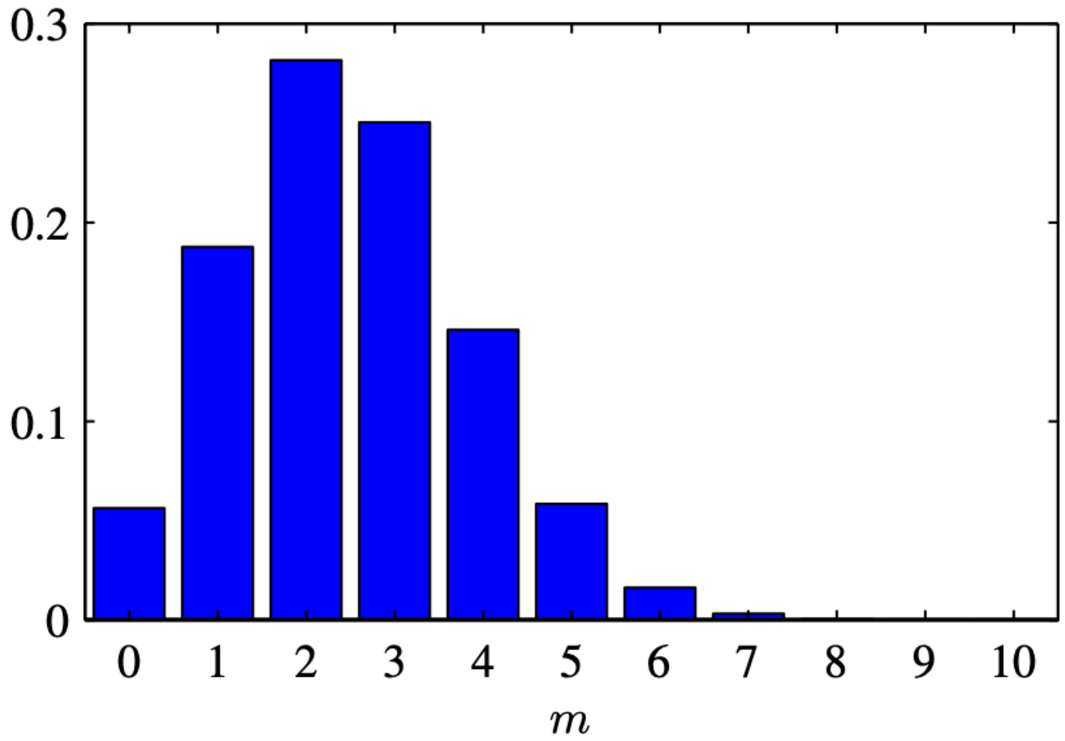
- Binomial distribution은 베르누이 시행(Bernoulli trials)을 N번 독립적으로 했을 때 얻을 수 있는 1의 갯수에 대한 확률 분포입니다.
- 총 시행 횟수 N과 1 값이 발생할 확률 μ를 파라메터로 갖습니다.
- Binomial distribution의 PMF는 다음과 같이 정의할 수 있습니다.
-

$$Bin(m \mid N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}$$

$$\text{where } \binom{N}{m} \equiv \frac{N!}{(N-m)!m!}$$

Binomial

Figure 2.1 Histogram plot of the binomial distribution (2.9) as a function of m for $N = 10$ and $\mu = 0.25$.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.70

Categorical-정의

- Categorical distribution은 K개의 discrete 값을 가질 수 있는 random variable에 대한 probability distribution입니다.
- Parameter로 각 카테고리에 대한 확률 값들을 나타내는 벡터 $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_K)$를 갖습니다.
- Random variable이 갖는 값은 one-hot 벡터로 나타낼 수 있습니다. 예를 들어 6개의 카테고리가 있고 random variable이 3번째 카테고리 값을 갖는다면 다음과 같이 나타낼 수 있습니다.
-

$$\mathbf{x} = (0,0,1,0,0,0)^T$$

- Categorical distribution의 PMF는 다음과 같이 표현할 수 있습니다.
-

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

Categorical-최적화

- Dataset $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$이 주어진 경우, likelihood function을 다음과 같이 표현할 수 있습니다.

- $$p(\mathbf{X}|\boldsymbol{\mu}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mu_k^{\mathbf{x}_{nk}} = \prod_{k=1}^{K}\mu_k^{\left(\Sigma_n \mathbf{x}_{nk}\right)} = \prod_{k=1}^{K}\mu_k^{N_k}$$

- Frequentist 관점에서 위 식을 최대화 하는 파라메터인 $\boldsymbol{\mu}$를 구할 수 있습니다. 또는 단조 증가 함수인 $\log$함수를 likelihood에 적용하여 최대화 할 수도 있습니다.

- $$\log p(\mathbf{X}|\boldsymbol{\mu}) = \sum_{k=1}^{K} N_k \log \mu_k$$

Categorical-최적화

- 단, μ_k값들의 합이 1이 되어야 하는 조건을 걸기 위해, Lagrange multiplier를 이용하여 다음 식을 최대화할 수 있습니다.

- $$\log p(\mathbf{X}\,|\,\boldsymbol{\mu}) = \sum_{k=1}^{K} N_k \log \mu_k + \lambda\Big(\sum_{k=1}^{K} \mu_k - 1\Big)$$

- 위 식을 최대화하는 μ_k는 다음과 같습니다.

- $$\mu_k = \frac{N_k}{N}$$

- 이는 전체 데이터 중 k번째 카테고리를 갖는 데이터의 비율과 같습니다.

Categorical-최적화

- 증명

$$\frac{\partial \log p(\mathbf{X}|\boldsymbol{\mu})}{\partial \mu_k} = \frac{N_k}{\mu_k} + \lambda, \frac{\partial \log p(\mathbf{X}|\boldsymbol{\mu})}{\partial \lambda} = \sum_{k=1}^{K} \mu_k - 1$$

- 위 두 편미분 값을 0으로 두면,

$$\frac{N_k}{\mu_k} + \lambda = 0, \sum_{k=1}^{K} \mu_k - 1 = 0$$

- 위 두식을 정리하면,

$$\lambda = -N, \mu_k = \frac{N_k}{N}$$

Multinomial-정의

- Multinomial distribution은 K개의 다른 카테고리를 가질 수 있는 random variable에서 N번 독립적으로 값을 얻었을 때, 각 카테고리가 N_k번씩 선택될 확률에 대한 분포입니다.
- Parameter로 각 카테고리에 대한 확률 값 $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_K)$와 시행 횟수 N을 갖습니다.
- Multinomial distribution의 PMF는 다음과 같이 표현할 수 있습니다.
-

$$Mult(N_1, N_2, \ldots, N_K | \boldsymbol{\mu}, N) = \binom{N}{N_1 N_2 \cdots N_K} \prod_{k=1}^{K} \mu_k^{N_k}$$

$$\text{where } \binom{N}{N_1 N_2 \cdots N_K} \equiv \frac{N!}{N_1! N_2! \cdots N_K!}$$

Gaussian-정의

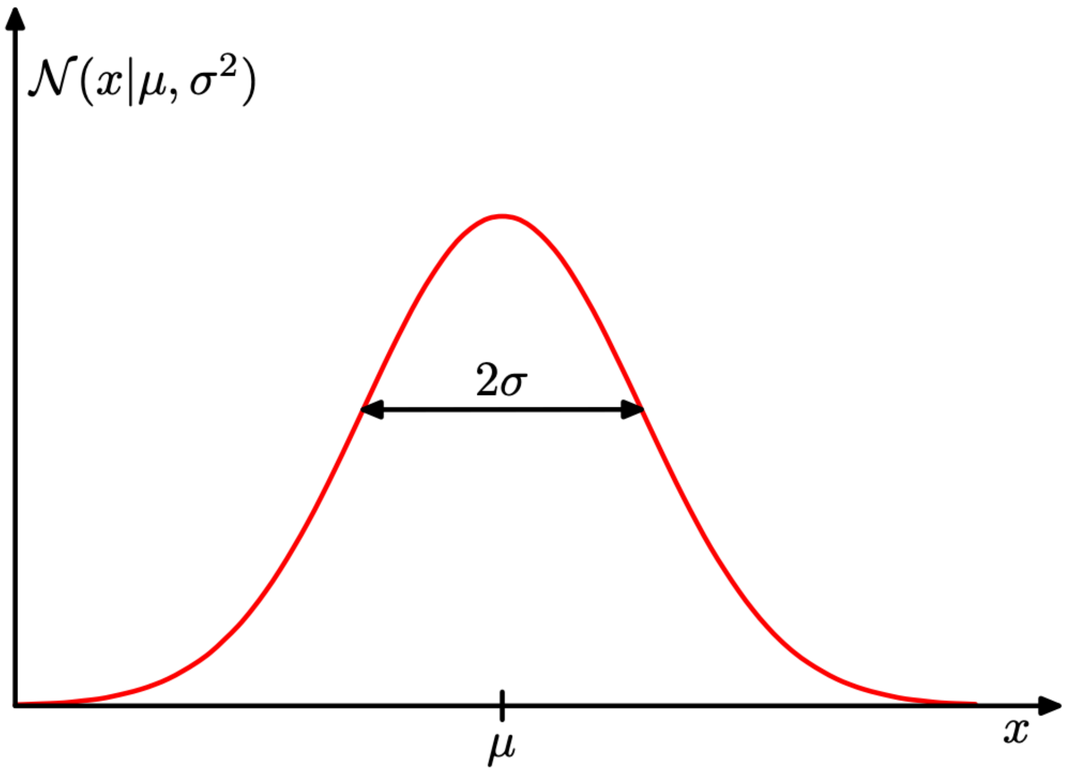
- Gaussian Distribution의 PDF(Probability Distribution Function)은 다음과 같습니다.
-

$$\mathcal{N}\left(x \mid \mu, \sigma^2\right) = \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

- Parameter로 mean값을 나타내는 μ와, 분산을 나타내는 σ^2을 갖습니다.

Gaussian

Figure 1.13 Plot of the univariate Gaussian showing the mean μ and the standard deviation σ.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.25

Gaussian-최적화

- Gaussian Distribution에서 독립적으로 샘플링한 데이터셋 $X = \{x_1, \ldots, x_N\}^T$에 대해 log likelihood function은 다음과 같이 쓸 수 있습니다.

$$\ln p(X \mid \mu, \sigma^2) = -\frac{N}{2}\ln 2\pi - \frac{N}{2}\ln \sigma^2 - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2$$

- Frequentist 관점에서 위 식을 최대화하는 $\boldsymbol{\mu}$와 $\boldsymbol{\Sigma}$는 다음과 같습니다.

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$$

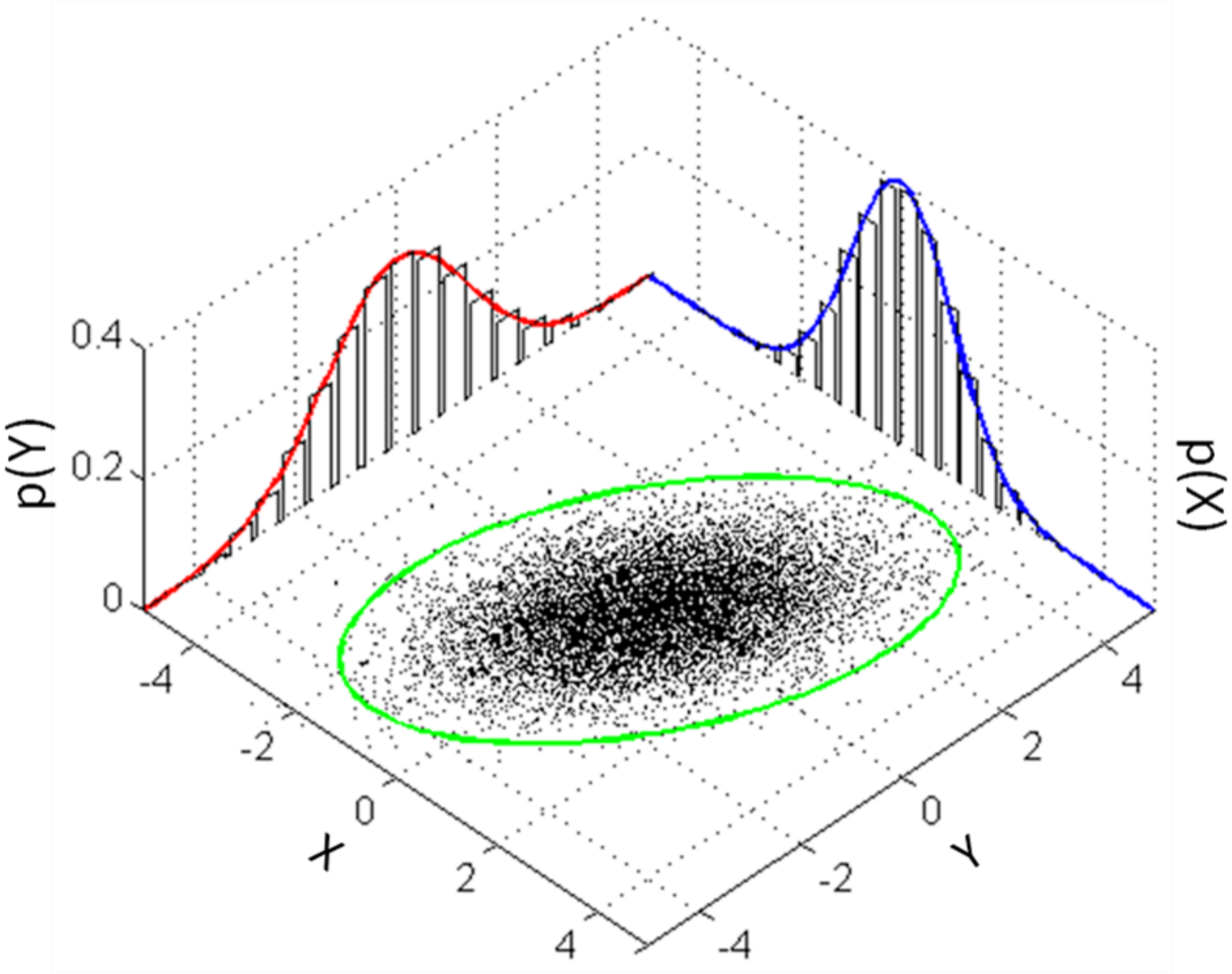
Multivariate Gaussian-정의

- 데이터를 나타내는 $\mathbf{x}$ 가 D 차원일 벡터일 때, Multivariate Gaussian Distribution의 PDF(Probability Distribution Function)은 다음과 같습니다.

- $$\mathcal{N}(\mathbf{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

- Parameter로 mean값을 나타내는 $\boldsymbol{\mu}$와, covariance matrix인 $\boldsymbol{\Sigma}$을 갖습니다.

Bivariate Gaussian distribution



en.wikipedia.org/wiki/Multivariate_normal_distribution

Multivariate Gaussian-최적화

- Multivariate Gaussian Distribution에서 독립적으로 샘플링한 데이터셋 $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}^T$에 대해 log likelihood function은 다음과 같이 쓸 수 있습니다.

- $$\ln p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{x}_n - \boldsymbol{\mu}\right)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}_n - \boldsymbol{\mu}\right)$$

- Frequentist 관점에서 위 식을 최대화하는 $\boldsymbol{\mu}$와 $\boldsymbol{\Sigma}$는 다음과 같습니다.

- $$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$
$$\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}}\right)^{\mathrm{T}}$$

Probability Theory

Joint & Marginal Probability Distribution

$P(\mathrm{x}=x,\mathrm{y}=y)$	y_1	y_2	y_3	$P(\mathrm{x}=x)$
x_1	3/20	5/20	4/20	12/20
x_2	2/20	3/20	3/20	8/20
$P(\mathrm{y}=y)$	5/20	8/20	7/20	20/20

Joint prob. distribution

$$P(\mathrm{x}=x,\mathrm{y}=y)$$

Marginal probability distribution

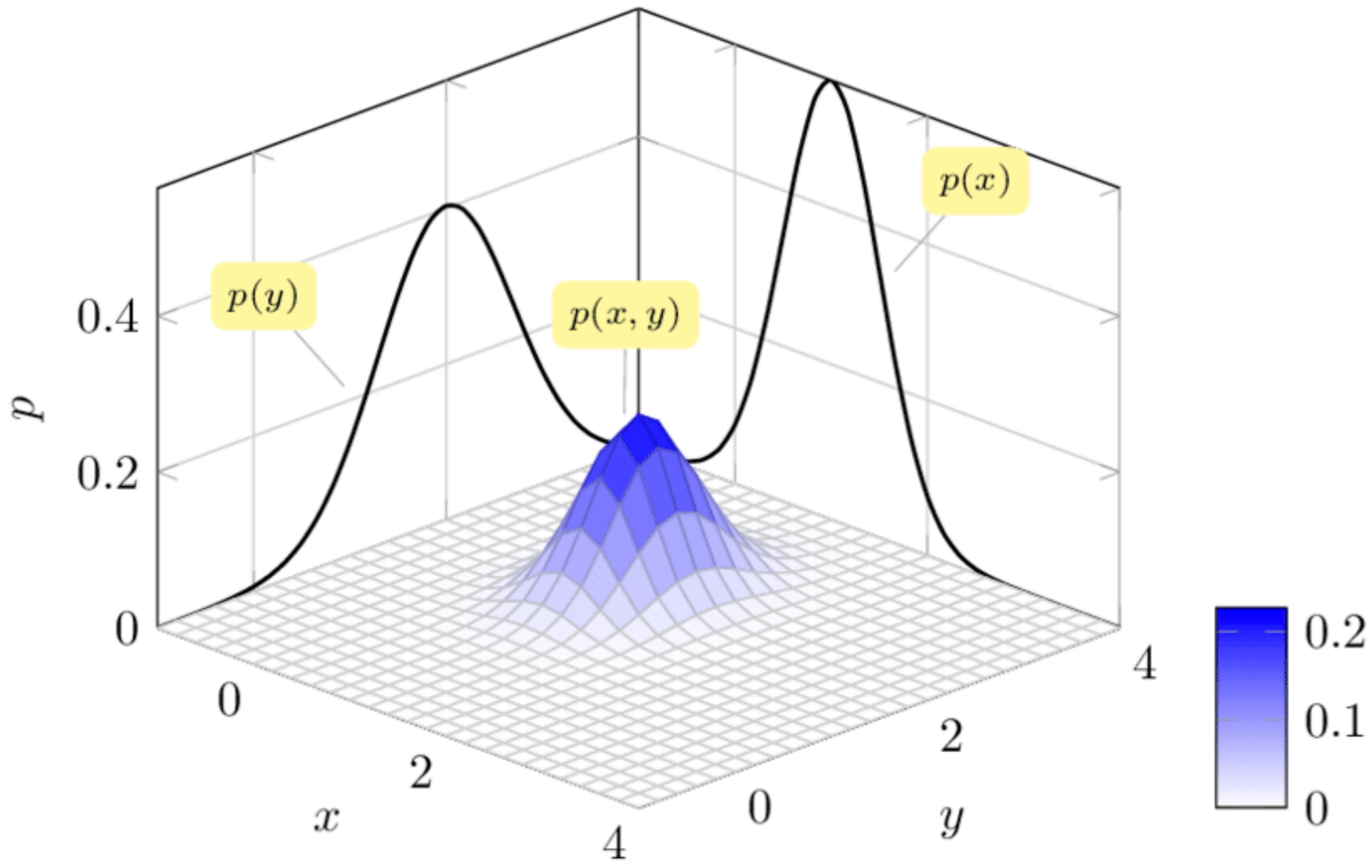
$$\forall x \in \mathrm{x}, P(\mathrm{x}=x) = \sum_y P(\mathrm{x}=x,\mathrm{y}=y)$$

$$\forall y \in \mathrm{y}, P(\mathrm{y}=y) = \sum_x P(\mathrm{x}=x,\mathrm{y}=y)$$

Continuous case

$$p(x) = \int p(x,y)\mathrm{d}y$$

Joint & Marginal Probability Distribution



https://www.researchgate.net/figure/Illustration-of-a-bivariate-Gaussian-distribution-The-marginal-and-joint-probability_fig1_320182941

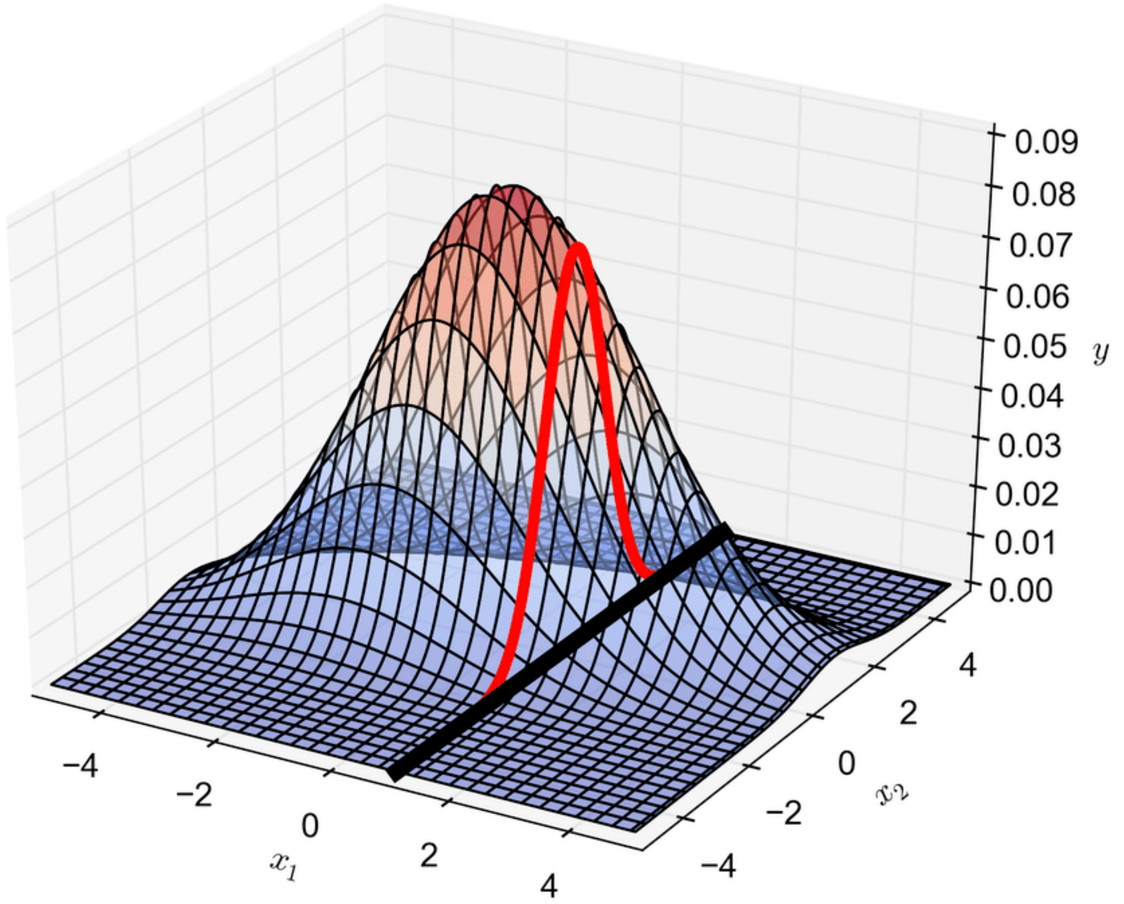
Conditional Probability Distribution

$P(\mathrm{x}=x,\mathrm{y}=y)$	y_1	y_2	y_3	$P(\mathrm{x}=x)$
x_1	3/20	5/20	4/20	12/20
x_2	2/20	3/20	3/20	8/20
$P(\mathrm{y}=y)$	5/20	8/20	7/20	20/20

Conditional probability

$$P(\mathrm{y}=y|\mathrm{x}=x)=\frac{P(\mathrm{y}=y,\mathrm{x}=x)}{P(\mathrm{x}=x)},\quad \text{when } P(\mathrm{x}=x)>0$$

Conditional Probability Distribution



https://www.researchgate.net/figure/Linking-two-points-using-a-bivariate-Gaussian-distribution-enables-the-conditional-pdf_fig1_318429539

Expectation

- Discrete probability distribution $p(x)$에 대한 function $f(x)$의 기댓값(expectation)은 다음과 같이 계산됩니다.
-

$$\mathbb{E}[f] = \sum_{x} p(x)f(x)$$

- 또는 $p(x)$가 continuous인 경우 다음과 같습니다.
-

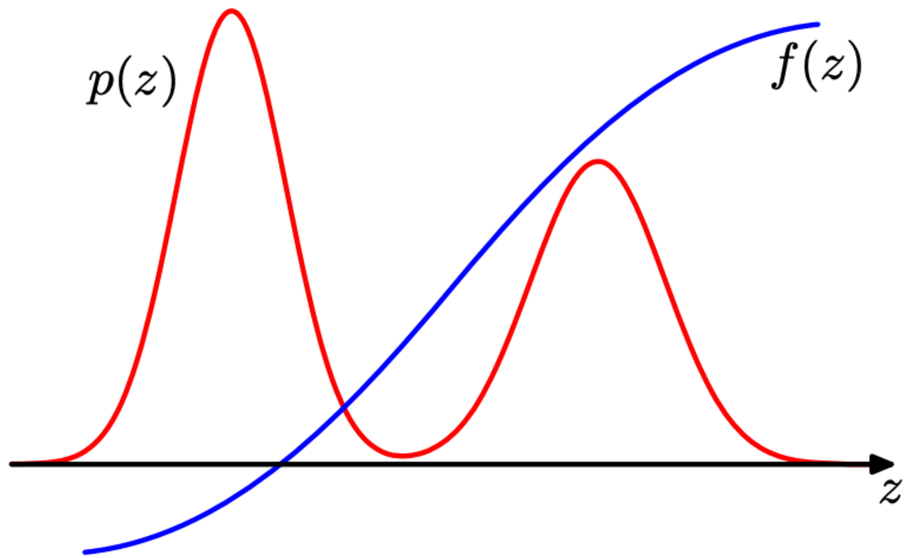
$$\mathbb{E}[f] = \int p(x)f(x)\mathrm{d}x$$

- Expectation은 다음과 같이 sampling을 통한 근사(approximation)으로 계산할 수도 있습니다.
-

$$\mathbb{E}[f] \simeq \frac{1}{N}\sum_{n=1}^{N} f\left(x_n\right)$$

Expectation

Figure 11.1 Schematic illustration of a function $f(z)$ whose expectation is to be evaluated with respect to a distribution $p(z)$.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.524

Bayesian Statistics

Prior

- 베이지안에서는 Parameter나 갖고 있지 않은 hidden variable $\mathbf{z}$를 고정된 값이 아닌 random variable로 여기고 이에 대한 distribution을 prior로 설정합니다.
- Parameter θ에 대한 prior로 $p(\theta)$와 같이 표기하며, hidden variable에 $\mathbf{z}$대한 prior로 $p(\mathbf{z})$와 같이 표기합니다.
- 데이터가 주어지기 이전이라는 의미에서 사전 분포(prior distribution)라고 칭합니다.
- Prior는 데이터와 무관하게 자유롭게 정할 수 있지만 likelihood function와 곱했을 때 같은 꼴의 수식을 가질 수 있도록 하는 conjugate prior를 주로 사용합니다.
- Bernoulli와 binomial에 대한 conjugate prior로 beta distribution이 있으며, categorical과 multinomial에 대한 conjugate prior로 Dirichlet distribution이 있습니다.
- Gaussian의 mean에 대한 conjugate prior는 그대로 Gaussian을 사용합니다.

Likelihood

- Likelihood는 parameter이나 z-variable가 달라짐에 따라 주어진 데이터가 일어날 가능성에 대한 함수입니다.

- Parameter θ가 달라짐에 따라 주어진 데이터 $\mathbf{x}$의 likelihood는 $\mathscr{L}(\theta|\mathbf{x})$또는 $p(\mathbf{x}|\theta)$로 표기하며, hidden variable $\mathbf{z}$가 달라짐에 따라 주어진 데이터 $\mathbf{x}$의 likelihood는 $\mathscr{L}(\mathbf{z}|\mathbf{x})$ 또는 $p(\mathbf{x}|\mathbf{z})$로 표기합니다.

- Frequentist 관점에서는 이 likelihood function을 최대화하는 parameter를 구합니다.

- data space에서의 함수가 아니라 parameter space에서의 함수입니다.

- parameter가 주어져 있고 데이터를 변수로 하는 conditional distribution과 PMF 또는 PDF식을 공유하므로 이에 유의해야 합니다.

- PMF와 PDF를 공유하되 parameter의 관점에서 바라보기 때문에 적분해서 1이 되지 않을 수 있으므로 distribution이 아닌 function입니다.

Likelihood-Bernoulli

적분해서 1이 되지 않는 function임에 유의

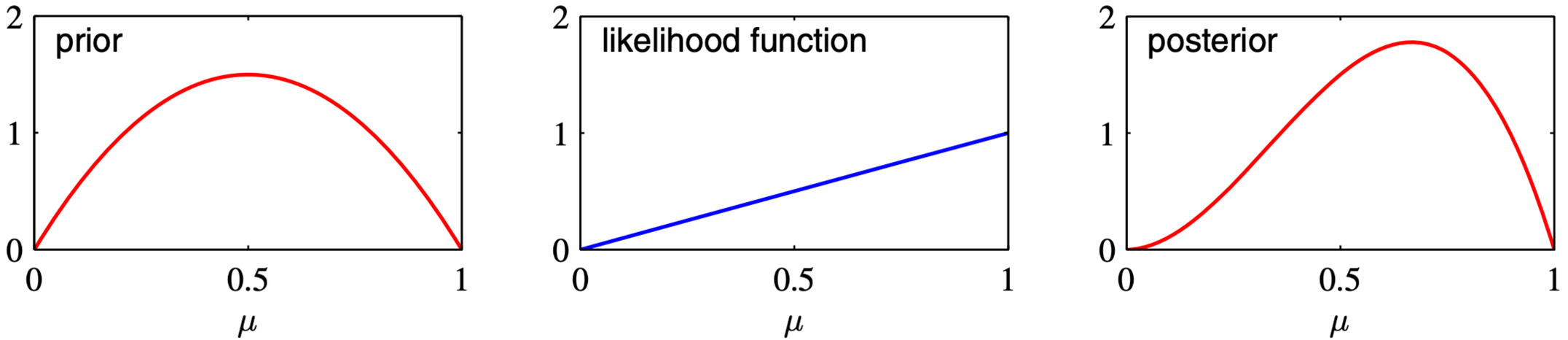


Figure 2.3 Illustration of one step of sequential Bayesian inference. The prior is given by a beta distribution with parameters $a = 2$, $b = 2$, and the likelihood function, given by (2.9) with $N = m = 1$, corresponds to a single observation of $x = 1$, so that the posterior is given by a beta distribution with parameters $a = 3$, $b = 2$.

$$Bin(m \mid N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m} = \binom{1}{1} \mu^1 (1-\mu)^0 = \mu$$

μ가 변수가 됨

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.73

Posterior

- Posterior는 데이터가 주어져 있을 때 parameter나 z-variable의 distribution입니다.
- Bayesian에서 주로 구하려는 대상이 posterior이며, parameter의 posterior의 경우 predictive distribution을 구하는데 사용됩니다(Bayesian regression).
- z-variable의 posterior의 경우 데이터를 좀 더 낮은 차원이나 간단한 형태로 맵핑하는데 사용됩니다(GMM, PPCA, VAE).

Bayes' Rule

- 베이지안 통계, 머신러닝의 근간을 이루는 식으로 prior, likelihood와 posterior의 관계를 나타냅니다.
- 데이터 $\mathbf{x}$가 주어졌을 때 parameter θ의 posterior는 다음과 같이 prior와 likelihood의 곱에 비례하는 형태로 나타낼 수 잇습니다.
-

$$p(\theta \mid \mathbf{x}) = \frac{p(\theta)p(\mathbf{x} \mid \theta)}{p(\mathbf{x})}$$

- $p(\mathbf{x})$는 evidence라고 부르며, prior와 likelihood의 곱에 나누어 확률분포(적분해서 1)가 되도록 normalizing constant 역할을 합니다.
-

$$p(\mathbf{x}) = \sum_{\theta} p(\theta)p(\mathbf{x} \mid \theta), \theta\text{가 discrete한 경우}$$

$$p(\mathbf{x}) = \int p(\theta)p(\mathbf{x} \mid \theta)d\theta, \theta\text{가 continous한 경우}$$

Beta Distribution

- Beta distribution은 Bernoulli distribution과 binomial distribution의 conjugate prior로 사용됩니다.

- Beta distribution의 parameter는 a와 b로 구성되며 PDF는 다음과 같습니다.

-

$$\text{Beta}(\mu \mid a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$$

where $\Gamma(x)$ is the gamma function

- Beta distribution은 discrete distribution에 대한 prior이지만 parameter에 대한 분포이므로 continuous distribution입니다.

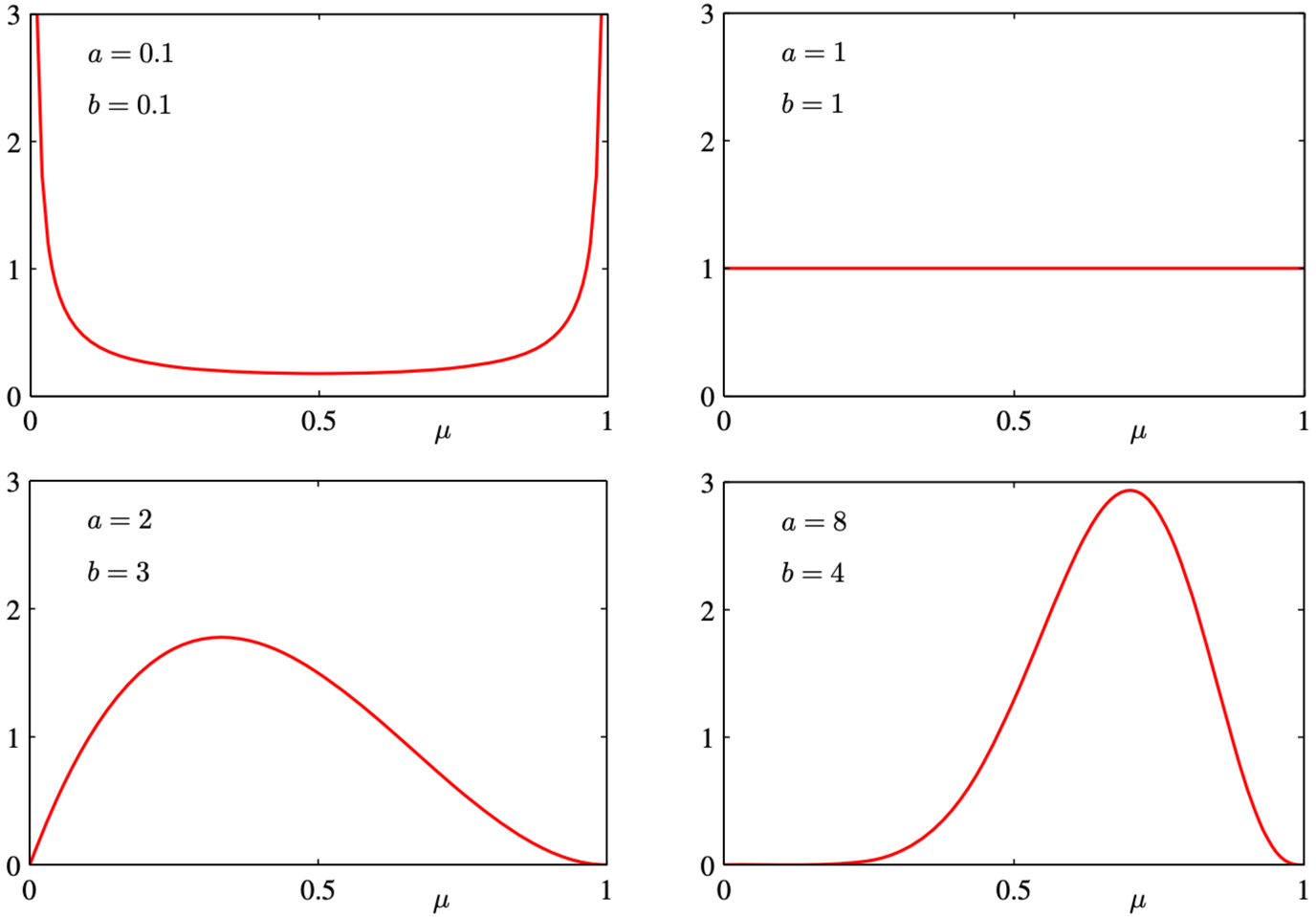
- Gamma function이 들어가 있는 $\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$ 부분은 normalizing constant 역할을 하며 shape을 정하는데 기여하지 않습니다.

Beta Distribution

- Beta distribution의 mean과 variance는 다음과 같습니다. Bernoulli와 conjugate관계를 가지면서 평균이 $\frac{a}{a+b}$가 되도록 만든 distribution이라 생각할 수 있습니다.
-

$$\mathbb{E}[\mu] = \frac{a}{a+b}$$

$$\text{var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)}$$



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.72

Beta Distribution

- Prior인 beta distribution와 likelihood인 binomial likelihood function를 이용하여 posterior를 구할 수 있습니다.
- 이 때, beta distribution은 conjugate prior이므로 posterior는 다시 beta distribution 꼴이 됩니다.
-

$$\text{Beta}(\mu \mid a, b) \cdot \text{Bin}(m \mid N, \mu) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1}(1-\mu)^{b-1} \cdot \binom{N}{m} \mu^{m}(1-\mu)^{N-m}$$

$$\propto \mu^{a+m-1}(1-\mu)^{b+l-1}$$

$$\propto \frac{\Gamma(a+m+b)}{\Gamma(a+m)\Gamma(b+l)} \mu^{a+m-1}(1-\mu)^{b+l-1}$$

$$= \text{Beta}(\mu \mid a+m, b+l), \; where \; l = N-m$$

Beta Distribution

$$\text{Beta}(\mu \mid 2,2) \times \text{Bin}(1 \mid 1,\mu) \propto \text{Beta}(\mu \mid 3,2)$$

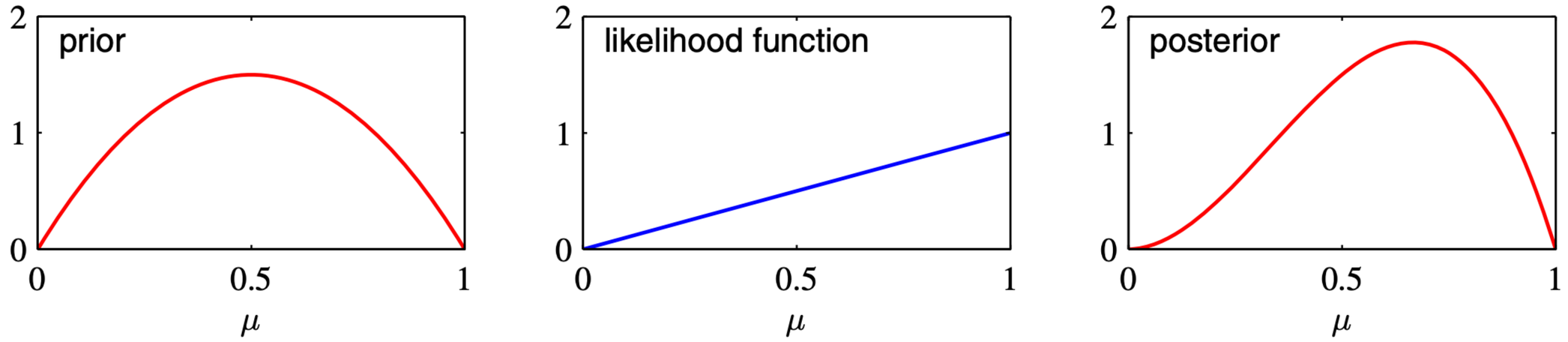


Figure 2.3 Illustration of one step of sequential Bayesian inference. The prior is given by a beta distribution with parameters $a = 2$, $b = 2$, and the likelihood function, given by (2.9) with $N = m = 1$, corresponds to a single observation of $x = 1$, so that the posterior is given by a beta distribution with parameters $a = 3$, $b = 2$.

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.73

Dirichlet Distribution

- Dirichlet distribution은 categorical distribution과 multinomial distribution의 conjugate prior로 사용됩니다.

- Dirichlet distribution의 parameter는 $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)^T$로 구성되며 PDF는 다음과 같습니다.

-

$$\mathrm{Dir}(\boldsymbol{\mu} \mid \boldsymbol{\alpha}) = \frac{\Gamma\left(\alpha_0\right)}{\Gamma\left(\alpha_1\right) \cdots \Gamma\left(\alpha_K\right)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1}$$

where $\Gamma(x)$ is the gamma function and $\alpha_0 = \sum_{k=1}^{K} \alpha_k$

- Dirichlet distribution은 discrete distribution에 대한 prior이지만 parameter에 대한 분포이므로 continuous distribution입니다.

Dirichlet Distribution

- Prior인 Dirichlet distribution에 likelihood인 multinomial likelihood function을 곱하여 posterior를 구할 수 있습니다.
- 이 때, Dirichlet distribution은 conjugate prior이므로 posterior는 다시 Dirichlet distribution 꼴이 됩니다.
-

$$\mathrm{Dir}(\boldsymbol{\mu}\,|\,\boldsymbol{\alpha})\cdot\mathrm{Mult}(\boldsymbol{m}\,|\,\boldsymbol{\mu},N)=\frac{\Gamma\left(\alpha_0\right)}{\Gamma\left(\alpha_1\right)\cdots\Gamma\left(\alpha_K\right)}\prod_{k=1}^{K}\mu_k^{\alpha_k-1}\binom{N}{m_1m_2\cdots m_K}\prod_{k=1}^{K}\mu_k^{m_k}$$

$$\propto\prod_{k=1}^{K}\mu_k^{\alpha_k+m_k-1}$$

$$\propto\frac{\Gamma\left(\alpha_0+m_0\right)}{\Gamma\left(\alpha_1+m_1\right)\cdots\Gamma\left(\alpha_K+m_K\right)}\prod_{k=1}^{K}\mu_k^{\alpha_k+m_k-1}$$

$$=\mathrm{Dir}(\boldsymbol{\mu}\,|\,\boldsymbol{\alpha}+\boldsymbol{m})\text{ , where }\boldsymbol{m}=(m_1,\ldots,m_K)$$

Dirichlet Distribution

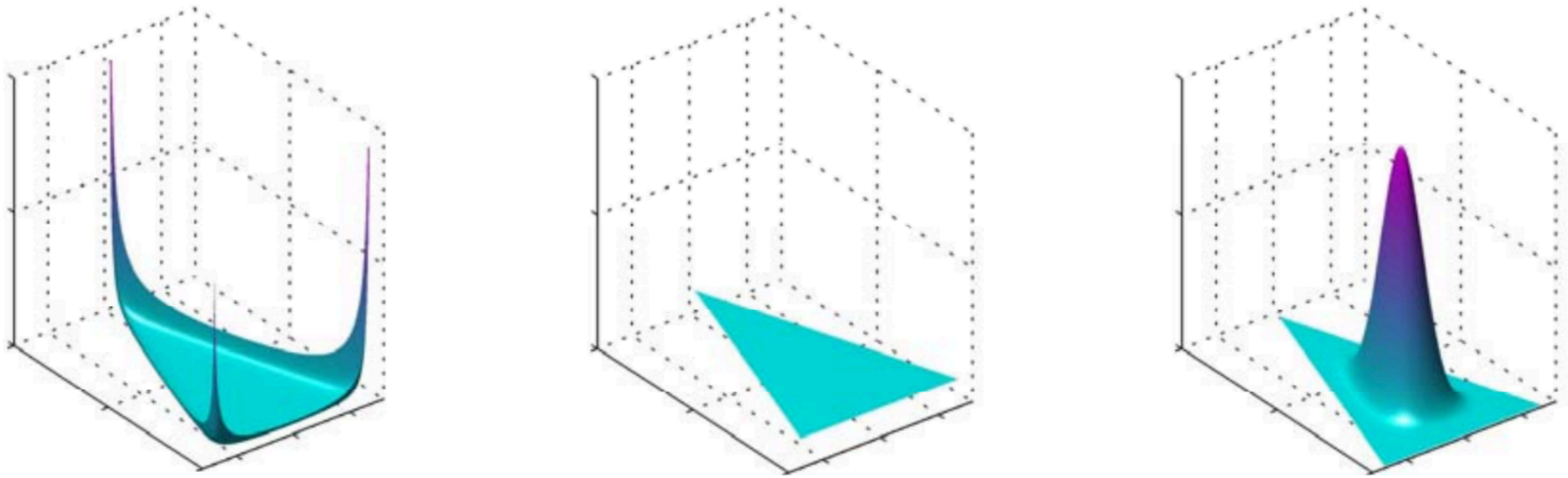


Figure 2.5 Plots of the Dirichlet distribution over three variables, where the two horizontal axes are coordinates in the plane of the simplex and the vertical axis corresponds to the value of the density. Here $\{\alpha_k\} = 0.1$ on the left plot, $\{\alpha_k\} = 1$ in the centre plot, and $\{\alpha_k\} = 10$ in the right plot.

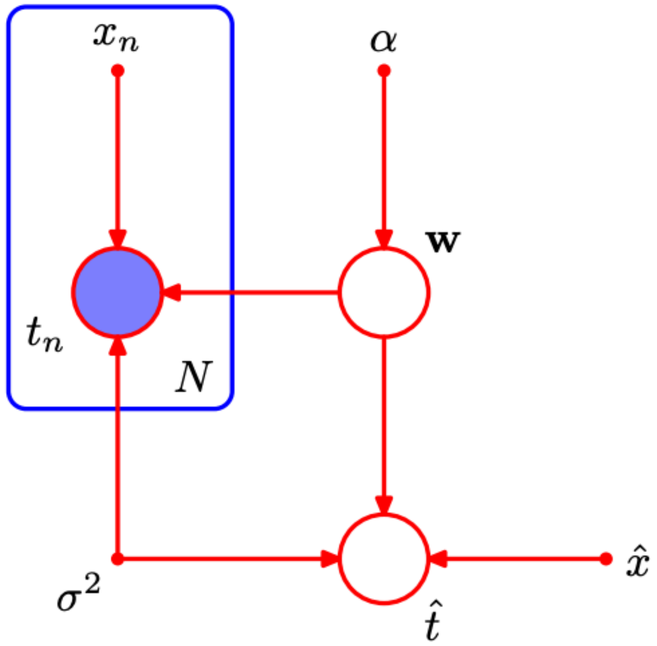
Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.78

Graphical Representation

Graphical Representation의 예

- Linear Regression

Figure 8.7 The polynomial regression model, corresponding to Figure 8.6, showing also a new input value $\widehat{x}$ together with the corresponding model prediction $\widehat{t}$.

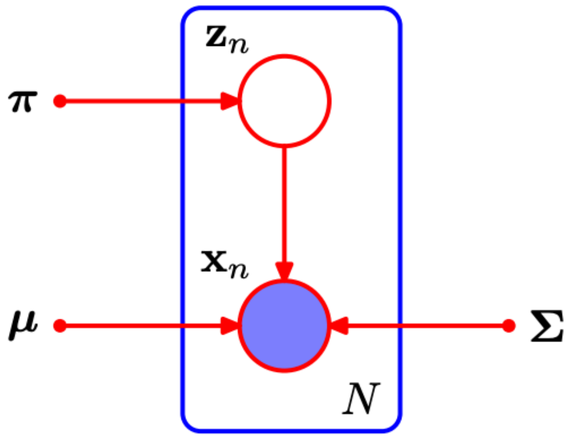


Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.365

Graphical Representation의 예

- Gaussian Mixture Models

Figure 9.6 Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{\mathbf{z}_n\}$, where $n = 1, \ldots, N$.

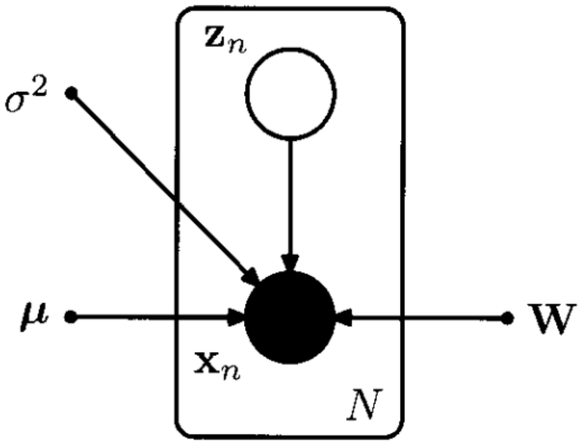


Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433

Graphical Representation의 예

- Probabilistic PCA

Figure 12.10 The probabilistic PCA model for a data set of N observations of $\mathbf{x}$ can be expressed as a directed graph in which each observation $\mathbf{x}_n$ is associated with a value $\mathbf{z}_n$ of the latent variable.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.574

Graphical Representation의 예

- Variational Auto-Encoders

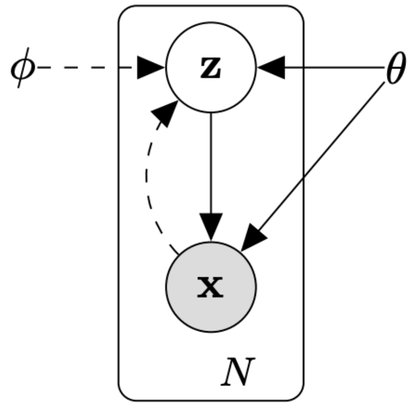
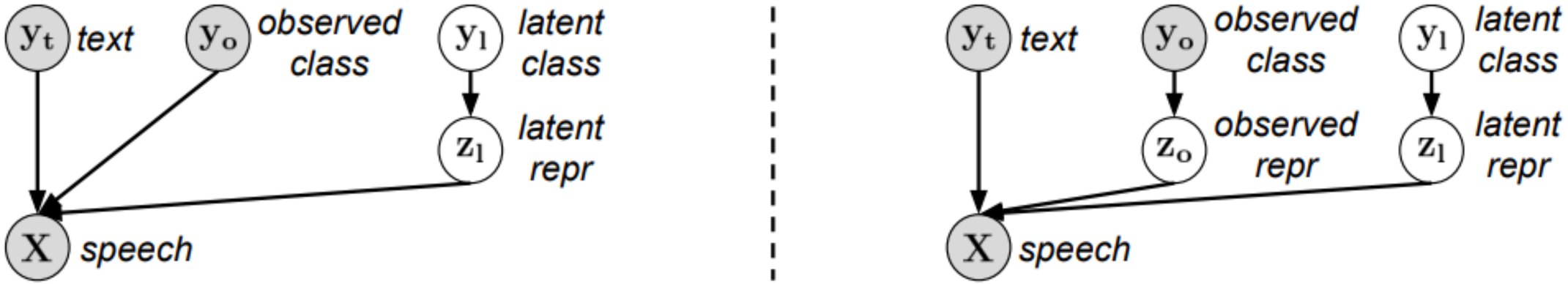


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.

Diederik P. Kinema, Max Welling. Auto-Encoder Variational Bayes.

Graphical Representation의 예

- Controllable speech synthesis (Tacotron)



Hsu, Wei-Ning, et al. "Hierarchical generative modeling for controllable speech synthesis." arXiv preprint arXiv:1810.07217 (2018).

Linear Regression

Linear Basis Function Models

- D차원의 데이터 $\mathbf{x} = (x_1, \ldots, x_D)^T$가 주어지고 이에 대한 weight $\mathbf{w} = (w_0, w_1, \ldots, w_D)^T$가 주어졌을 때 linear combination의 output y를 다음과 같이 나타낼 수 있습니다.
-

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

- 또는 데이터 $\mathbf{x}$를 임의의 basis function $\phi : \mathbb{R}^D \mapsto \mathbb{R}^M$에 의해 매핑시키고 weight $\mathbf{w} = (w_0, w_1, \ldots, w_M)^T$을 적용해볼 수 있습니다.
-

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

- w_0은 bias라 부르며, $\phi_0(\mathbf{x}) = 1$로 정의하여 다음과 같이 위 식을 간단히 나타낼 수 있습니다.
-

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \text{ where } \mathbf{w} = \left(w_0, \ldots, w_{M-1}\right)^{\mathrm{T}} \text{ and } \boldsymbol{\phi} = \left(\phi_0, \ldots, \phi_{M-1}\right)^{\mathrm{T}}$$

Maximum Likelihood and Least Squares

- Input values $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_D)^T$에 대한 target values $\mathbf{t} = (t_1, \ldots, t_D)^T$가 dataset으로 주어졌을 때 다음과 같이 deterministic 함수와 노이즈를 포함한 식을 세워볼 수 있습니다.

- $$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

- 노이즈 값 ϵ이 Gaussian distribution을 따른다 할 때 variance의 역수 precision을 β로 두어 datapoint 하나에 대한 likelihood를 표현할 수 있습니다.

- $$p(t \mid \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}\left(t \mid y(\mathbf{x}, \mathbf{w}), \beta^{-1}\right)$$

Maximum Likelihood and Least Squares

- Dataset 안의 각 datapoint들이 각각 independent한 sample이라는 가정하에 dataset 전체의 likelihood를 나타낼 수 있습니다.

- $$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n \mid \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right), \beta^{-1}\right)$$

- 최적화 단계에서 계산의 편의를 위해 log를 씌워 다시 쓰면 다음과 같습니다. 이를 log-likelihood라고 부릅니다.

- $$\ln p(\mathbf{t} \mid \mathbf{w}, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}\left(t_n \mid \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right), \beta^{-1}\right)$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})$$

, where $E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right) \right\}^2$

Maximum Likelihood and Least Squares

- Log-likelihod를 최대화 하는 $\mathbf{w}$를 찾기 위해 gradient를 구합니다.
- $$\nabla \ln p(\mathbf{t} \mid \mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right) \right\} \boldsymbol{\phi}\left(\mathbf{x}_n\right)^{\mathrm{T}}$$
- Gradient를 0으로 만드는 $\mathbf{w}$를 구합니다.
- $$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}\left(\mathbf{x}_n\right)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{\phi}\left(\mathbf{x}_n\right) \boldsymbol{\phi}\left(\mathbf{x}_n\right)^{\mathrm{T}} \right)$$

Maximum Likelihood and Least Squares

- Likelihood를 최대화 하는 $\mathbf{w}$를 행렬을 통해서 나타내면 다음과 같습니다.
-

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$$

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

- 같은 방법으로 likelihood를 최대화하는 β를 구하면 다음과 같습니다.
-

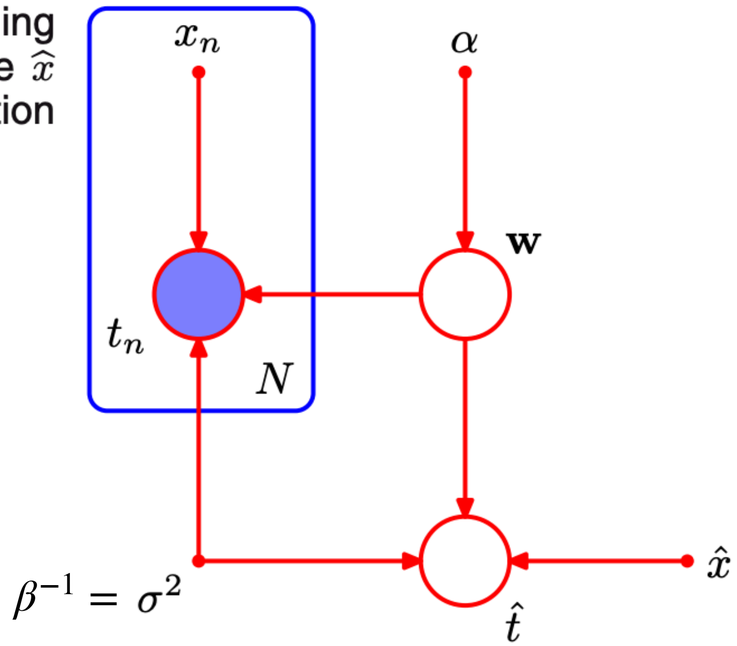
$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2$$

Bayesian
Linear
Regression

Bayesian Linear Regression

- Bayesian linear regression에서는 weight $\mathbf{w}$를 고정된 값으로 두지 않고 random variable로 설정합니다.
- Weight의 prior를 설정하면 데이터가 들어왔을 때 likelihood function과 곱하여 posterior를 구할 수 있습니다.
- 이렇게 구한 posterior를 통해 새로운 입력 샘플이 들어왔을 때 출력의 predictive distribution을 구할 수 있습니다.

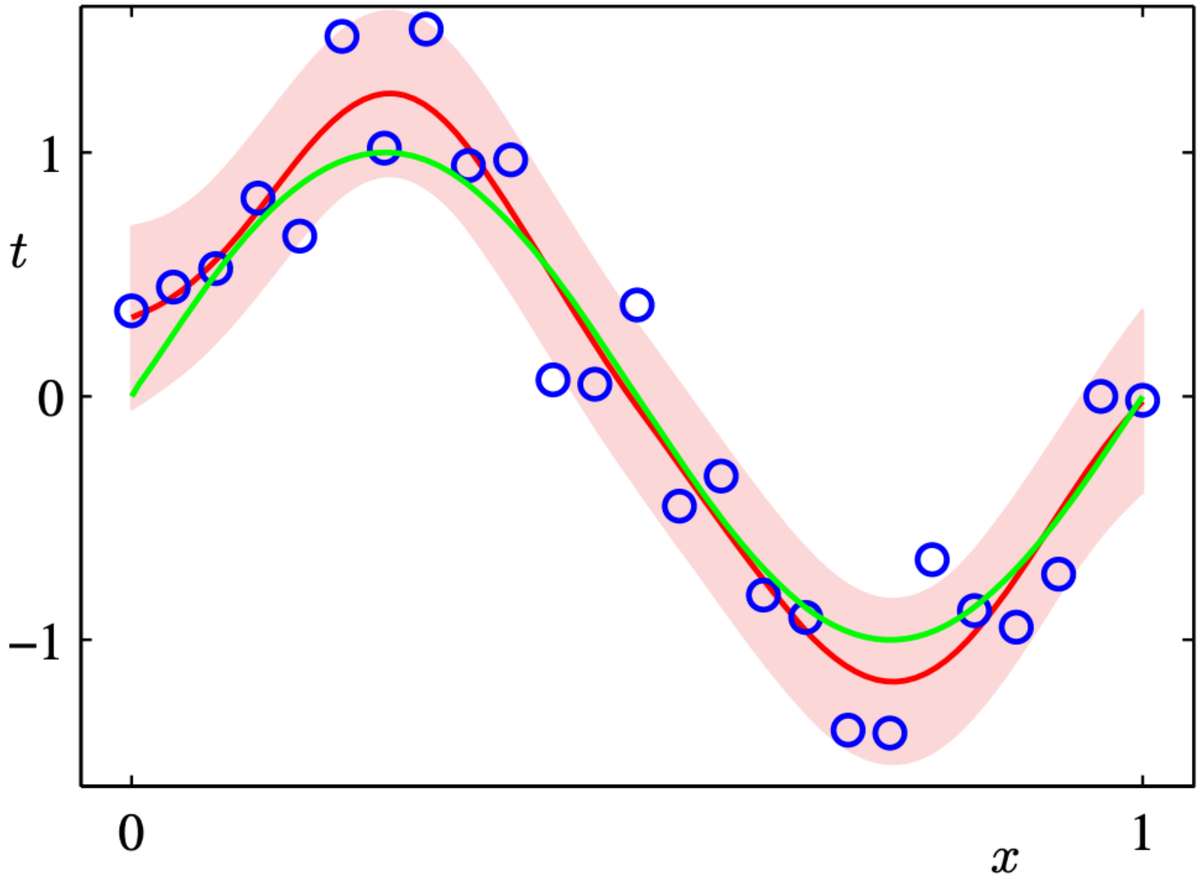
Figure 8.7 The polynomial regression model, corresponding to Figure 8.6, showing also a new input value $\widehat{x}$ together with the corresponding model prediction $\widehat{t}$.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.365

Bayesian Linear Regression

- Predictive distribution



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.157

Bayesian Linear Regression

- 기존 머신러닝에서는 linear regression, Gaussian process, probabilistic PCA, linear dynamical system 등 linear Gaussian model을 이용한 많은 모델들이 있습니다.

- Linear Gaussian model을 이용하면 marginal distribution이나 posterior distribution을 구하기가 용이하기 때문입니다.

- 다음 공식들은 random variable $\mathbf{y}$가 random variable $\mathbf{x}$의 linear transform을 mean으로 하는 Gaussian distribution일 때, marginal distribution과 posterior를 closed form으로 계산한 식입니다.

- 공식을 외울 필요는 없지만 model이 linear Gaussian인 경우 다음과 같이 유용하게 전개됨을 알고 있어야 합니다.

Bayesian Linear Regression

Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{2.113}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \tag{2.114}$$

the marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \tag{2.115}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \tag{2.116}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \tag{2.117}$$

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.93

Bayesian Linear Regression Parameter Distribution

- Weight $\mathbf{w} = (w_0, w_1, \ldots, w_D)^T$ 가 random variable이라 하고 prior로 mean $\mathbf{m}_0$과 covariance matrix $\mathbf{S}_0$인 gaussian distribution을 갖는다 가정합니다.
-
$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w} \,|\, \mathbf{m}_0, \mathbf{S}_0\right)$$
- 데이터 $\mathbf{t} = (t_1, \ldots, t_D)^T$가 주어진 상황에서 posterior는 식(2.116)에 의해 다음과 같이 유도됩니다.
-
$$p(\mathbf{w} \,|\, \mathbf{t}) = \mathcal{N}\left(\mathbf{w} \,|\, \mathbf{m}_N, \mathbf{S}_N\right)$$

where,

$$\mathbf{m}_N = \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}\right)$$
$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$$

Bayesian Linear Regression Parameter Distribution

- prior가 다음과 같이 zero mean과 diagonal covariance matrix를 갖는 경우를 생각해봅시다.
-
$$p(\mathbf{w}\,|\,\alpha) = \mathcal{N}\left(\mathbf{w}\,|\,\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$$

- 이 경우 posterior를 좀 더 간단한 식으로 유도할 수 있습니다.
-
$$p(\mathbf{w}\,|\,\mathbf{t}) = \mathcal{N}\left(\mathbf{w}\,|\,\mathbf{m}_N, \mathbf{S}_N\right)$$

$$\text{where,}\quad \mathbf{m}_N = \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

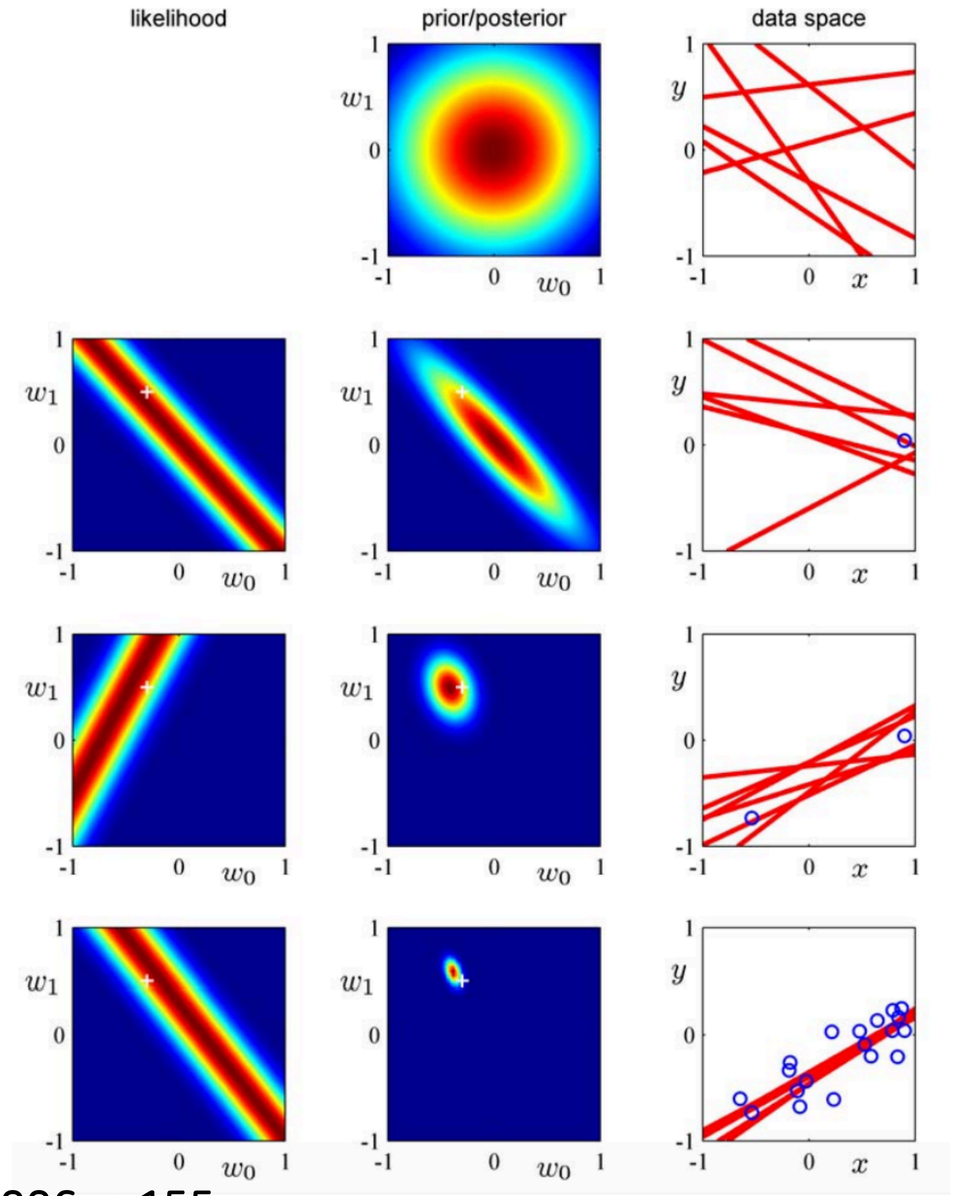
$$\mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$$

- 앞에서 얻은 posterior에 log를 씌우고 weight $\mathbf{w}$에 무관한 항들을 constant로 두면 다음 식과 같이 됩니다.
-
$$\ln p(\mathbf{w}\,|\,\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2 - \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \text{const}.$$

- Posterior distribution을 최대화하는 것은 L2 regularzation이 추가된 sum-of-squares error를 최소화하는 것으로 생각할 수 있습니다.

Bayesian Linear Regression Parameter Distribution

- 가장 위 줄은 데이터가 주어지지 않았을 때 prior의 분포와 prior에서 샘플링한 weight가 그리는 함수들을 나타냅니다.
- 점차 데이터가 주어짐에 따라 (data space에서의 작은 파란 원) likelihood function과 prior를 곱하여 posterior distribution을 업데이트합니다.
- 이에 따라 점점 posterior의 범위는 좁아지며 posterior로부터 샘플링한 weight가 그리는 data space상의 함수들은 데이터 주위를 지나는 것을 관찰할 수 있습니다.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.155

Bayesian Linear Regression Predictive Distribution

- Posterior를 이용해서 새로운 데이터 $\mathbf{x}$가 주어졌을 때의 prediction t의 distribution을 다음과 같이 구할 수 있습니다.
-

$$p(t \mid \mathbf{t}, \alpha, \beta) = \int p(t \mid \mathbf{w}, \beta) p(\mathbf{w} \mid \mathbf{t}, \alpha, \beta) \mathrm{d}\mathbf{w}$$

- 직관적 이해를 돕기 위해 이 수식을 expectation식으로 나타내면 다음과 같습니다.
-

$$p(t \mid \mathbf{t}, \alpha, \beta) = \mathbb{E}_{p(\mathbf{w} \mid \mathbf{t}, \alpha, \beta)}[p(t \mid \mathbf{w}, \beta)]$$

- 또는 샘플링에 의한 근사로 나타내면 다음과 같이 됩니다.
-

$$p(t \mid \mathbf{t}, \alpha, \beta) = \sum_{l=1}^{L} p(t \mid \mathbf{w}^l, \beta), \ \mathbf{w}^l \text{ is a sample from } p(\mathbf{w} \mid \mathbf{t}, \alpha, \beta)$$

Bayesian Linear Regression Predictive Distribution

- Predictive distribution의 계산은 식 (2.115)에 의해 다음과 같이 유도됩니다.
-

$$p(t \mid \mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}\left(t \mid \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})\right)$$

$$\text{where } \sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

- $\sigma_N^2(\mathbf{x})$의 첫번째 항은 데이터 자체의 노이즈를 말하고, 두번째 항은 parameter $\mathbf{w}$와 연관되어지는 모델의 불확실성(uncertainty)를 나타냅니다.

Bayesian Linear Regression Predictive Distribution

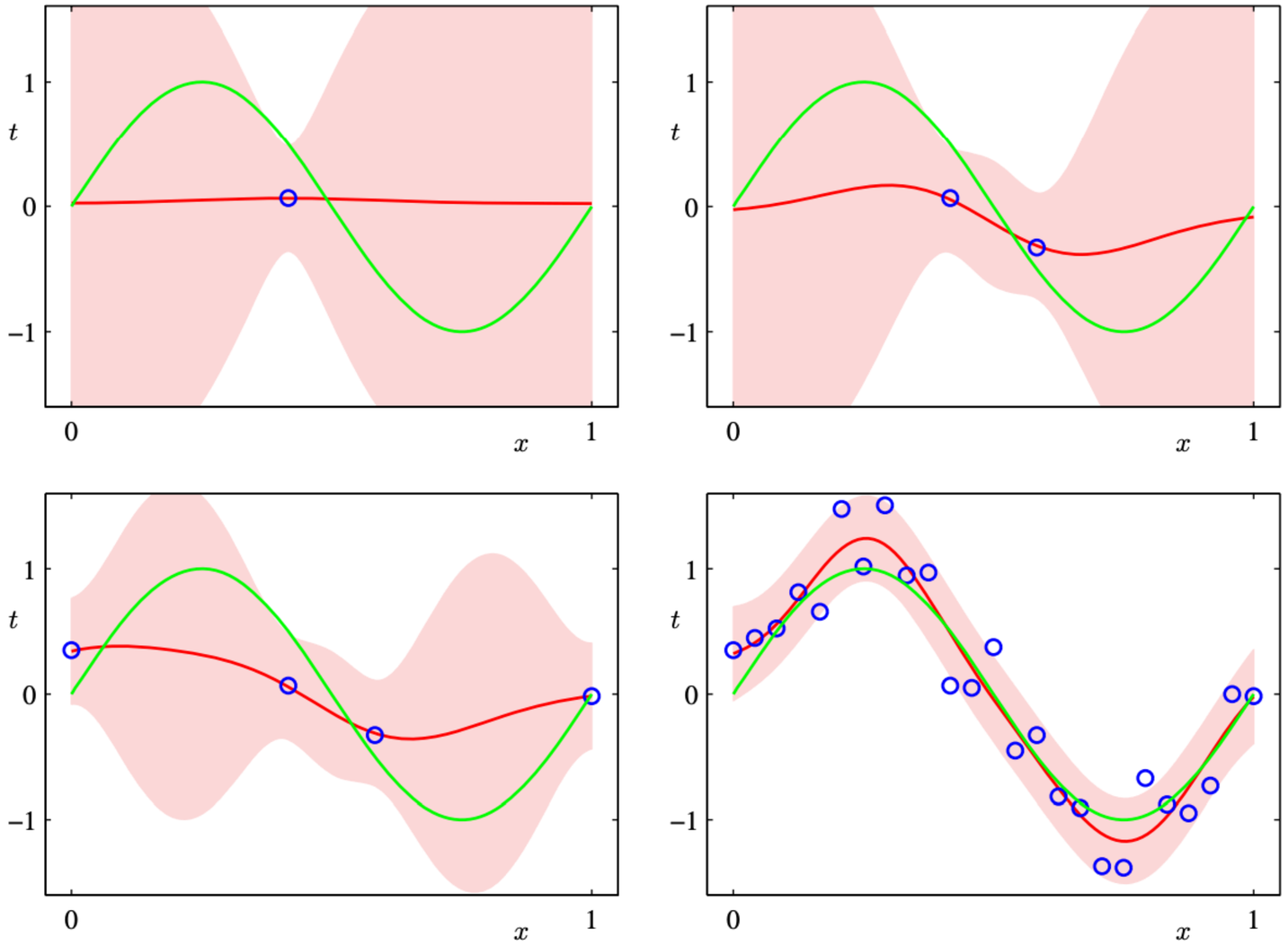


Figure 3.8 Examples of the predictive distribution (3.58) for a model consisting of 9 Gaussian basis functions of the form (3.4) using the synthetic sinusoidal data set of Section 1.1. See the text for a detailed discussion.

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.157

Bayesian Linear Regression Predictive Distribution

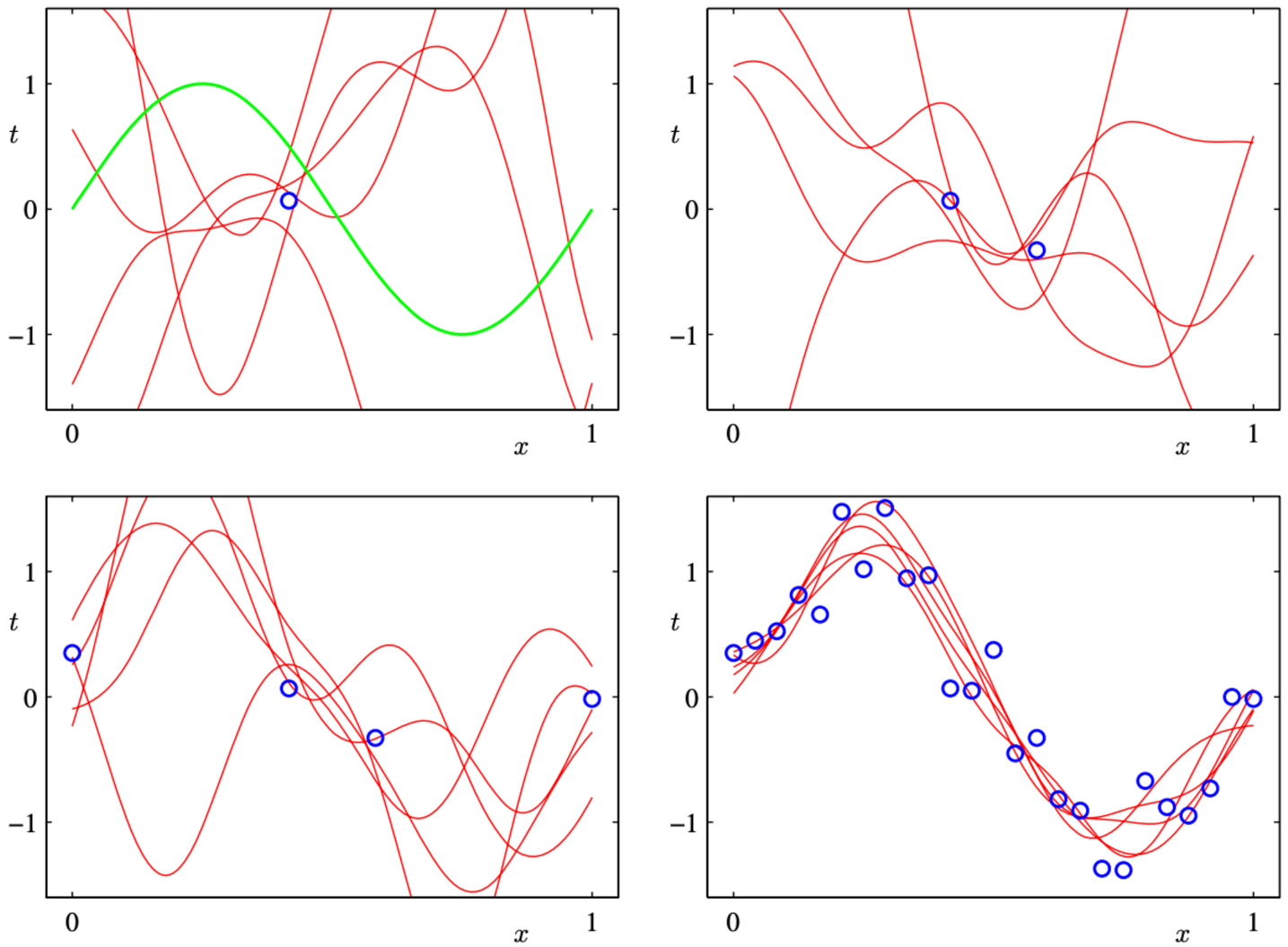


Figure 3.9 Plots of the function $y(x, \mathbf{w})$ using samples from the posterior distributions over $\mathbf{w}$ corresponding to the plots in Figure 3.8.

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.158

Bayesian Linear Regression Marginal Likelihood

- 모델의 parameter인 α와 β를 정하기 위해 Maximum Likelihood 방법을 사용합니다.
- 이를 위해 α와 β가 condition으로 주어졌을 때 트레이닝 데이터의 likelihood를 구해야 합니다.
- weight $\mathbf{w}$가 고정된 값이 아니라 distribution으로 주어져 있으므로 다음과 같이 $\mathbf{w}$에 대해 적분하여 marginal likelihood를 구합니다.
-

$$p(\mathbf{t}\,|\,\alpha,\beta) = \int p(\mathbf{t}\,|\,\mathbf{w},\beta)p(\mathbf{w}\,|\,\alpha)\mathrm{d}\mathbf{w}$$

- 위 식은 expectation식으로 다음과 같이 쓸 수 있습니다.
-

$$p(\mathbf{t}\,|\,\alpha,\beta) = \mathbb{E}_{p(\mathbf{w}|\alpha)}[p(\mathbf{t}\,|\,\mathbf{w},\beta)]$$

- 위 식은 prior에서 샘플링한 weight로 구한 dataset의 likelihood의 기댓값이 됩니다.

Bayesian Linear Regression Marginal Likelihood

- 앞서 weight $\mathbf{w}$와 data precision β가 주어졌을 때 dataset $\mathbf{t}$의 likelihood $p(\mathbf{t}\,|\,\mathbf{w},\beta)$와, weight distribution의 precision α가 주어졌을 때 $\mathbf{w}$의 $p(\mathbf{w}\,|\,\alpha)$는 다음과 같이 정했습니다.
-
$$p(\mathbf{t}\,|\,\mathbf{w},\beta)=\prod_{n=1}^{N}\mathcal{N}\left(t_n\,|\,\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right),\beta^{-1}\right),\ p(\mathbf{w}\,|\,\alpha)=\mathcal{N}\left(\mathbf{w}\,|\,\mathbf{0},\alpha^{-1}\mathbf{I}\right)$$
- 위 식을 대입해 (2.115)를 이용하여 dataset $\mathbf{t}$의 marginal likelihood를 구하면 다음과 같습니다.
-
$$p(\mathbf{t}\,|\,\alpha,\beta)=\left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2}\int\exp\{-E(\mathbf{w})\}\mathrm{d}\mathbf{w}$$

$$E(\mathbf{w})=\beta E_D(\mathbf{w})+\alpha E_W(\mathbf{w})$$

where

$$=\frac{\beta}{2}\|\mathbf{t}-\boldsymbol{\Phi}\mathbf{w}\|^2+\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

data error　　weight penalty

-

상세한 유도과정은 Pattern Recognition And Machine Learning 2006, p.166-167 참조

Bayesian Linear Regression Marginal Likelihood

- 적분식을 계산하고 log marginal likelihood를 구하면 다음과 같이 유도됩니다.
-

$$\ln p(\mathbf{t} \mid \alpha, \beta) = \frac{M}{2} \ln \alpha + \frac{N}{2} \ln \beta - E\left(\mathbf{m}_N\right) - \frac{1}{2} \ln |\mathbf{A}| - \frac{N}{2} \ln(2\pi)$$

$$\text{where } E\left(\mathbf{m}_N\right) = \frac{\beta}{2} \left\| \mathbf{t} - \mathbf{\Phi} \mathbf{m}_N \right\|^2 + \frac{\alpha}{2} \mathbf{m}_N^{\mathrm{T}} \mathbf{m}_N \text{ and } \mathbf{A} = \alpha \mathbf{I} + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}$$

상세한 유도과정은 Pattern Recognition And Machine Learning 2006, p.166-167 참조

Bayesian Linear Regression Marginal Likelihood

- Marginal Likelihood 값을 계산하면 데이터에 적합한 polynomial의 차수 즉, 모델 복잡도를 알 수 있습니다.

- 이는 차수가 증가함에 따라 과적합(over-fitting)되어 RMS error가 줄어드는 least square를 이용한 regression방식과는 다른 모습입니다.

Figure 3.14 Plot of the model evidence versus the order M, for the polynomial regression model, showing that the evidence favours the model with $M = 3$.

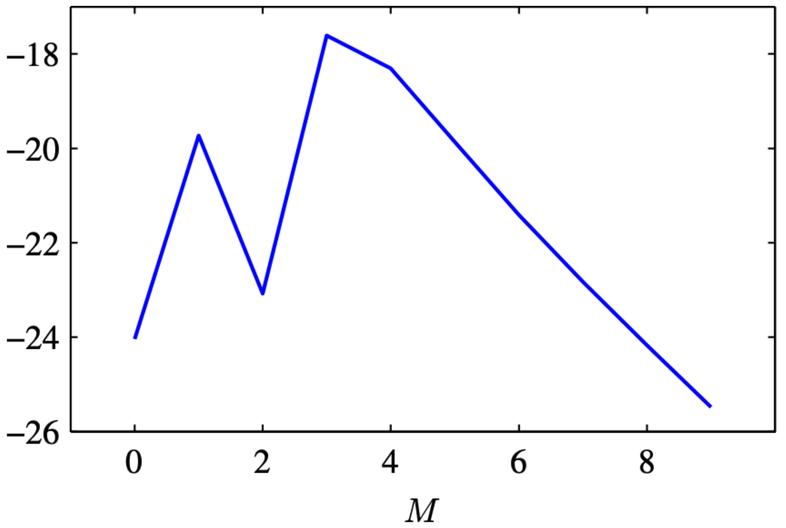
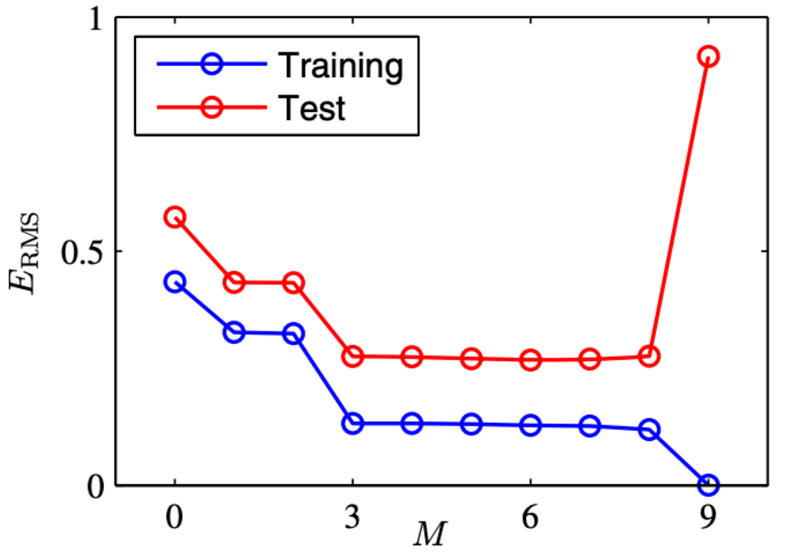


Figure 1.5 Graphs of the root-mean-square error, defined by (1.3), evaluated on the training set and on an independent test set for various values of M.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, 상 p.168 하 p.8

Bayesian Linear Regression Maximum Marginal Likelihood

- 앞서 구한 marginal likelihood 식을 최대화 하는 α와 β를 구하면 다음과 같습니다.
-

$$\alpha = \frac{\gamma}{\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N}$$

$$\frac{1}{\beta} = \frac{1}{N-\gamma}\sum_{n=1}^{N}\left\{t_n - \mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2$$

$$\gamma = \sum_i \frac{\lambda_i}{\alpha+\lambda_i}, \text{ where } \lambda_i \text{ are eigenvalues of } \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$$

- α와 β는 closed form 으로 구해지지 않으며 초기화 후 iteration을 통해 구할 수 있습니다.
- iteration시 α와 β뿐 아니라, 이를 얻기 위해 필요한 weight posterior의 mean $\mathbf{m}_N$, covariance matrix $\mathbf{S}_N$도 함께 변경됩니다.

상세한 유도과정은 Pattern Recognition And Machine Learning 2006, p.168-169 참조

Fully Bayesian Linear Regression

- 앞선 Bayesian linear regression모델은 weight 분포의 파라메터 precision α와 data의 precision β를 고정된 값으로 두었습니다.

- 이보다 한걸음 더 나아가 α와 β값 또한 고정시키지 않고 확률분포로 정하여 더욱 유연한 모델을 만들어 볼 수 있습니다.

- Scikit-learn의 sklearn.linear_model.BayesianRidge class에서 이 모델을 구현하고 있습니다.

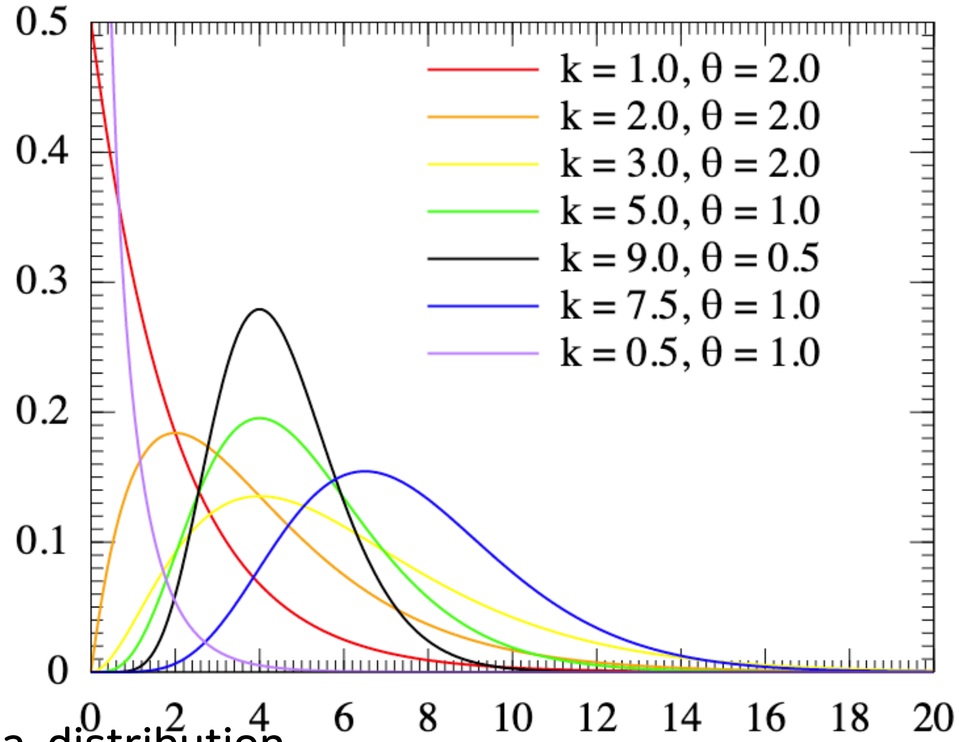
- α와 β값은 precision을 나타내며 이러한 종류의 값은 conjugate prior로 gamma distribution을 갖습니다.

Fully Bayesian Linear Regression (gamma distribution)

- Gamma distribution은 variance나 precision에 대한 conjugate prior입니다.

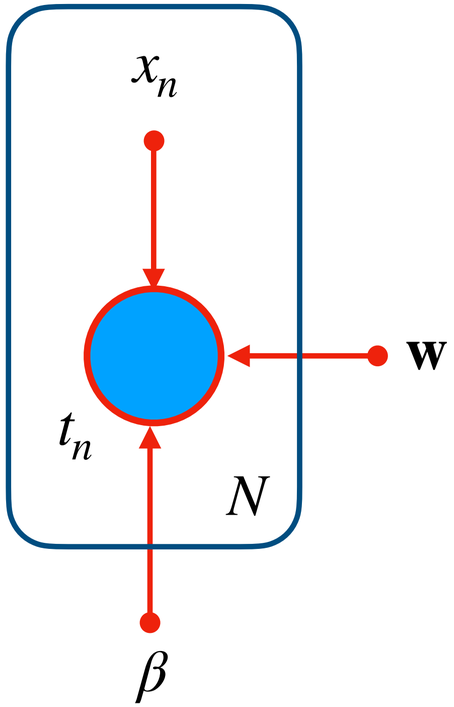
$$p(x) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{\alpha}{\theta}}$$

- parameter로는 shape을 결정하는 k와 scale을 결정하는 θ가 있습니다.

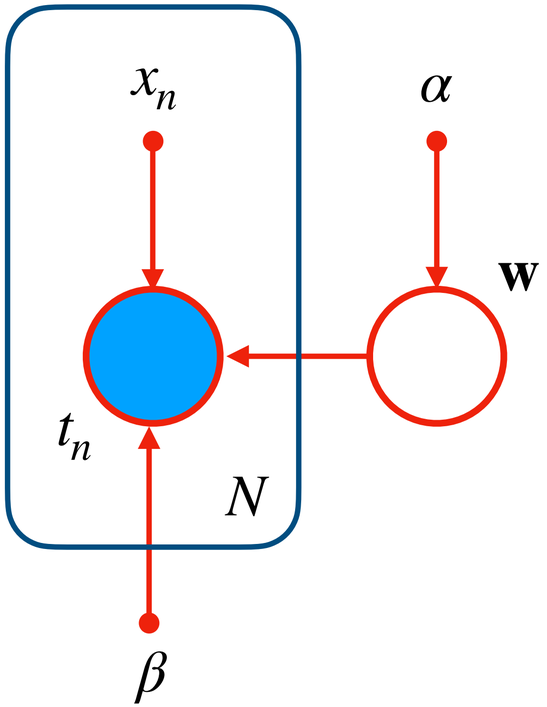


https://en.wikipedia.org/wiki/Gamma_distribution

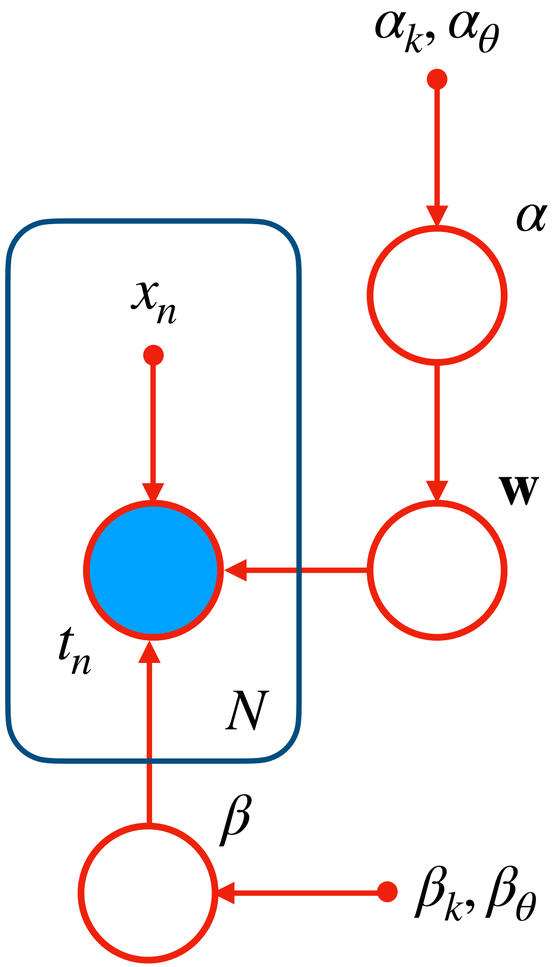
Linear Regression, Bayesian, Fully Bayesian



Linear Regression
(point estimation)



Bayesian Linear Regression
(empirical Bayes)



Fully-Bayesian
Linear Regression

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.365

딥러닝 에스프레소

딥러닝을 위한 베이지안 통계 Day2

K-means, GMM, PCA, PPCA, Auto-Encoder

2022

멀티캠퍼스

박수철

K-means

K-means

- Clustering은 unsupervised learning으로 target value가 없는 dataset이 주어집니다.
- K-means는 machine learning의 가장 기초적인 clustering 알고리즘입니다.
- K개의 중심(centroid)을 찾고, 데이터들은 가장 가까운 중심에 해당하는 cluster에 대응시킵니다.
- K개의 centroid들은 각 cluster들의 평균 지점(mean)에 해당하기 때문에 K-means라는 이름이 붙었습니다.
- Dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$에 대해 K개로 clustering한다고 합시다. 각 cluster의 mean이 μ_k이고 data point $\mathbf{x}_n$이 k번째 cluster에 속함을 indicator variable $r_{nk} \in \{0,1\}$로 표현하여 다음과 같이 objective function을 쓸 수 있습니다.
-

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \parallel \mathbf{x}_n - \boldsymbol{\mu}_k \parallel^2$$

K-means

- Objective function J를 최소화하기 위한 r_{nk}는 각 data point index n에 대해 $\| \mathbf{x}_n - \boldsymbol{\mu}_k \|^2$ 항이 최소가 되는 cluster index k를 대응시키는 방법으로 구할 수 있습니다.

- $$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \| \mathbf{x}_n - \boldsymbol{\mu}_j \|^2 \\ 0 & \text{otherwise} \end{cases}$$

- data point $\mathbf{x}_n$에 대응하는 cluster를 추정한다, 예상한다라는 맥락에서 위 과정을 expectation step이라 부릅니다.

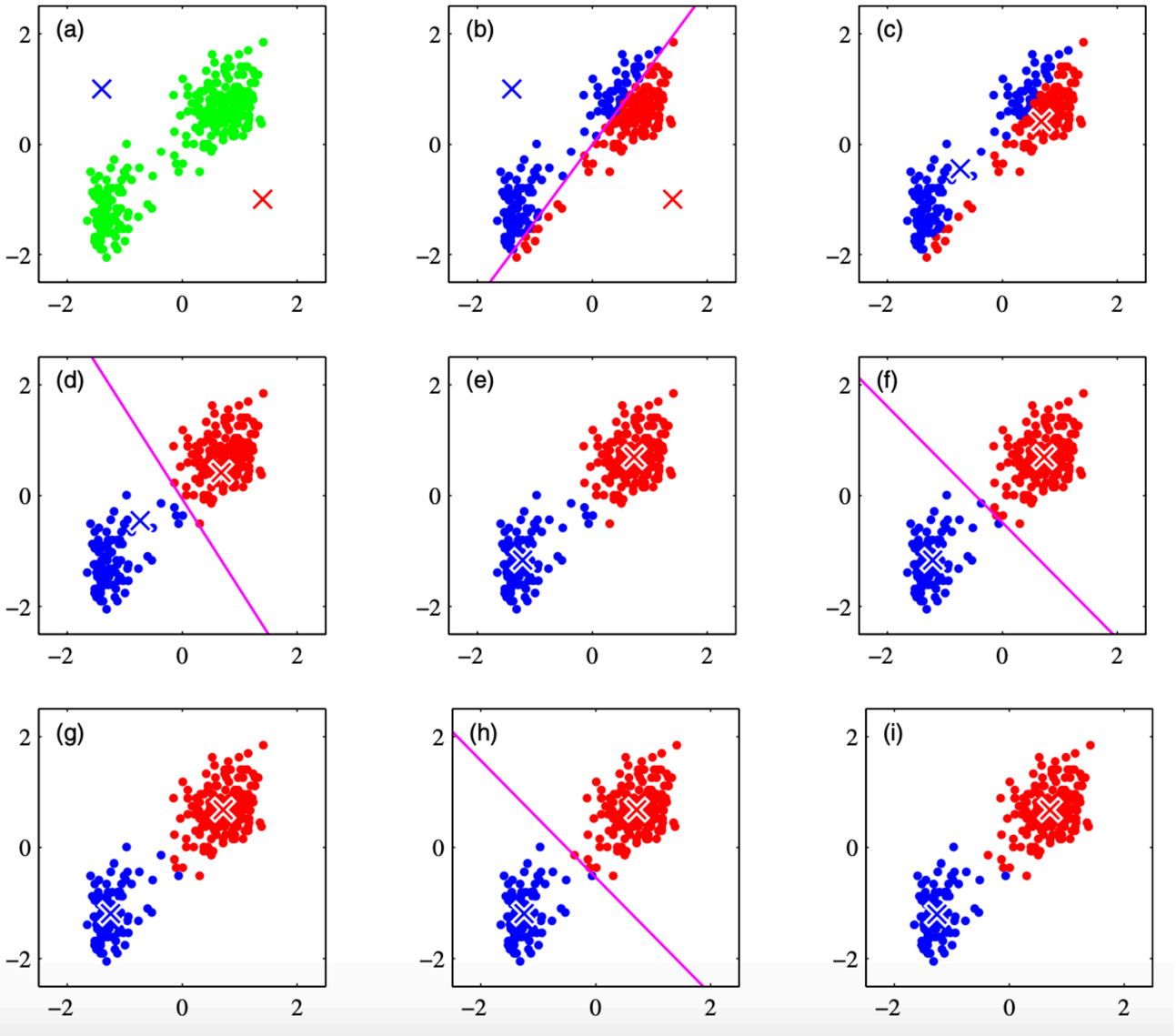
- Objective function J를 최소화하는 $\boldsymbol{\mu}_k$는 J를 미분한 식을 0으로 두고 $\boldsymbol{\mu}_k$에 대해 식을 전개하여 얻을 수 있습니다.

- $$\boldsymbol{\mu}_k = \frac{\sum_n \boldsymbol{r_{nk}} \mathbf{x_n}}{\sum_n \boldsymbol{r_{nk}}}$$

- Expectation step에서 추정한 r_{nk}값들을 근거로 J를 최대화하기 위한 parameter $\boldsymbol{\mu}_k$를 구한다는 맥락에서 위 과정을maximization step이라고 부릅니다.

- Expectation과 maximization step을 번갈아 가면서 작동시키는 것을 EM 알고리즘이라 부릅니다.

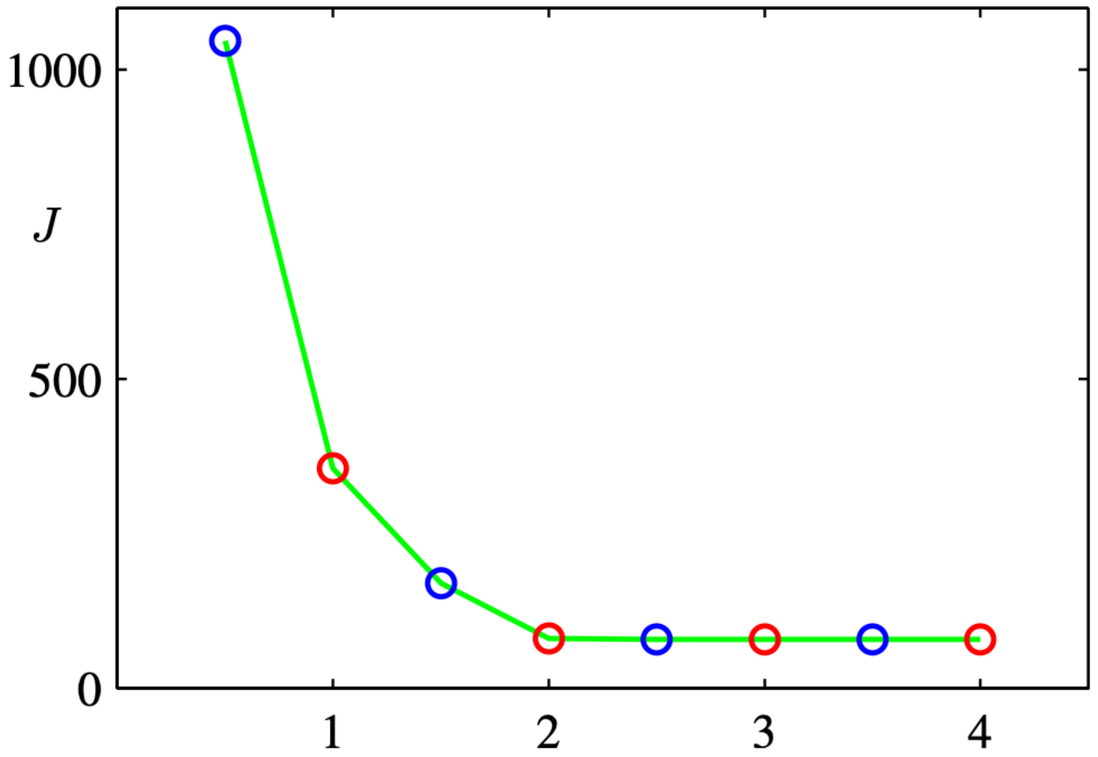
K-means



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.426

K-means

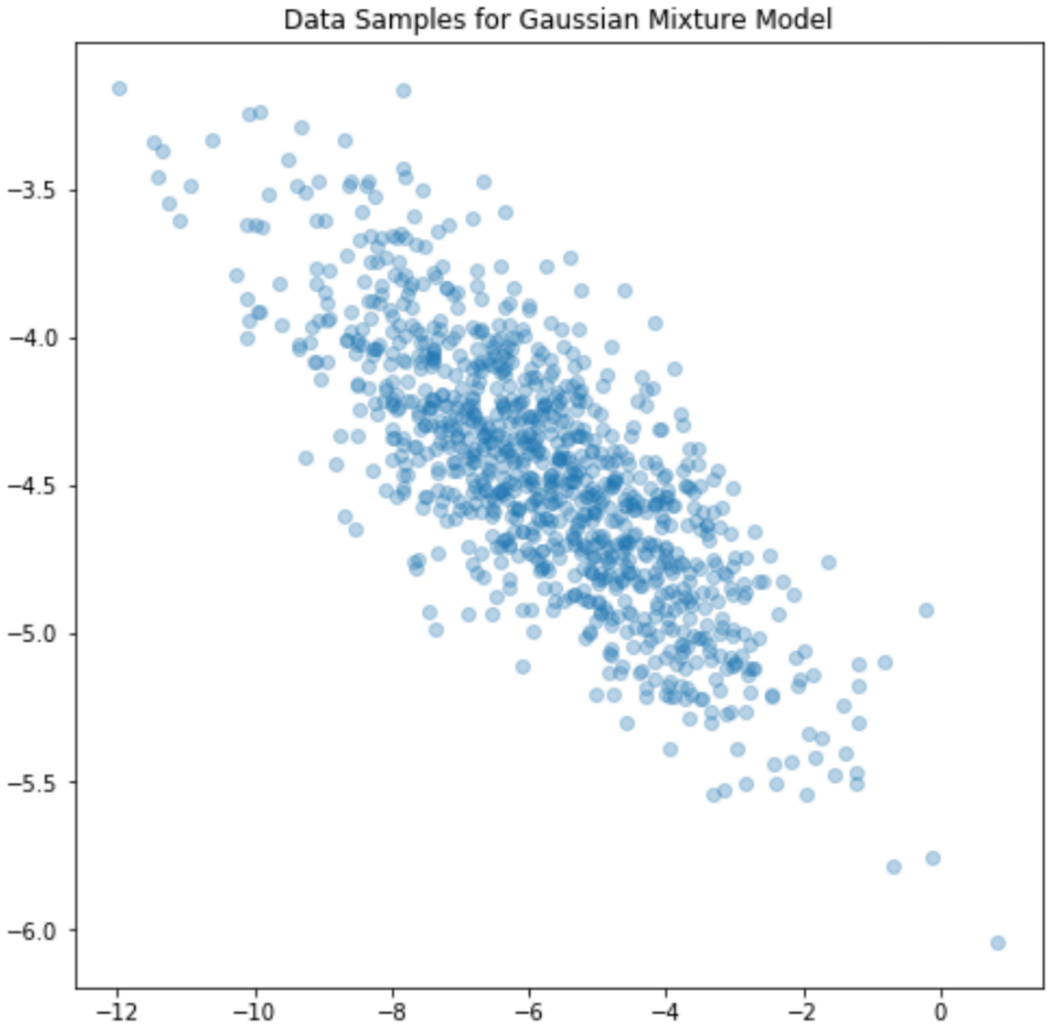
Figure 9.2 Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.426

Gaussian Mixture Models

Gaussian Models



Gaussian Models

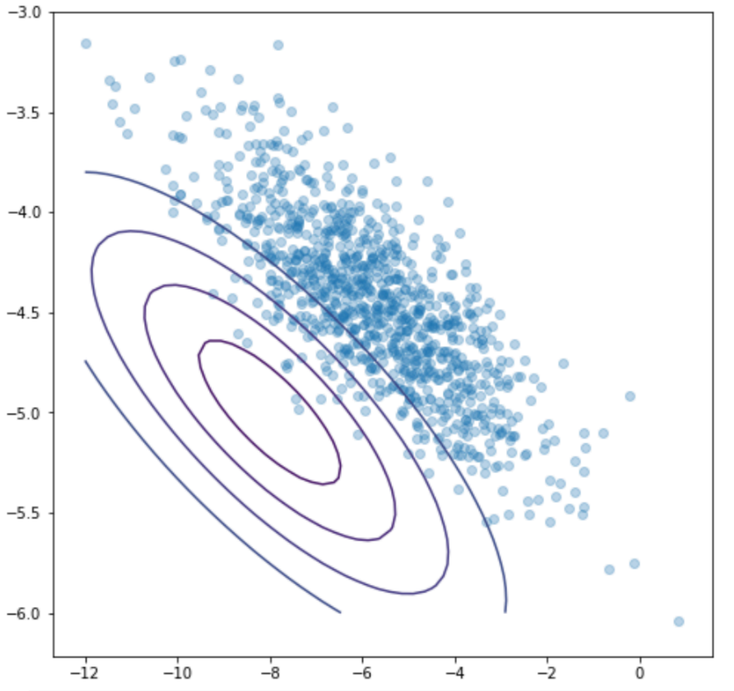
- Single multivariate Gaussian distribution을 이용해 dataset의 distribution을 모델링하고자 합니다.

- $$p_\theta(x) = N(x \mid \mu, \Sigma)$$
$$= \frac{1}{\sqrt{(2\pi)^D \mid \Sigma \mid}} \exp\left[-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T\right]$$

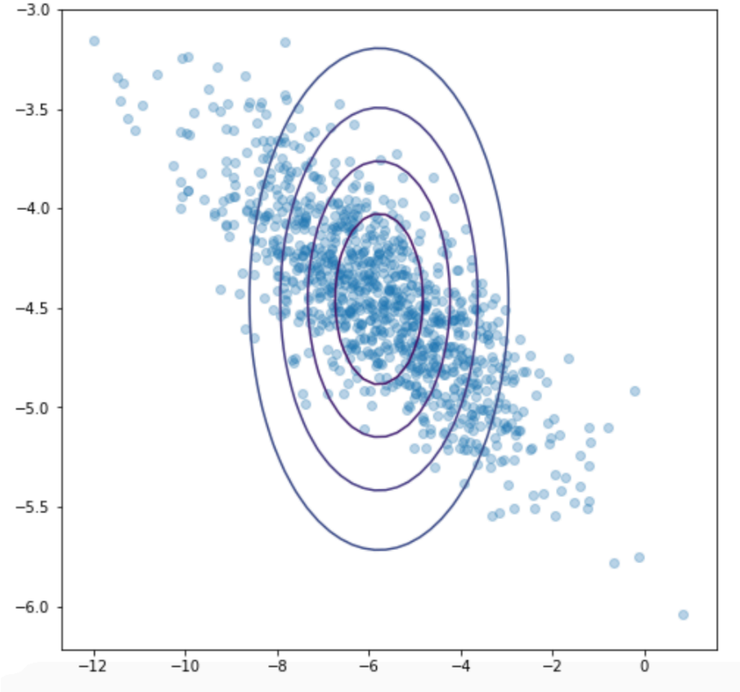
- 이를 위해서 maximum likelihood 방식을 이용합니다. 즉, dataset의 likelihood를 최대화하는 parameters $\theta = \{\mu, \Sigma\}$를 구합니다.

- $$\arg\max_\theta p_\theta(X), \textit{ where } X = \{x_1, x_2, \ldots, x_N\}$$

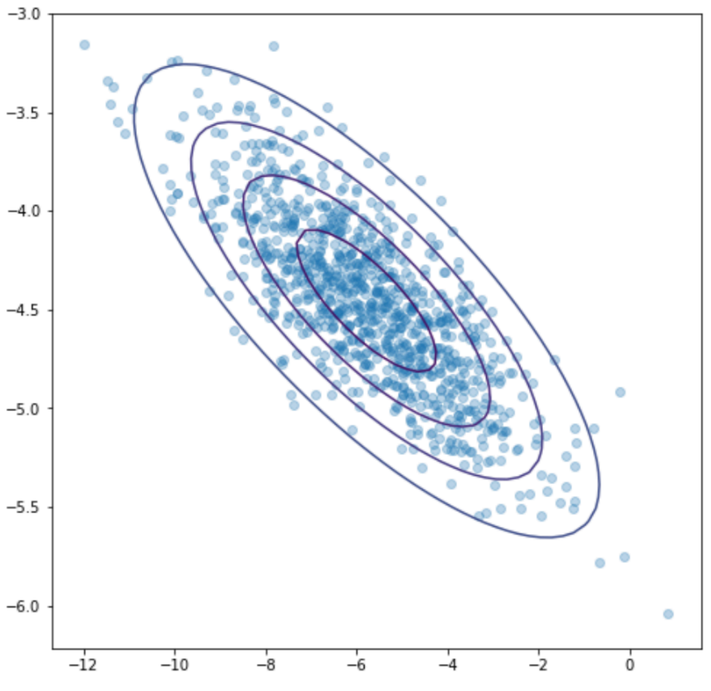
Gaussian Models



낮은 likelihood를 갖는 μ, Σ가 맞춰진 상황



likelihood가 최대가 되도록 μ가 맞춰졌으나 Σ가 적합하지 않은 상황



likelihood가 최대가 되도록 μ와 Σ가 맞춰진 상황

Gaussian Models

- Distribution $p_\theta(x)$에서 data samples X 전체의 likelihood는 다음과 같습니다. 이 때, 각 samples들은 I.I.D. (independent and identically distributed)로 가정합니다.
-
$$p(X) = \prod_{n=1}^{N} p(x_n),\ X = \{x_1, x_2, \ldots, x_N\}$$
- 위 식에 log를 씌워 Dataset 전체의 log-likelihood를 구하면 다음과 같습니다.
-
$$\log p_\theta(X) = \log \prod_{n=1}^{N} p_\theta(x_n) = \sum_{n=1}^{N} \log p_\theta(x_n)$$
- Single multivariate Gaussian distribution에서 data point 하나의 log-likelihood는 다음과 같습니다.
-
$$\log p_\theta(x_n) = -\frac{D}{2}\log 2\pi - \log|\Sigma| - \frac{1}{2}(x_n - \mu)\Sigma^{-1}(x_n - \mu)^T$$
- 따라서 dataset 전체의 log-likelihood는 다음과 같이 계산됩니다.
-
$$\log p_\theta(X) = \sum_{n=1}^{N} -\frac{D}{2}log2\pi - log|\Sigma| - \frac{1}{2}(x_n - \mu)\Sigma^{-1}(x_n - \mu)^T$$

Gaussian Models

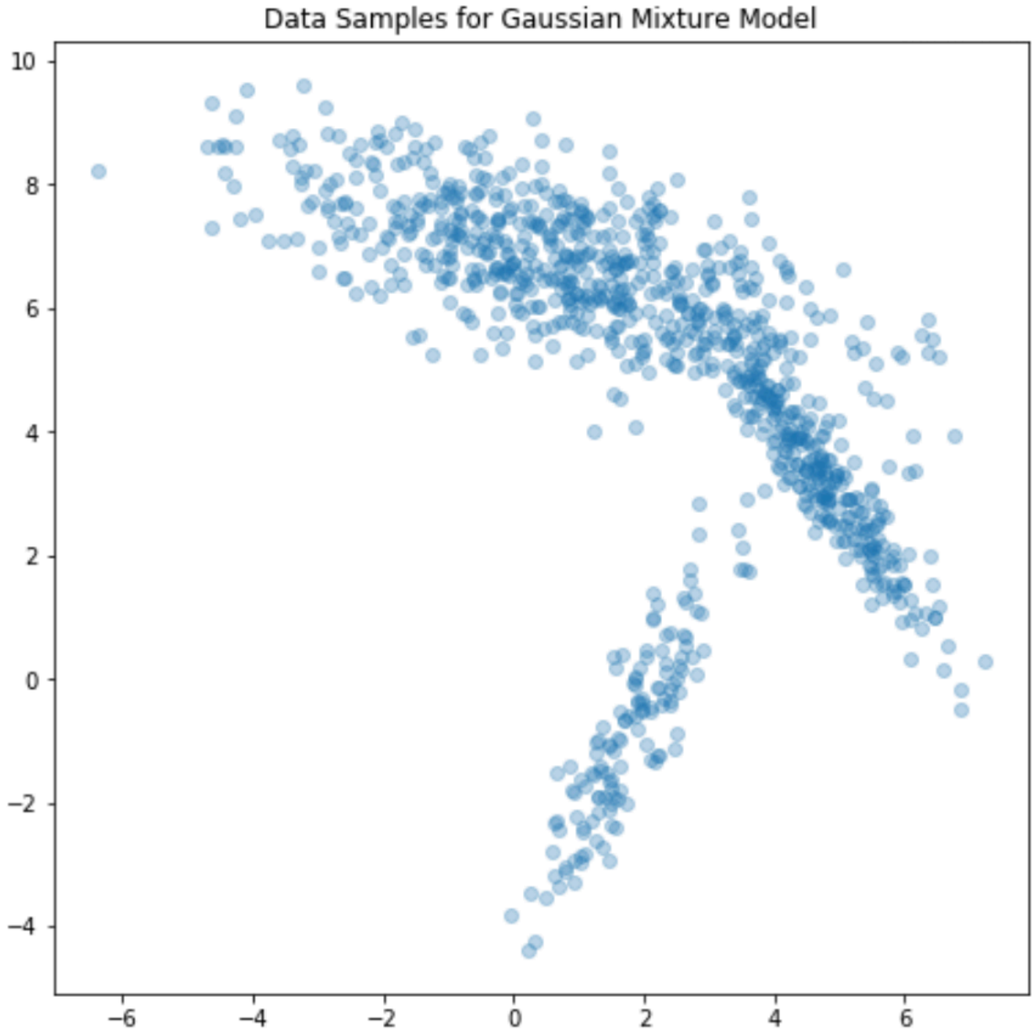
- $\log p_\theta(X)$가 최대가 되도록하는 parameters μ, Σ를 다음과 같이 closed form solution을 구할 수 있습니다.
-

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T$$

유도 과정 Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.93-94

Gaussian Mixture Models



Reference: Christopher Bishop, Pattern Recognition and Machine Learning, p.430-441

Gaussian Mixture Models

- 여러 Multivariate Gaussian distributions를 이용해 dataset의 distribution을 모델링하고자 합니다.

- $$p_\theta(x) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)$$

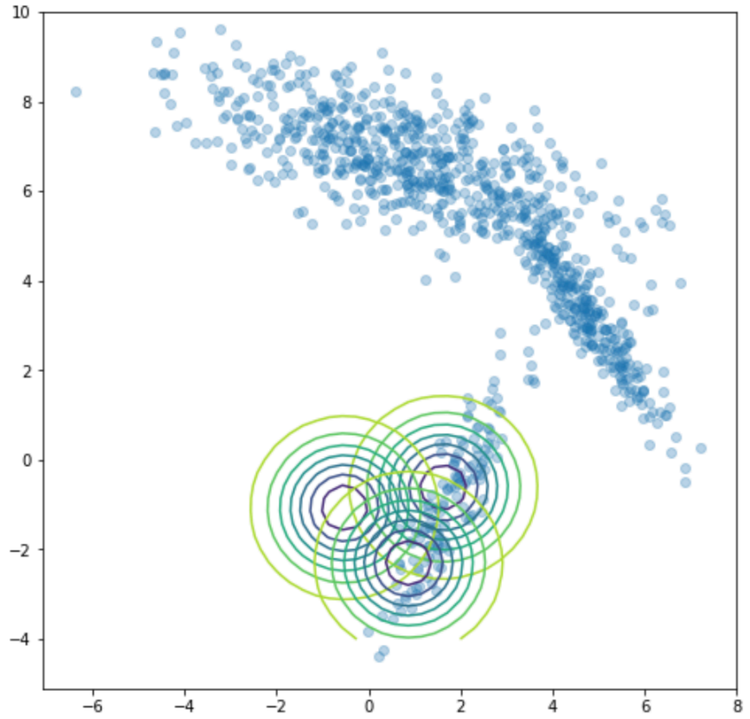
- 위 식에서 K는 Gaussian distribution의 mixture 갯수를 뜻합니다. 각 Gaussian component 별로 parameters인 mixing coefficient π_k와 mean μ_k, covariance matrix Σ_k가 존재합니다.

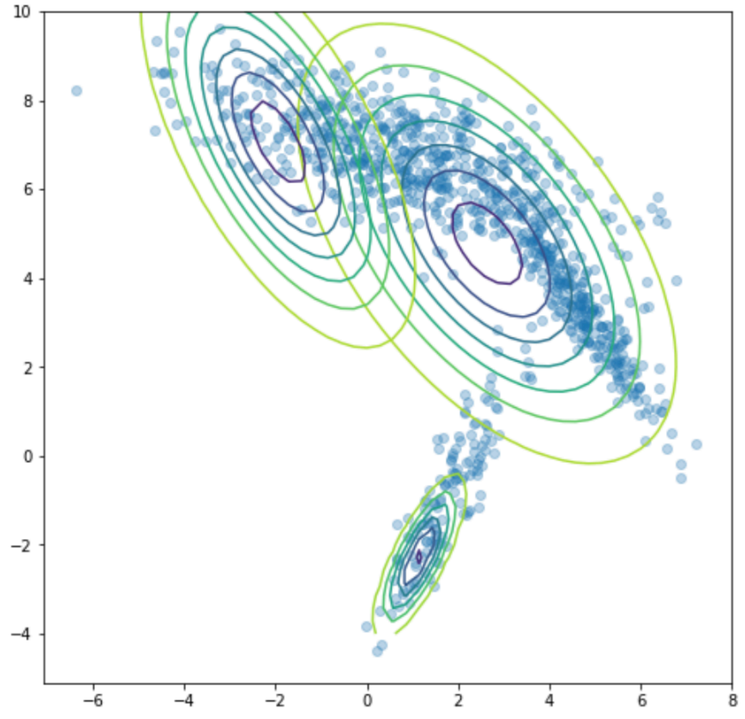
- Maximum likelihood 방식을 이용하여 dataset에 맞는 distribution의 parameters를 찾습니다. 즉, dataset의 likelihood를 최대화하는 parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}$를 구합니다.

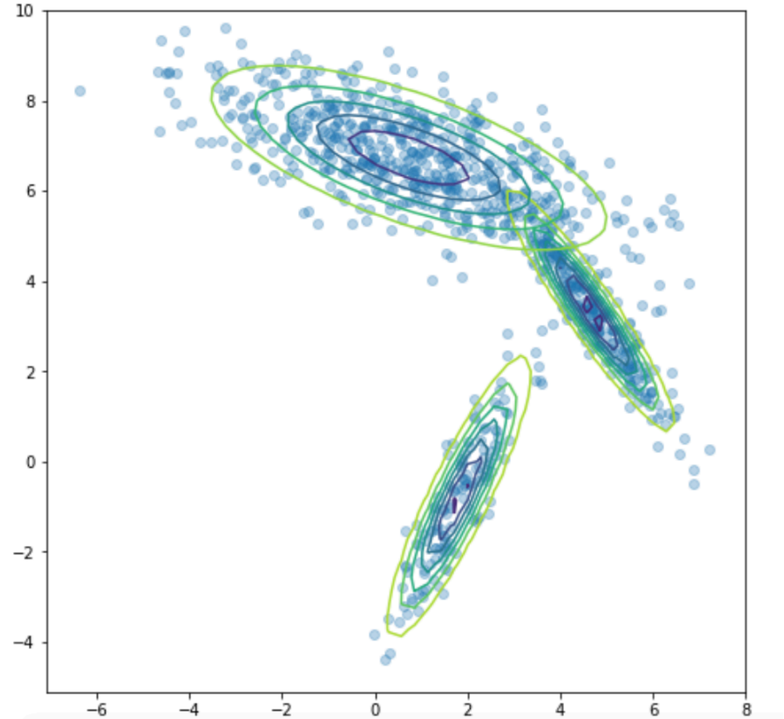
- $$\arg\max_{\theta} p_\theta(X), \; where \; X = \{x_1, x_2, \ldots, x_N\}$$

Gaussian Mixture Models

- dataset의 likelihood가 커지도록 parameters가 업데이트 되는 모습







Gaussian Mixture Models

- Data point 하나는 다음과 같은 두 순서 의해서 생성되었다고 볼 수 있습니다.
  1. K개의 components(Gaussian distributions) 중 하나를 선택
  2. 선택한 component에서 data point 샘플링

- Mixture의 각 component들이 선택될 확률을 categorical distribution으로 나타냅니다.

- $$p_\theta(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

- 이 때, z는 선택된 component를 나타내는 one-hot vector로 표현됩니다. 예를 들어 총 5개의 components가 있고 3번째 component가 선택되었다면 $z = (0,0,1,0,0)^T$를 갖게 됩니다. z_k는 one-hot vector의 k번째 element를 뜻합니다.

- Component가 정해졌을 때, 즉 z가 주어졌을 때 data point의 likelihood는 Gaussian distribution으로 나타냅니다.

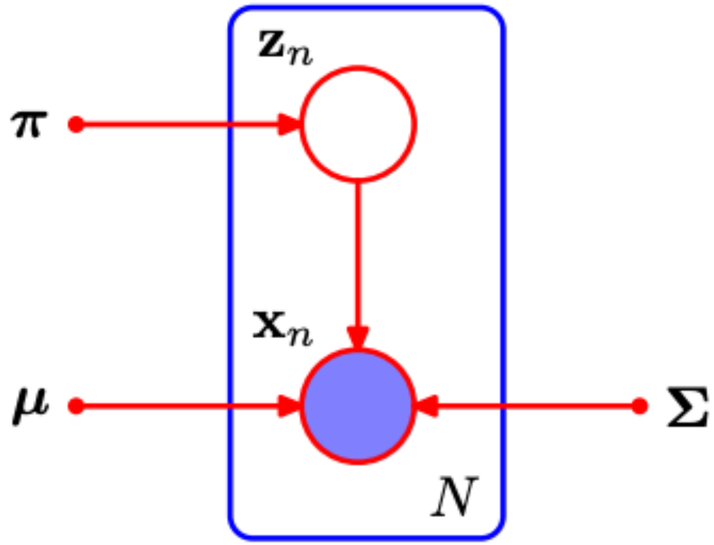
- $$p_\theta(x \mid z) = \prod_{k=1}^{K} N(x \mid \mu_k, \Sigma_k)^{z_k}$$

Gaussian Mixture Models

- z를 특정하지 않은 marginal likelihood는 다음과 같이 구할 수 있습니다.
-

$$p_\theta(x) = \sum_z p_\theta(z)p_\theta(x \mid z) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)$$

- Gaussian mixture models을 graphical representation으로 나타낼 수 있습니다.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433

Gaussian Mixture Models

- Single Gaussian Model의 경우와 마찬가지로 I.I.D. dataset 전체의 likelihood는 다음과 같습니다.
-

$$p_\theta(X) = \prod_{n=1}^{N} p_\theta(x_n),\ X = \{x_1, x_2, \ldots, x_N\}$$

- Gaussian mixture models 경우 log-likelihood는 다음과 같이 전개됩니다.
-

$$\log p_\theta(X) = \log \prod_{n=1}^{N} p_\theta(x_n) = \sum_{n=1}^{N} \log p_\theta(x_n) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k)$$

- Single Gaussian model과 다르게 log-likelihood를 최대화하는 parameters를 closed form으로 구할 수 없습니다.

Gaussian Mixture Models

- log-likelihood가 최대가 되는 mean μ_k을 구해보기 위해 μ_k로 미분하고 이를 0으로 두면 다음과 같습니다.
-

$$0 = \sum_{n=1}^{N} \frac{\pi_k N(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)} \Sigma^{-1}(x_n - \mu_k)$$

- 만약 위 식에서 $\frac{\pi_k N(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)}$부분을 어떤 값으로 고정한다면 μ_k를 구할 수 있습니다.

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) \Sigma^{-1}(x_n - \mu_k)$$

Σ를 양변에 곱하면

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) x_n - \sum_{n=1}^{N} \gamma(z_{nk}) \mu_k$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n, \; where \; N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Gaussian Mixture Models

- 앞에서 고정한 값은 사실 어떤 data point x_n이 주어졌을 때 k번째 component로부터 생성되었을 확률을 나타내는 posterior로 해석할 수 있습니다. GMM에서 이를 responsibility라고 말하기도 합니다.

Prior
$p(z_k = 1)$

Likelihood
$p(x_n \mid z_k = 1)$

$$0 = \sum_{n=1}^{N} \frac{\pi_k N(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)} \Sigma^{-1}(x_n - \mu_k)$$

Marginal Likelihood(Evidence)

$$\sum_{z_j} p(z_j = 1) p(x_n \mid z_j = 1) = p(x_n)$$

Prior x Likelihood / Evidence = Posterior

$$p(z_{nk} = 1 \mid x_n)$$

Gaussian Mixture Models

- posterior를 고정하면 mean μ_k을 비롯해 mixing coefficient π_k와 covariance matrix Σ_k도 closed form으로 구할 수 있습니다.

- $$\pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n,$$

$$\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k)(x_n-\mu_k)^T$$

where $\gamma(z_{nk}) = \sum_{n=1}^{N}\frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j N(x_n|\mu_j,\Sigma_j)}$ and $N_k = \sum_{n=1}^{N}\gamma(z_{nk})$

Gaussian Mixture Models

- Parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}$를 구하고 나면 그에 따라 posterior인 responsibility $\gamma(z_{nk})$값도 변하게 됩니다. 따라서 EM 알고리즘이라 불리는 iteration을 통해 최적화를 진행하게 됩니다.

- Expectation-Maximization Algorithm for GMM

  1. parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}$를 임의의 값으로 초기화 한다.

  2. E-step : responsibility를 구한다.

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)}$$

  3. M-step : responsibility를 고정하고 parameters를 다시 구한다.

$$\pi_k^{new} = \frac{N_k}{N}, \mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n, \Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$where\ N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

  4. 정해진 step을 돌거나, 정해놓은 수렴 기준에 도달할 때 까지 E-step과 M-step을 반복한다.

Gaussian Mixture Models

- 현재 가지고 있는 dataset은 $X = \{x_1, x_2, \ldots, x_N\}$입니다. 이를 incomplete dataset이라고 합니다.

- 모든 X에 대응하는 모든 latent variables $Z = \{z_1, z_2, \ldots, z_N\}$도 가지고 있다면 이를 complete dataset이라고 할 수 있습니다.

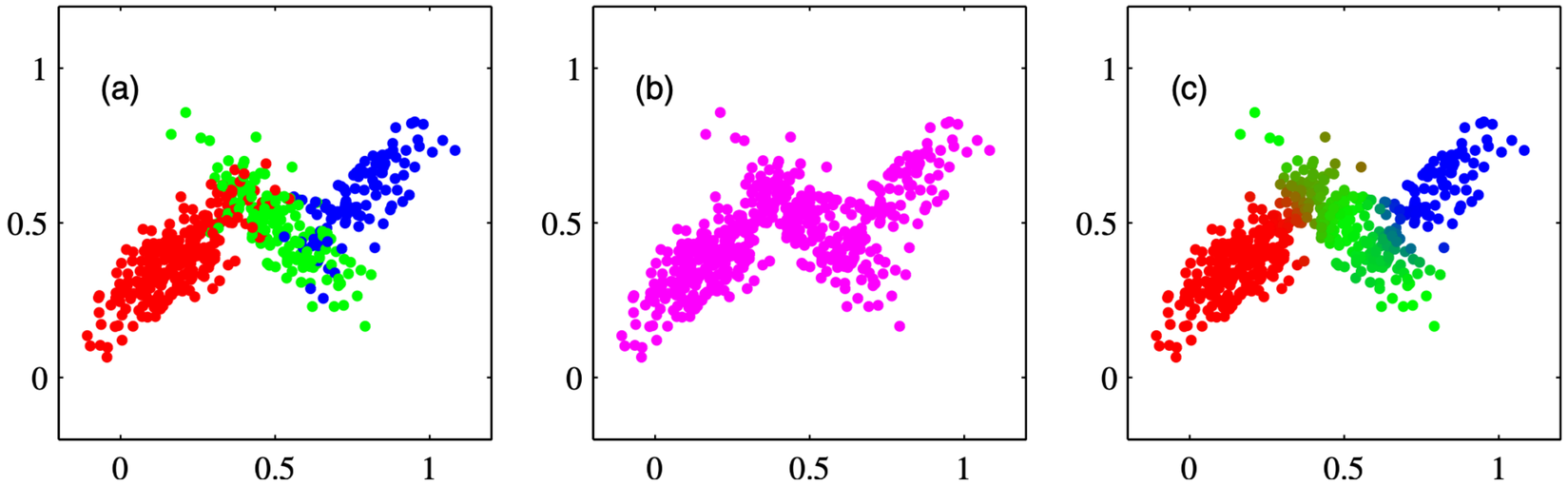


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of $\mathbf{z}$, corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of $\mathbf{z}$ and just plotting the $\mathbf{x}$ values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point $\mathbf{x}_n$, obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433

Gaussian Mixture Models

- Dataset으로 $X = \{x_1, x_2, \ldots, x_N\}$와 $Z = \{z_1, z_2, \ldots, z_N\}$가 주어진 경우 complete data log-likelihood를 다음과 같이 쓸 수 있습니다.
- $$\log p_\theta(X, Z) = \log p_\theta(Z) + \log p_\theta(X \mid Z) = \sum_{n=1}^{N} [\log p_\theta(z_n) + \log p_\theta(x_n \mid z_n)]$$
- $p_\theta(z)$는 categorical distribution, $p_\theta(x \mid z)$는 Gaussian distribution이므로 complete data-log-likelihood를 최대화하는 것은 앞에서 보인 것 처럼 쉽게 전개할 수 있습니다.
- $\log p_\theta(Z)$를 최대화하는 parameter π_k를 구하면 다음과 같습니다.
- $$\pi_k = \frac{N_k}{N}, \text{ where } N_k = \sum_{n=1}^{N} z_{nk}$$
- Z가 주어졌을 때 $\log p_\theta(X \mid Z)$를 최대화하는 parameters μ_k, Σ_k를 구하면 다음과 같습니다.
- $$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} x_n,$$
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$

Gaussian Mixture Models

Posterior(responsibility)가 주어졌을 때
EM알고리즘의 M-step

$$\pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n,$$

$$\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k)(x_n-\mu_k)^T$$

$$\text{where } \gamma(z_{nk}) = \sum_{n=1}^{N}\frac{\pi_k N(x_n\,|\,\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j N(x_n\,|\,\mu_j,\Sigma_j)}$$

$$\text{and } N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

Complete dataset X, Z가 주어졌을 때
complete data log-likelihood의 최대화

$$\pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}z_{nk}x_n,$$

$$\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N}z_{nk}(x_n-\mu_k)(x_n-\mu_k)^T$$

$$\text{where } N_k = \sum_{n=1}^{N}z_{nk}$$

Gaussian Mixture Models

- 앞서 EM 알고리즘을 통해 구한 방법은 posterior $p_\theta(Z|X)$로 가상의 데이터 z-variables를 만들고 complete data log-likelihood를 최대화한 것과 같다고 할 수 있습니다.

- 따라서 E-step과 M-step은 다음과 같이 일반할 수 있습니다.

  E-step : dataset 전체의 posterior $p_\theta(Z|X)$를 구한다.

  M-step : posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

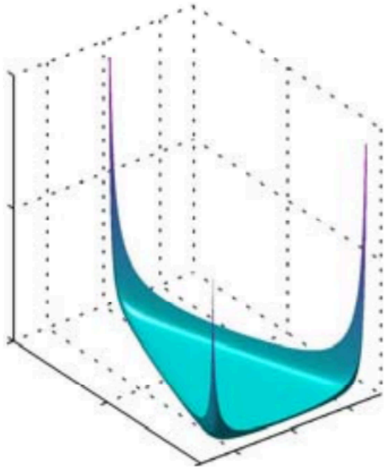
$$\theta^{new} = \arg\max_{\theta} \mathbb{E}_{p_{\theta old}(Z|X)}[\log p_\theta(X, Z)]$$

Variational Gaussian Mixture Models

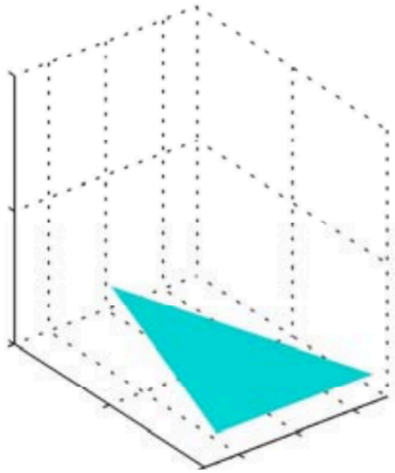
Variational Gaussian Mixture Models

- Gaussian Mixture Models에서 parameters로 사용되었던 mixing coefficients π_k, mean μ_k, covariance matrix Σ_k를 random variables로 여기고 probability distribution을 갖도록 모델링 합니다.

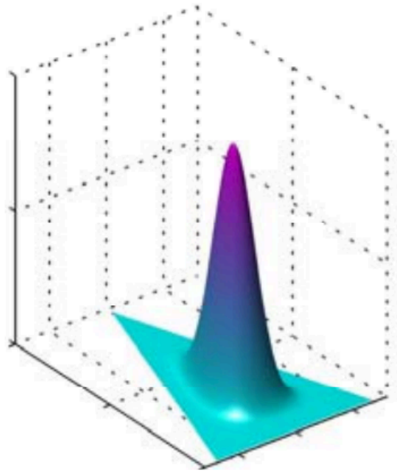
- Mixing coefficients π_k에 대한 prior는 Dirichlet로 정하고 이의 concentration parameter $\alpha_1, \ldots, \alpha_K$를 모두 α_0로 일치시킵니다. 또한 mean μ_k과 covariance matrix Σ_k에 대한 prior는 Gaussian-Wishart로 정합니다.



$\alpha_0 = 0.1$



$\alpha_0 = 1$

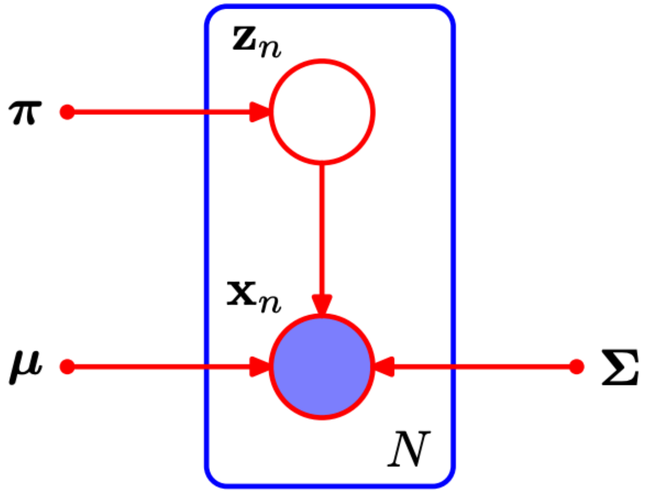


$\alpha_0 = 10$

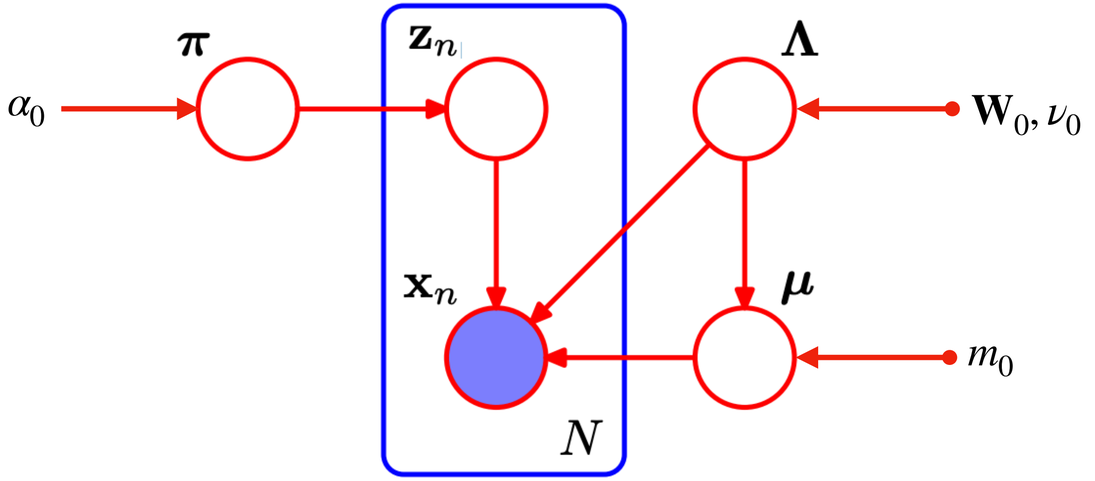
Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.78

Variational Gaussian Mixture Models

GMM



Variational GMM



Mixing Coefficients $p_\theta(z) = \prod_{k=1}^{K} \pi_k^{z_k}$

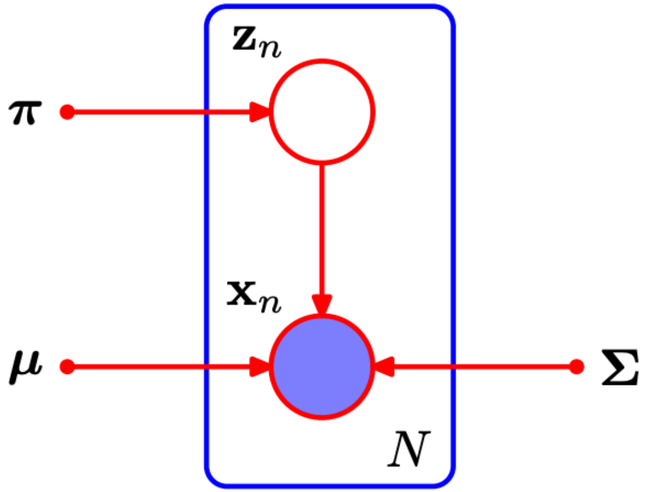
$p_\theta(z \mid \pi) = \prod_{k=1}^{K} \pi_k^{z_k}$

Dirichlet Prior $p_\theta(\pi) = Dir(\pi \mid \alpha_0) = C(\alpha_0) \prod_{k=1}^{K} \pi_k^{\alpha-1}$

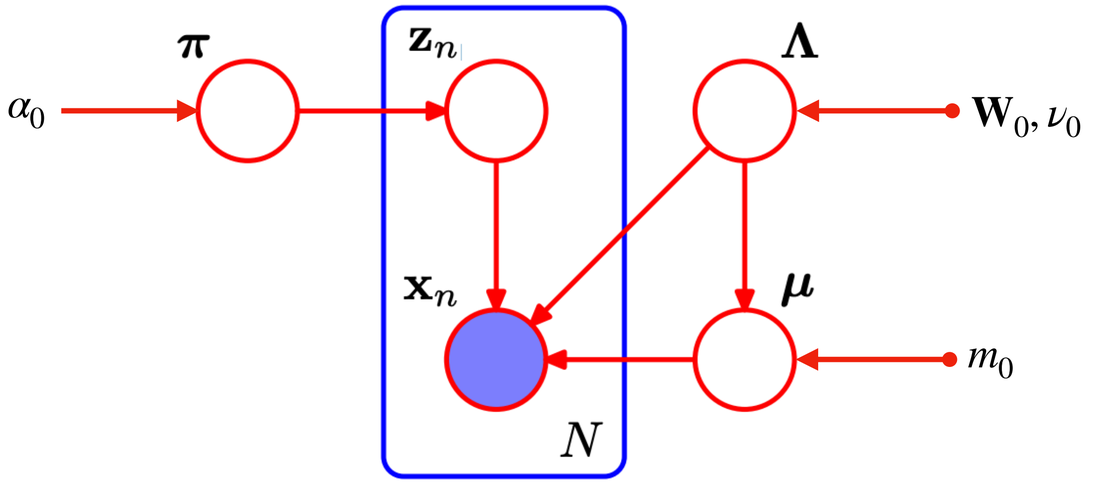
Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433, p.475

Variational Gaussian Mixture Models

GMM



Variational GMM



Conditional distribution

$$p_\theta(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

$$p_\theta(x \mid z, \mu, \Lambda) = \prod_{k=1}^{K} N(x \mid \mu_k, \Lambda_k^{-1})^{z_k}$$

Gaussian-Wishart Prior

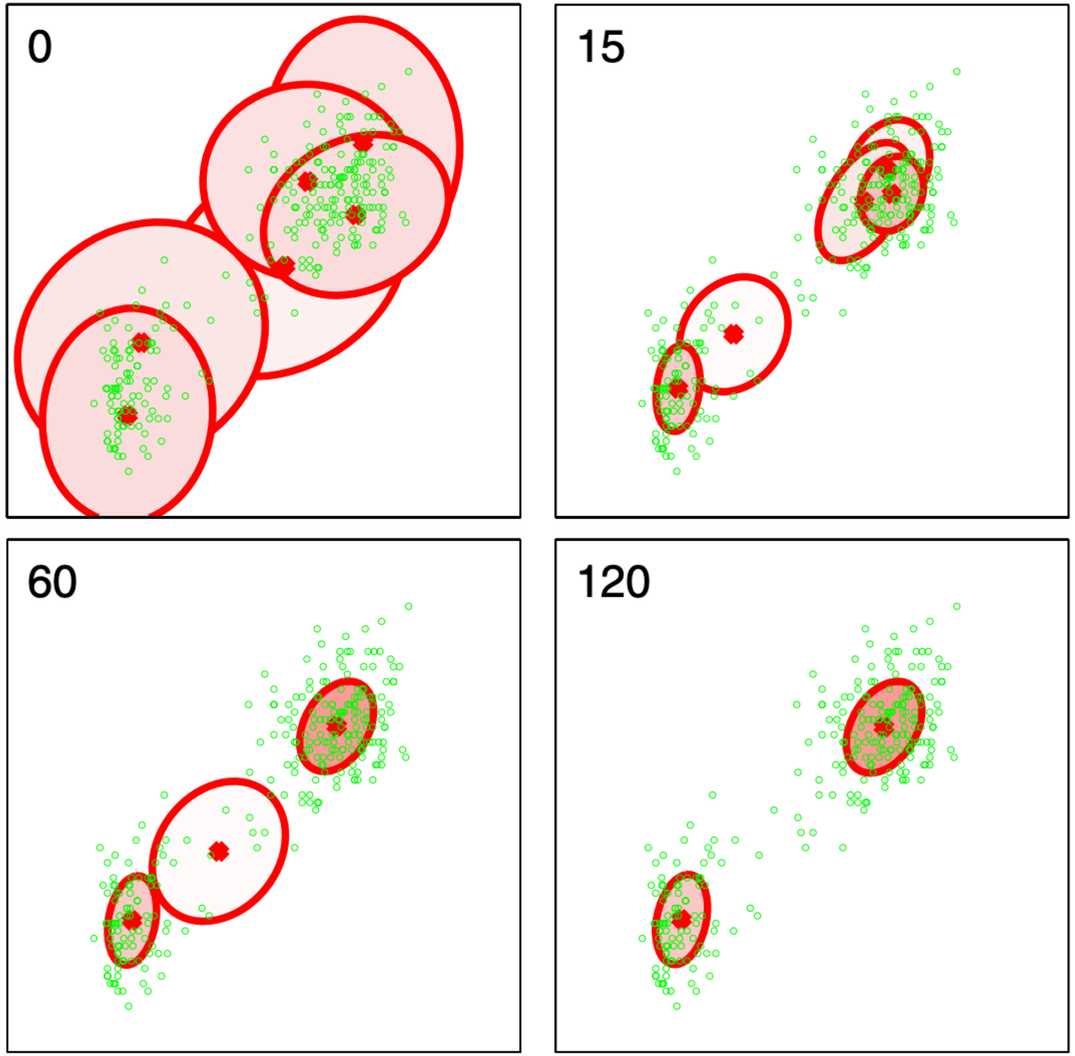
$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) p(\boldsymbol{\Lambda})$$

$$= \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\mu}_k \mid \mathbf{m}_0, \left(\beta_0 \boldsymbol{\Lambda}_k\right)^{-1}\right) \mathcal{W}\left(\boldsymbol{\Lambda}_k \mid \mathbf{W}_0, \nu_0\right)$$

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.433, p.475

Variational Gaussian Mixture Models

- Mixing coefficients π_k에 대한 prior를 Dirichlet로 두고 concentration parameter α_0을 낮은 값(1e-3)으로 설정하면, EM 알고리즘이 진행함에 따라 component 갯수가 적게 변하는 것을 볼 수 있습니다.

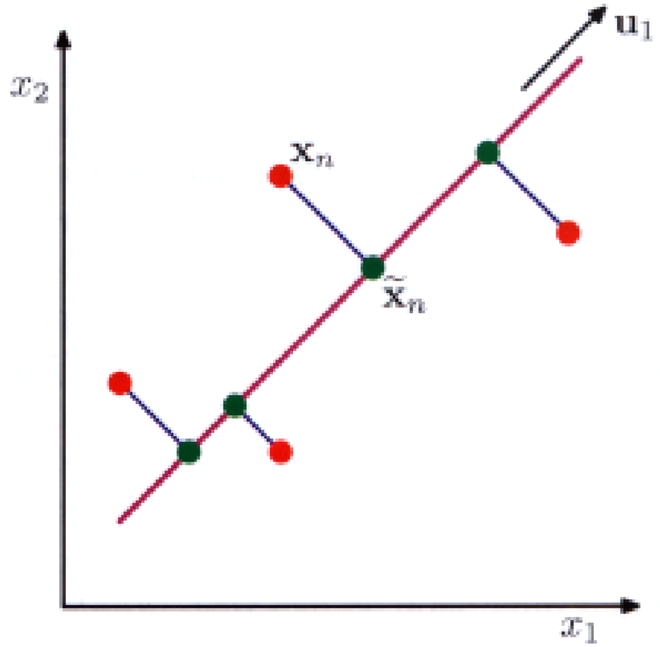


Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.480

PCA
(Principal Component Analysis)

PCA

- PCA (Principal Component Analysis)는 dimension reduction의 한 방법입니다.
- 이는 다차원을 가진 data에서 중요한 속성만을 남겨 lossy data compression에 이용되거나, feature extraction으로써 이미지 인식이나 음성 인식 등에 사용될 수 있습니다.
- 또는 2차원이나 3차원에 매핑하여 visualization을 하는데 사용될 수 있습니다.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.561

PCA

- 크기가 N인 dataset $X = (x_1, x_2, \ldots, x_N)$가 주어졌다고 합시다. 이 때, 데이터 x_n의 dimension은 D라고 정의합니다.
- PCA의 목적은 데이터를 D차원보다 작은 M차원으로 projection하는 것입니다. 이 때, projected data의 분산이 최대가 되도록 합니다.
- 먼저 $M = 1$인 경우를 생각해봅시다. D차원의 본래 space상에서 1차원의 subspace로 data들이 projection됩니다. 1차원의 basis가 되는 unit vector를 u_1이라고 둡니다.
- 데이터 x_n을 u_1 위에 projection시킨 점을 z_n이라하면, $z_n = \frac{u_1^T x_n}{u_1^T u_1} u_1 = (u_1^T x_n) u_1$이 됩니다.
- 데이터셋의 모든 데이터에 대해 projection을 시키고, projection된 데이터들의 mean $\bar{z}_n$을 구하면, $\bar{z}_n = \frac{1}{N} \sum_{n=1}^{N} (u_1^T x_n) u_1 = u_1^T (\frac{1}{N} \sum_{n=1}^{N} x_n) u_1 = (u_1^T \bar{x}) u_1$가 됩니다. 단, $\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$, 전체 데이터들의 평균.
- 따라서 projection된 후 basis u_1상의 분산은 $\frac{1}{N} \sum_{n=1}^{N} u_1^T x_n - u_1^T \bar{x} = u_1^T S u_1$가 됩니다. 단, $S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$, 전체 데이터들의 covariance matrix.

PCA

- PCA의 목적은 위 식 $u_1^T S u_1$을 최대화하는 u_1을 구하는 것으로 구체화 됩니다.
- u_1은 크기가 1인 unit vector이므로 u_1의 크기를 키워서 최대화할 수는 없고, 크기가 1이라는 condition을 고려합니다.
- 즉, Lagrange multiplier λ_1을 도입하고, $u_1^T u = 1$이라는 condition을 붙입니다.$u_1^T S u_1 + \lambda_1(1 - u_1^T u_1)$
- 위 식을 u_1에 대해 미분하면 $Su_1 - \lambda_1 u_1$이 되고, 이를 0으로 두어 정리하면, $Su_1 = \lambda_1 u_1$이 됩니다.
- u_1은 S의 eigenvector이고, λ_1은 S의 eigenvalue임을 알 수 있습니다.
- projection된 data의 variance인 $u_1^T S u_1$는 S의 가장 큰 eigenvector로 u_1을 설정함으로써 최대화 됩니다.
- 이러한 방식에 의해, M차원의 subspace로 projection하여 variance가 최대가 되도록 하려면, data covariance matrix S에서 가장 큰 M개의 eigenvector $u_1, \ldots, u_M$을 basis로 하도록 해야함을 알 수 있습니다.

Probabilistic PCA

Probabilitic PCA

- PPCA(Probabilistic PCA)는 dimension reduction에 사용되는 PCA의 (semi) Bayesian 버전입니다. parameters를 random variables로 두지 않았으므로 fully Bayesian이라고 볼 수 없습니다.

- GMM에서 discrete한 z-variable을 갖는데 반해, PPCA에서는 continous한 z-variable을 갖습니다.

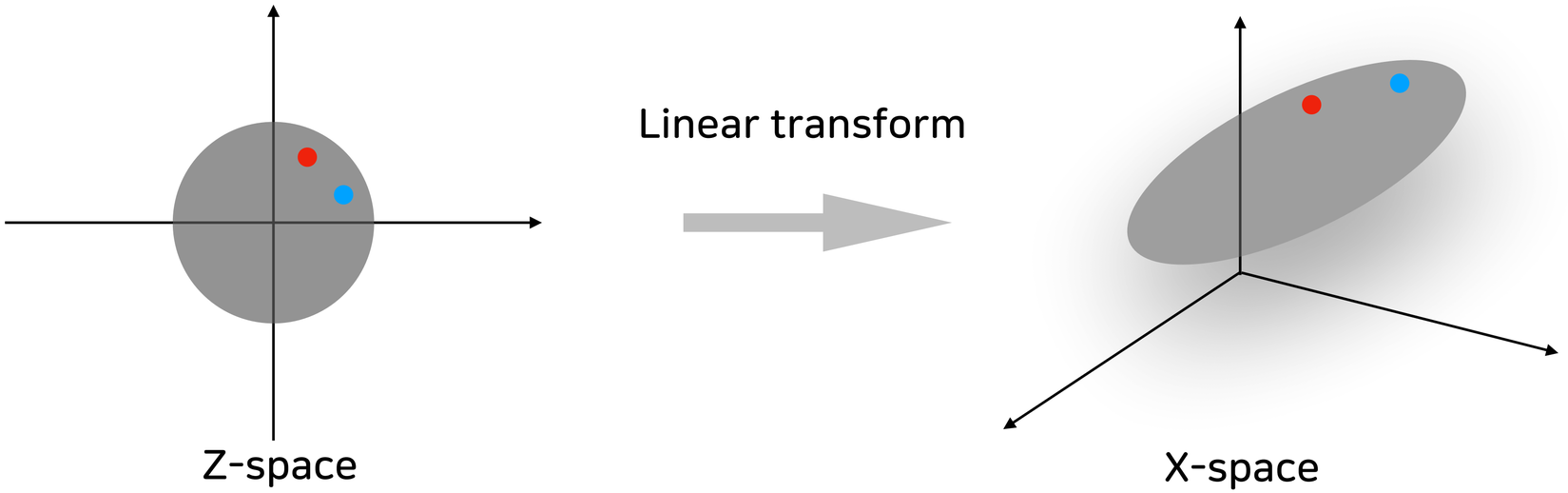
- GMM에서 data point 하나를 얻기 위해 다음과 같은 순서에 의해 샘플링 하였습니다.

  1. K개의 components(Gaussian distributions) 중 하나를 선택
  2. 선택한 component에서 data point 샘플링

- PPCA에서는 다음과 같은 순서에 의해 샘플링이 이루어집니다.

  1. Z-space의 Gaussian distribution에서 샘플 하나를 선택
  2. 샘플을 linear transform을 통해 X-space로 맵핑
  3. 맵핑된 값을 mean으로 하는 또 다른 Gaussian distribution에서 data point 샘플링

Probabilistic PCA



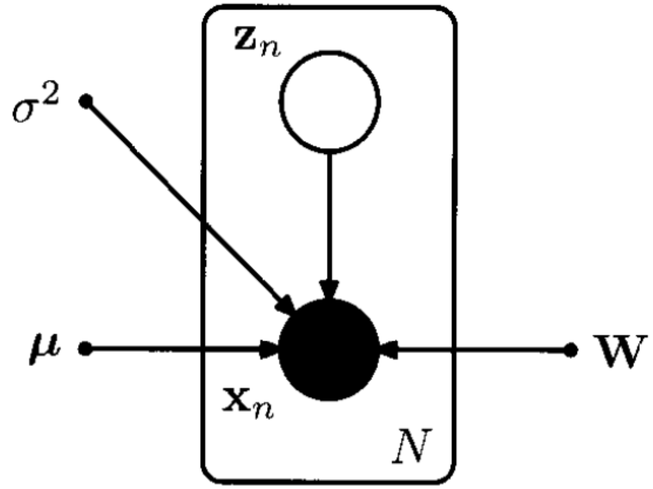
Prior	$p_\theta(z) = N(z \mid 0, I)$
Conditional	$p_\theta(x \mid z) = N(x \mid Wz + \mu, \sigma^2 I)$

(Linear-Gaussian Model)

References :
Michael E. Tipping and Christopher M. Bishop. Probabilistic Principal Component Analysis. 1999.
Christopher M. Bishop. Pattern Recognition and Machine Learning. p.570-580

Probabilistic PCA

Graphical Representation



Prior	$p_\theta(z) = N(z \mid 0, I)$
Conditional	$p_\theta(x \mid z) = N(x \mid Wz + \mu, \sigma^2 I)$

(Linear-Gaussian Model)

Probabilitic PCA

- Maximum likelihood를 통해 parameters를 구하기 위해 marginal likelihood를 구합니다.
-

$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(z)p_\theta(x|z)dz = \int N(z|0,I)N(x|Wz+\mu,\sigma^2 I)dz$$

$$= N(x|\mu, C),\ where\ C = WW^T + \sigma^2 I$$

$$p(x) = N(x|\mu, \Lambda^{-1})$$

$$p(y|x) = N(y|Ax+b, L^{-1})$$

$$p(y) = N(y|A\mu+b, L^{-1}+A\Lambda^{-1}A^T)$$

PRML p.93

- 위 식을 통해 dataset X의 marginal log-likelihood를 구합니다.
-

$$log\ p_\theta(X) = \sum_{n=1}^{N} log\ p_\theta(x_n)$$

$$= -\frac{ND}{2}log(2\pi) - \frac{N}{2}log|C| - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^T C^{-1}(x_n-\mu)$$

Probabilistic PCA Maximum Likelihood

- GMM에서와 다르게 marginal likelihood가 Gaussian이 되고, 이를 최대화하는 parameters를 다음과 같이 closed form으로 구할 수 있습니다.

- $W = U_M(L_M - \sigma^2 I)^{1/2}R$

$U_M L_M U_M^T = S$ (*data covariance matrix*)

$U_M = [v_1 \quad v_2 \quad \cdots \quad v_M]$, *where* $v_1, v_2, \ldots, v_M$ = *eigenvectors given by the data covariance matrix*

$$L_M = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_M \end{bmatrix}$$, *where* $\lambda_1, \lambda_2, \ldots, \lambda_M$ = *the corresponding eigenvalues*

R_M = *an arbitary* $M \times M$ *orthogonal matrix*

- $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$

- $\sigma^2 = \frac{1}{D-M}\sum_{i=M+1}^{D} \lambda_i$

Probabilistic PCA Expectation-Maximization

- 또는 GMM에서 했듯이 EM알고리즘으로 iteration을 통해 구할 수도 있습니다.

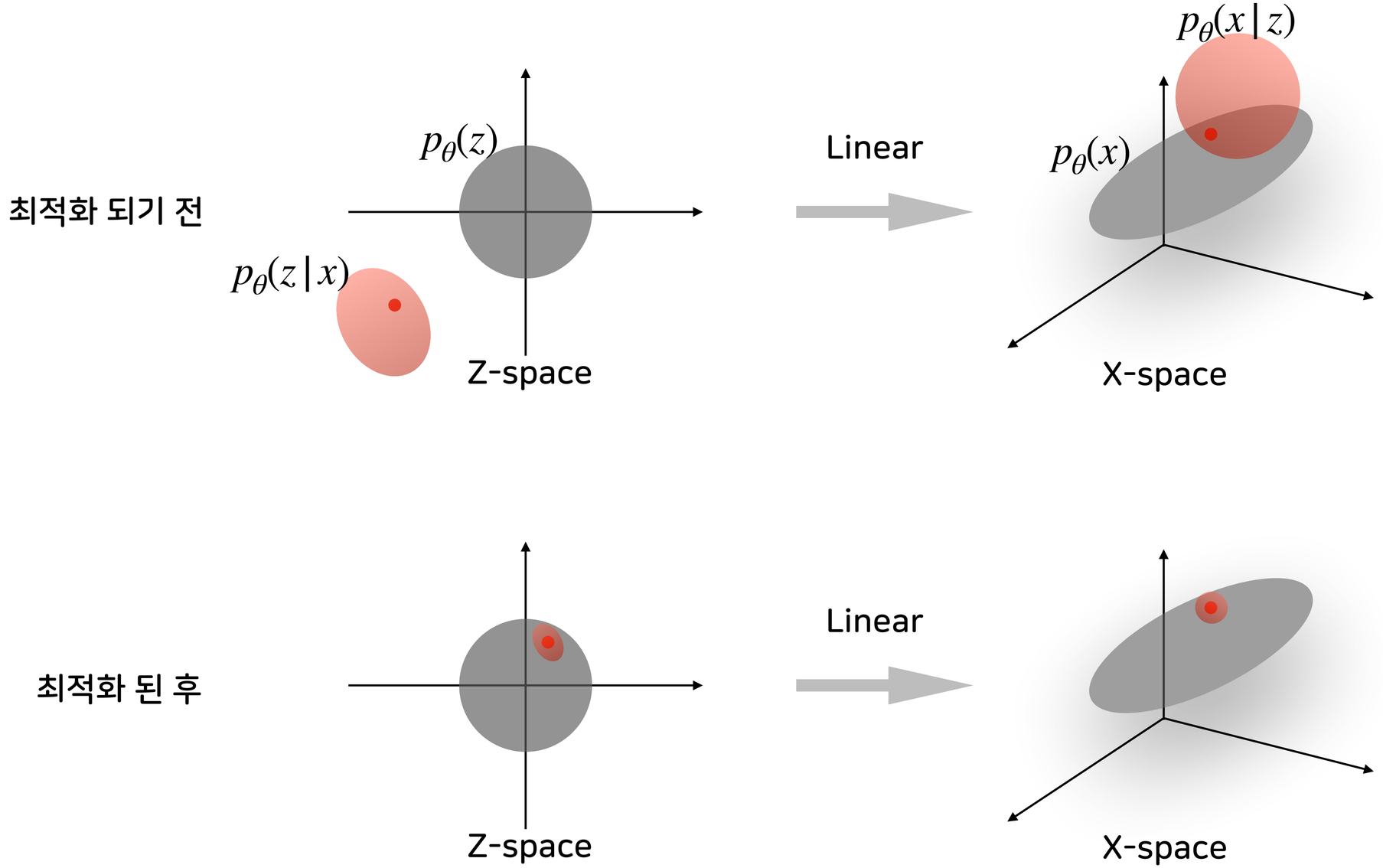
- PPCA에서 sampling은 두 번 이루어집니다.

  1. Prior $p_\theta(z)$에서 sampling
  2. Linear transform후 얻어진 conditional distribution $p_\theta(x \mid z)$에서 sampling

- 위 sampling을 통해 가지고 있는 dataset이 잘 나올 수 있도록 하기 위해서는 다음과 같은 두가지 조건을 만족해야 합니다.

  1. posterior $p_\theta(z \mid x)$에서 뽑은 sample이 prior $p_\theta(z)$에서 높은 likelihood를 가져야 한다.
  2. posterior $p_\theta(z \mid x)$에서 뽑은 sample로 linear transform을 시킨 후 얻은 conditional distribution $p_\theta(x \mid z)$에서 가지고 있는 dataset이 높은 likelihood를 가져야 한다.

Probabilistic PCA Expectation-Maximization



Probabilistic PCA Expectation-Maximization

- 위 두가지 조건을 만족하도록 하는 parameters를 구하기 위해 수식을 전개하면 다음과 같이 됩니다.

- $$\theta^{new} = \arg\max_{\theta} \mathbb{E}_{p_{\theta old}(Z|X)}[\underbrace{\log p_\theta(Z)}_{1.} + \underbrace{\log p_\theta(X|Z)}_{2.}]$$

- 1. posterior $p_\theta(z|x)$에서 뽑은 sample이 prior $p_\theta(z)$에서 높은 likelihood를 가져야 한다.
  2. posterior $p_\theta(z|x)$에서 뽑은 sample로 linear transform을 시킨 후 얻은 conditional distribution $p_\theta(x|z)$에서 가지고 있는 dataset이 높은 likelihood를 가져야 한다.

- 이는 곧 GMM에서 했던 EM 알고리즘의 M-step과 같은 식이 됩니다.

- $$\theta^{new} = \arg\max_{\theta} \mathbb{E}_{p_{\theta old}(Z|X)}[\log p_\theta(X, Z)]$$

Probabilistic PCA Expectation-Maximization

- M-step에서는 posterior에 의해 가상으로 만들어진 complete data의 log-likelihood를 최대화 합니다. complete data log-likelihood는 다음과 같이 계산됩니다.

- $$\begin{aligned}\log p_\theta(X,Z) &= \log p_\theta(Z) + \log p_\theta(X|Z) \\ &= \log \prod_{n=1}^{N} p_\theta(z_n) + \log \prod_{n=1}^{N} p_\theta(x_n|z_n) \\ &= \sum_{n=1}^{N} \log p_\theta(z_n) + \sum_{n=1}^{N} \log p_\theta(x_n|z_n) \\ &= \sum_{n=1}^{N} \log N(z_n|0,I) + \sum_{n=1}^{N} \log N(x_n|Wz_n+\mu,\sigma^2 I)\end{aligned}$$

- 또한 posterior $p_\theta(Z|X)$에 대한 complete data log-likelihood $\log p_\theta(X,Z)$의 기댓값은 다음과 같습니다.

- $$\begin{aligned}\mathbb{E}_{p_\theta(Z|X)}\left[\ln p_\theta(\mathbf{X},\mathbf{Z})\right] = -\sum_{n=1}^{N}\Bigg\{ &\frac{D}{2}\ln\left(2\pi\sigma^2\right) + \frac{1}{2}\mathrm{Tr}\left(\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\right) \\ &+\frac{1}{2\sigma^2}\left\|\mathbf{x}_n-\boldsymbol{\mu}\right\|^2 - \frac{1}{\sigma^2}\mathbb{E}\left[\mathbf{z}_n\right]^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\left(\mathbf{x}_n-\boldsymbol{\mu}\right) \\ &+\frac{1}{2\sigma^2}\mathrm{Tr}\left(\mathbb{E}\left[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\right]\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)\Bigg\}\end{aligned}$$

Probabilistic PCA Expectation-Maximization

- E-step : 각 data point 마다 posterior $p_\theta(z \mid x)$를 구합니다.

$$p_\theta(z \mid x) = N(z \mid M^{-1}W^T(x-\mu), \sigma^2 M^{-1})$$

$$where\ M = W^T W + \sigma^2 I$$

- M-step : posterior를 이용해 parameters를 업데이트 합니다. μ는 sample mean이 자명하므로 closed form으로 구합니다.

$$W_{new} = \left[\sum_{n=1}^{N}(x_n - \bar{x})E[z_n]^T\right]\left[\sum_{n=1}^{N}E[z_n z_n^T]\right]^{-1}$$

$$\sigma^2_{new} = \frac{1}{ND}\sum_{n=1}^{N}\left\{\|x_n - \bar{x}\|^2 - 2E[z_n]^T W_{new}^T(x_n - \bar{x}) + Tr(E[z_n z_n^T]W_{new}^T W_{new})\right\}$$

where $E[z_n z_n^T] = \sigma^2 M^{-1} + E[z_n]E[z_n]^T$

Expectation & Maximization

Expectation & Maximization

- 앞서 GMM과 PPCA에서 해왔던 EM을 일반화한 형태를 써보면 다음과 같습니다.
- 1. Parameter θ_{old}를 초기화 한다.
- 2. E-step : posterior $p(\mathbf{Z}|\mathbf{X}, \theta^{old})$를 구한다.
- 3. M-step : 새로운 parameter θ^{new}를 구한다.
-

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right)$$

where $$\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right) = \mathbb{E}_{p\left(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old}\right)}[p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})]$$

$$= \sum_{\mathbf{Z}} p\left(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old}\right) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

- 4. Log likelihood나 parameter 값이 수렴할 때까지 $\boldsymbol{\theta}^{old}$를 $\boldsymbol{\theta}^{new}$로 대체하고 E-step과 M-step을 반복한다.

KL-Divergence

Information Theory

- 다음과 같은 기준들에 의해 정보(information)을 수량화 합니다.
- 1. 자주 일어나는 사건은 낮은 정보량을 갖는다.

  2. 드물게 일어나는 사건은 높은 정보량을 갖는다.

  3. 독립된 사건의 정보량은 각 사건의 정보량을 더하여 구한다.
-

$$h(x) = -\log p(x)$$

- Random variable $\mathbf{x}$의 distribution이 $p(x)$일 때, $\mathbf{x}$의 엔트로피(entropy)는 다음과 같이 정의 합니다.
-

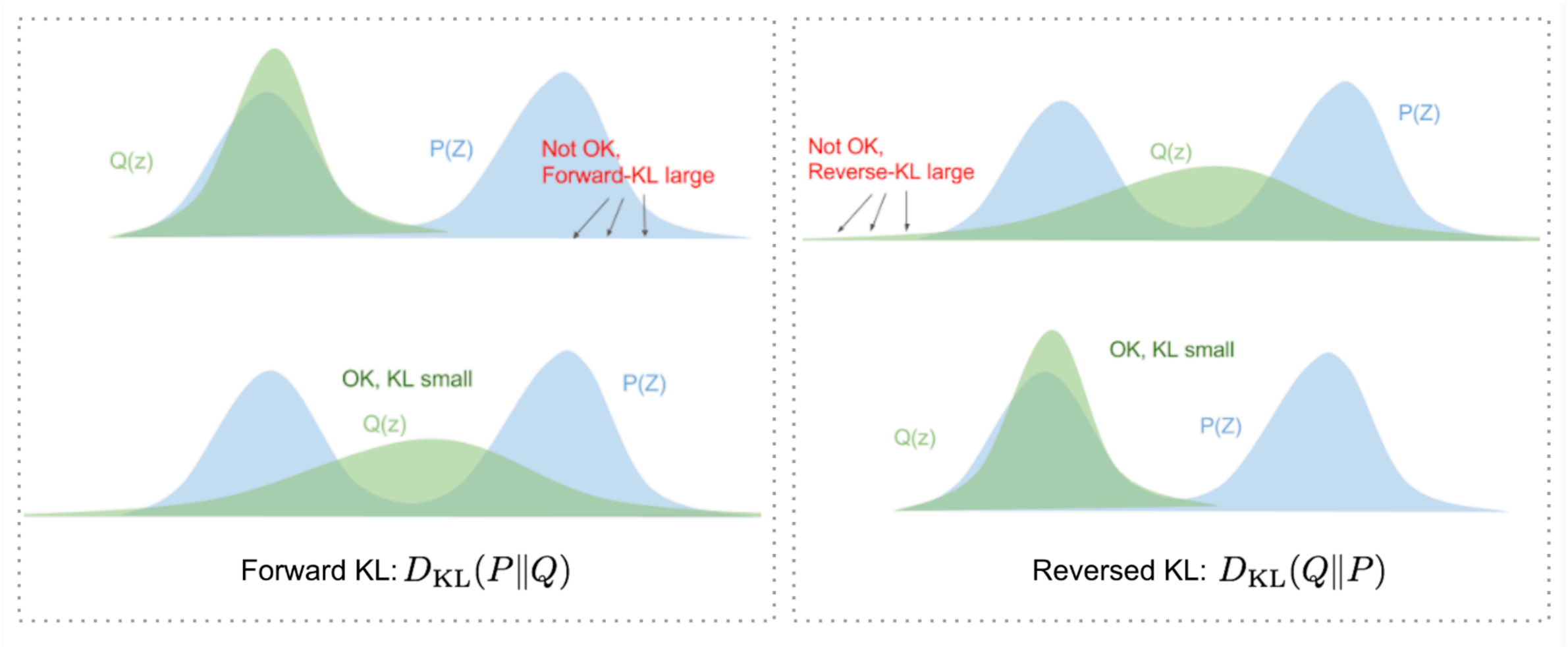
$$H[\mathbf{x}] = \mathbb{E}_{p(x)}[h(x)] = \begin{cases} \sum_x p(x)\{-\log p(x)\}, \text{ discrete} \\ \int p(x)\{-\log p(x)\}dx, \text{ continuous} \end{cases}$$

KL-Divergence

- KL-divergence는 두 분포의 차이를 재는 범함수(functional)입니다.
- 두 분포 P, Q에 대해서 다음과 같이 KL-divergence를 정의합니다.
- $$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{x\sim P}\left[\log\frac{P(x)}{Q(x)}\right] = \begin{cases} \sum_x P(x)\log\frac{P(x)}{Q(x)}, \text{ discrete} \\ \int P(x)\log\frac{P(x)}{Q(x)}dx, \text{ continuous} \end{cases}$$
- KL-divergence는 symmetric하지 않으므로 수학적으로 distance의 개념이 아닙니다.
- $$D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$$
- KL-divergence 값은 항상 0보다 같거나 크며, 두 distribution P, Q가 같을 때만 0이 됩니다.
- $$\mathbb{E}_{x\sim P}\left[-\log\frac{Q(x)}{P(x)}\right] \geq -\log\left(\mathbb{E}_{x\sim P}\left[\frac{Q(x)}{P(x)}\right]\right) = -\log\left(\sum_x P(x)\frac{Q(x)}{P(x)}\right) = 0$$

by Jensson's inequality

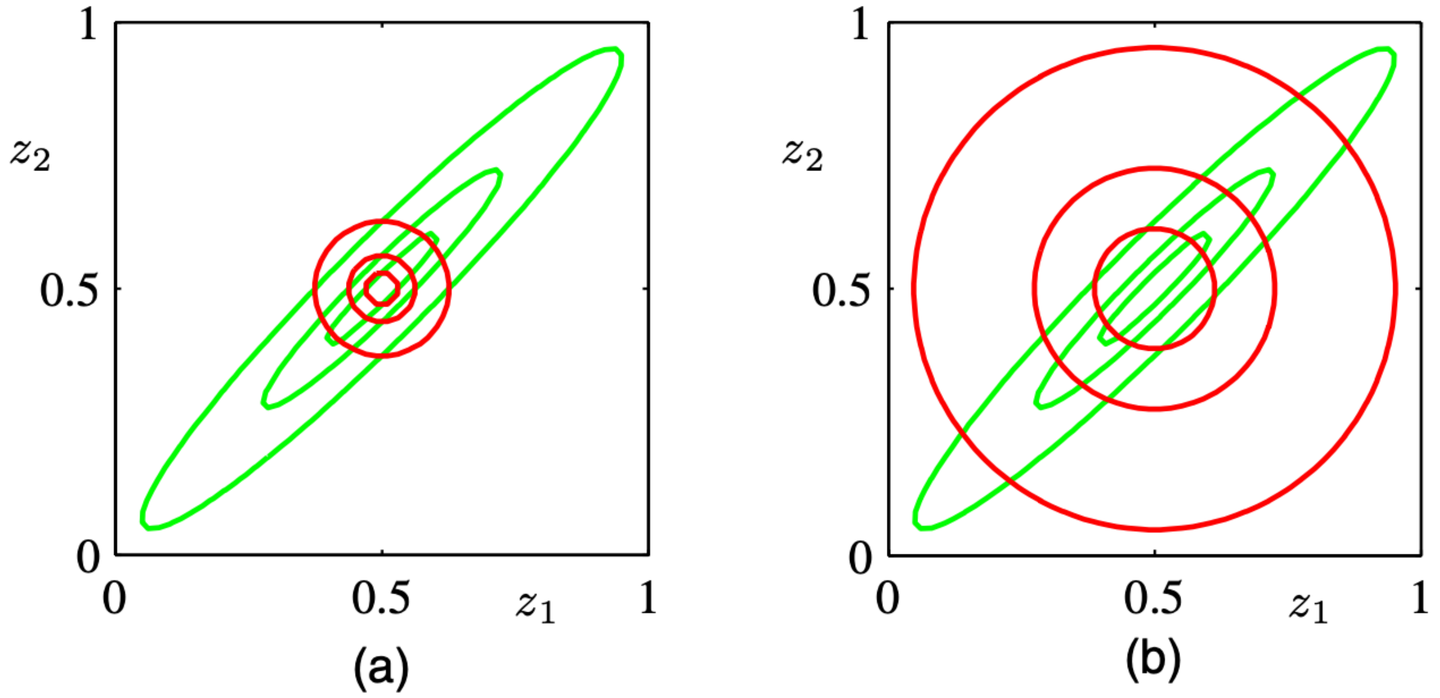
KL-Divergence



https://blog.evjang.com/2016/08/variational-bayes.html

KL-Divergence

Figure 10.2 Comparison of the two alternative forms for the Kullback-Leibler divergence. The green contours corresponding to 1, 2, and 3 standard deviations for a correlated Gaussian distribution $p(\mathbf{z})$ over two variables z_1 and z_2, and the red contours represent the corresponding levels for an approximating distribution $q(\mathbf{z})$ over the same variables given by the product of two independent univariate Gaussian distributions whose parameters are obtained by minimization of (a) the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$, and (b) the reverse Kullback-Leibler divergence $\mathrm{KL}(p\|q)$.



Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.468

KL-Divergence

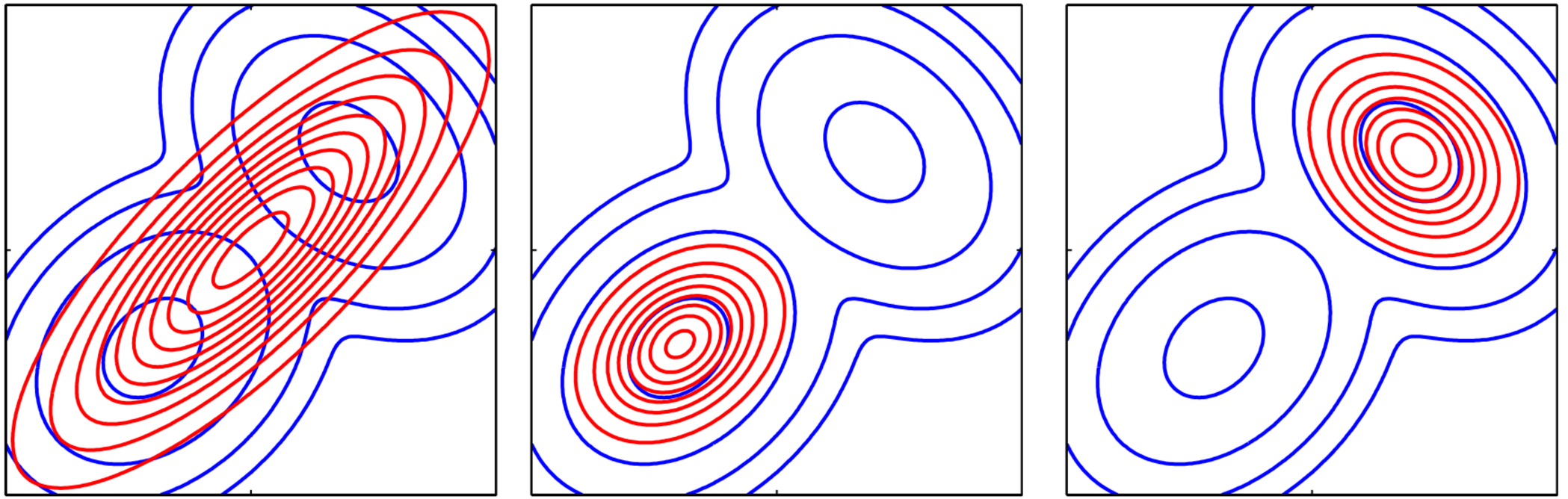


Figure 10.3 Another comparison of the two alternative forms for the Kullback-Leibler divergence. (a) The blue contours show a bimodal distribution $p(\mathbf{Z})$ given by a mixture of two Gaussians, and the red contours correspond to the single Gaussian distribution $q(\mathbf{Z})$ that best approximates $p(\mathbf{Z})$ in the sense of minimizing the Kullback-Leibler divergence $\mathrm{KL}(p\|q)$. (b) As in (a) but now the red contours correspond to a Gaussian distribution $q(\mathbf{Z})$ found by numerical minimization of the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$. (c) As in (b) but showing a different local minimum of the Kullback-Leibler divergence.

Christopher M. Bishop, Pattern Recognition And Machine Learning 2006, p.469

General EM

General EM-posterior 계산의 문제점

- EM 알고리즘은 E-step에서 posterior $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$를 구하는 과정이 필수적입니다.
-

$$p_\theta(\mathbf{Z}\,|\,\mathbf{X}) = \frac{p_\theta(\mathbf{Z})p_\theta(\mathbf{X}\,|\,\mathbf{Z})}{p_\theta(\mathbf{X})}$$

- 그러나 분모에 있는 $p_\theta(\mathbf{X})$를 구하기 위해 적분 또는 summation을 해야 하는데 실제로 어려운 (intractable) 경우가 있습니다.
-

$$p_\theta(\mathbf{X}) = \int p_\theta(\mathbf{Z})p_\theta(\mathbf{X}\,|\,\mathbf{Z})d\mathbf{Z},\ \mathbf{Z}\text{가 continous인 경우,}$$

$$= \sum_{\mathbf{Z}} p_\theta(\mathbf{Z})p_\theta(\mathbf{X}\,|\,\mathbf{Z}),\ \mathbf{Z}\text{가 discrete인 경우}$$

General EM-Variational Distribution의 도입

- E-step에서 posterior $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$를 구하는 것은 다음과 같이 KL-divergence를 최소화 하는 variational distribution $q_\phi^*(\mathbf{Z})$를 구하는 것으로 대체할 수 있습니다.

-

$$q_\phi^*(\mathbf{Z}) = \arg\min_{q_\phi(\mathbf{Z})} \mathrm{KL}(q\|p)$$

$$\text{where } \mathrm{KL}(q\|p) = -\mathbb{E}_{q_\phi(\mathbf{Z})}[\ln p_\theta(\mathbf{Z}\,|\,\mathbf{X}) - q_\phi(\mathbf{Z})]$$

$$= -\sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \left\{ \frac{p_\theta(\mathbf{Z}\,|\,\mathbf{X})}{q_\phi(\mathbf{Z})} \right\}$$

- $\mathrm{KL}(q\|p)$는 $q_\phi(\mathbf{Z})$와 $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$가 같을 때 0이 되므로 $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$를 대신해 $q_\phi^*(\mathbf{Z})$를 사용할 수 있습니다.

General EM-ELBO 사용

- posterior $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$를 구하기 어려운 경우 $\mathrm{KL}(q\|p)$ 식을 직접 계산할 수 없으므로 우회하여 구해봅니다.

- Marginal log-likelihood $\ln p_\theta(\mathbf{X})$와 ELBO라 불리는 $\mathscr{L}(q,\boldsymbol{\theta})$와 KL-divergence $\mathrm{KL}(q\|p)$ 간에는 다음과 같은 식이 성립합니다.

-

$$\ln p_\theta(\mathbf{X}) = \mathscr{L}(q,\boldsymbol{\theta}) + \mathrm{KL}(q\|p)$$

$$\text{where } \mathscr{L}(q,\boldsymbol{\theta}) = \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \left\{ \frac{p_{\boldsymbol{\theta}}(\mathbf{X},\mathbf{Z})}{q_\phi(\mathbf{Z})} \right\}$$

$$\mathrm{KL}(q\|p) = -\sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \left\{ \frac{p_{\boldsymbol{\theta}}(\mathbf{Z}\,|\,\mathbf{X})}{q_\phi(\mathbf{Z})} \right\}$$

General EM-ELBO 유도

- $$\ln p_\theta(\mathbf{X}) = \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln p_\theta(\mathbf{X})$$

$$= \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \frac{p_\theta(\mathbf{X}, \mathbf{Z})}{p_\theta(\mathbf{Z} \mid \mathbf{X})}$$

$$= \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \frac{p_\theta(\mathbf{X}, \mathbf{Z})}{q_\phi(\mathbf{Z})} \frac{q_\phi(\mathbf{Z})}{p_\theta(\mathbf{Z} \mid \mathbf{X})}$$

$$= \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \frac{p_\theta(\mathbf{X}, \mathbf{Z})}{q_\phi(\mathbf{Z})} - \sum_{\mathbf{Z}} q_\phi(\mathbf{Z}) \ln \frac{p_\theta(\mathbf{Z} \mid \mathbf{X})}{q_\phi(\mathbf{Z})}$$

$$= \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q \| p)$$

General EM-ELBO 사용

- Parameter θ가 고정되어 있을 때 marginal log-likelihood $\ln p_\theta(\mathbf{X})$는 고정이므로 KL-divergence $\mathrm{KL}(q\|p)$를 최소화하는 것은 ELBO $\mathscr{L}(q,\boldsymbol{\theta})$를 최대화하는 것이 됩니다.

- 이점을 이용해 이용해 $\mathrm{KL}(q\|p)$를 최소화하는 것이 아닌 $\mathscr{L}(q,\boldsymbol{\theta})$를 최대화하는 방향으로 $p_\theta(\mathbf{Z}\,|\,\mathbf{X})$를 근사하는 $q_\phi(\mathbf{Z})$를 구할 수 있습니다.

-

$$q_\phi^*(\mathbf{Z}) = \underset{q_\phi(\mathbf{Z})}{\arg\min}\ \mathrm{KL}(q\|p) = \underset{q_\phi(\mathbf{Z})}{\arg\max}\ \mathscr{L}(q,\boldsymbol{\theta})$$

General EM-Variational EM

- EM 알고리즘에서 posterior를 구하는 E-step과 model parameter를 업데이트하는 M-step은 variational posterior $q_\phi(\mathbf{Z})$를 이용하여 다음과 같이 일반화 할 수 있습니다.

(Classic) EM Algorithm

- E-step : dataset 전체의 posterior $p_\theta(Z|X)$를 구한다.
- M-step : posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

$$\theta^{new} = \arg\max_{\theta} \mathbb{E}_{p_{\theta old}(\mathbf{Z}|\mathbf{X})}[\log p_\theta(\mathbf{X}, \mathbf{Z})]$$

Variational EM Algorithm

- E-step : dataset 전체의 variational posterior를 구한다.

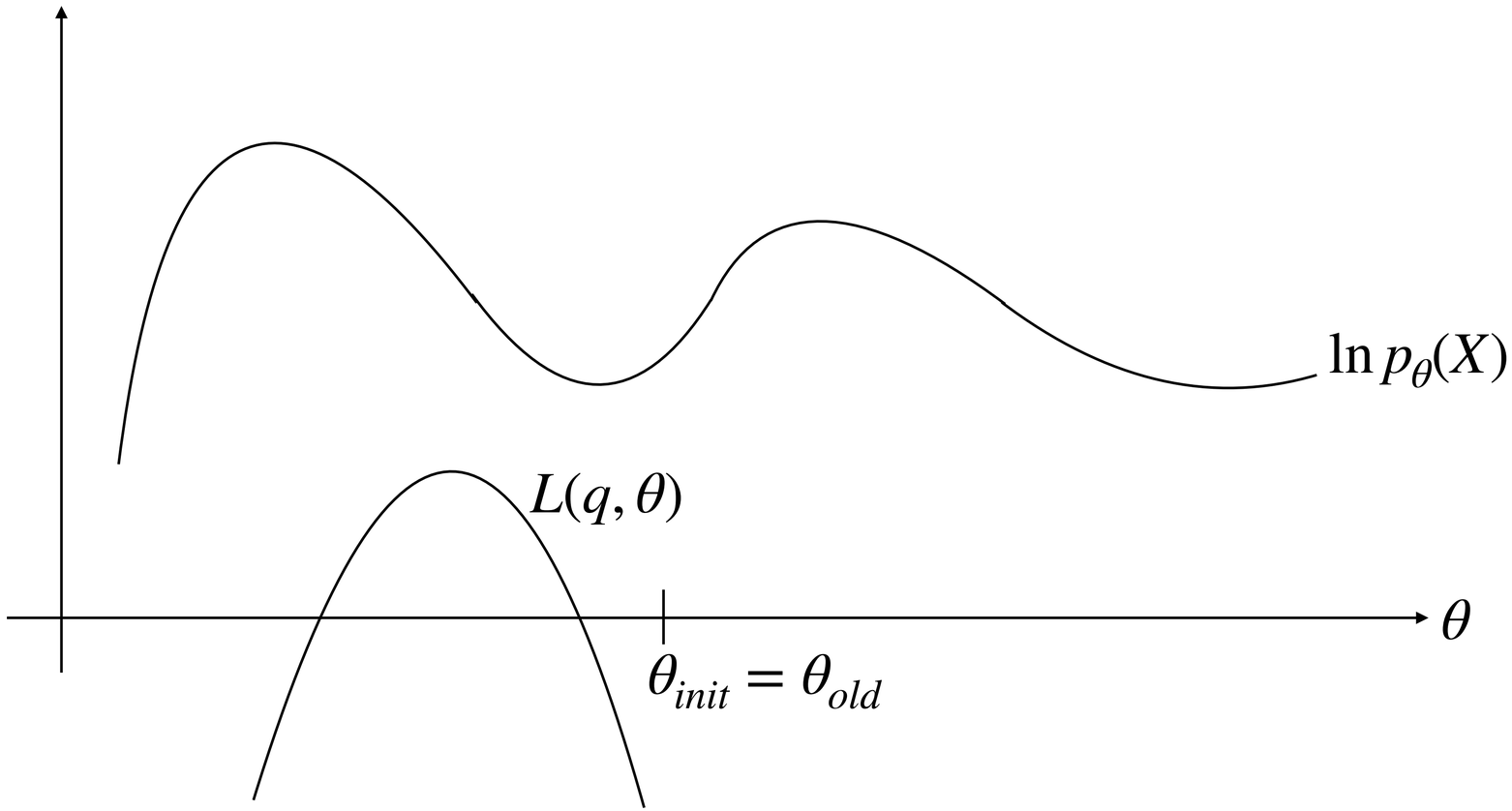
$$q_\phi^*(\mathbf{Z}) = \arg\max_{q_\phi(\mathbf{Z})} \mathscr{L}(q, \boldsymbol{\theta})$$

- M-step : variational posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

$$\theta^{new} = \arg\max_{\theta} \mathbb{E}_{q_\phi^*(\mathbf{Z})}[\log p_\theta(\mathbf{X}, \mathbf{Z})]$$

General EM

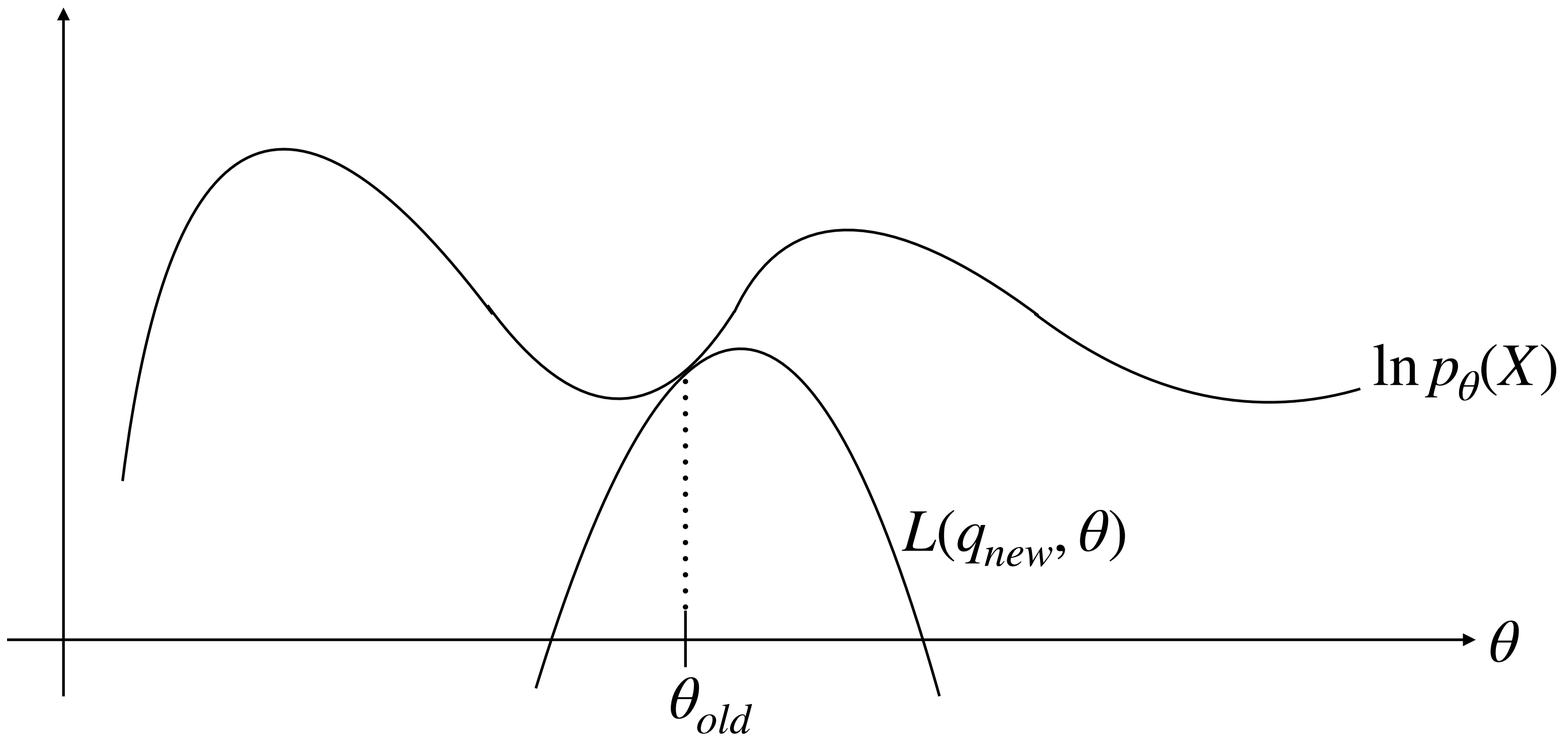
Initialize



General EM

E-step

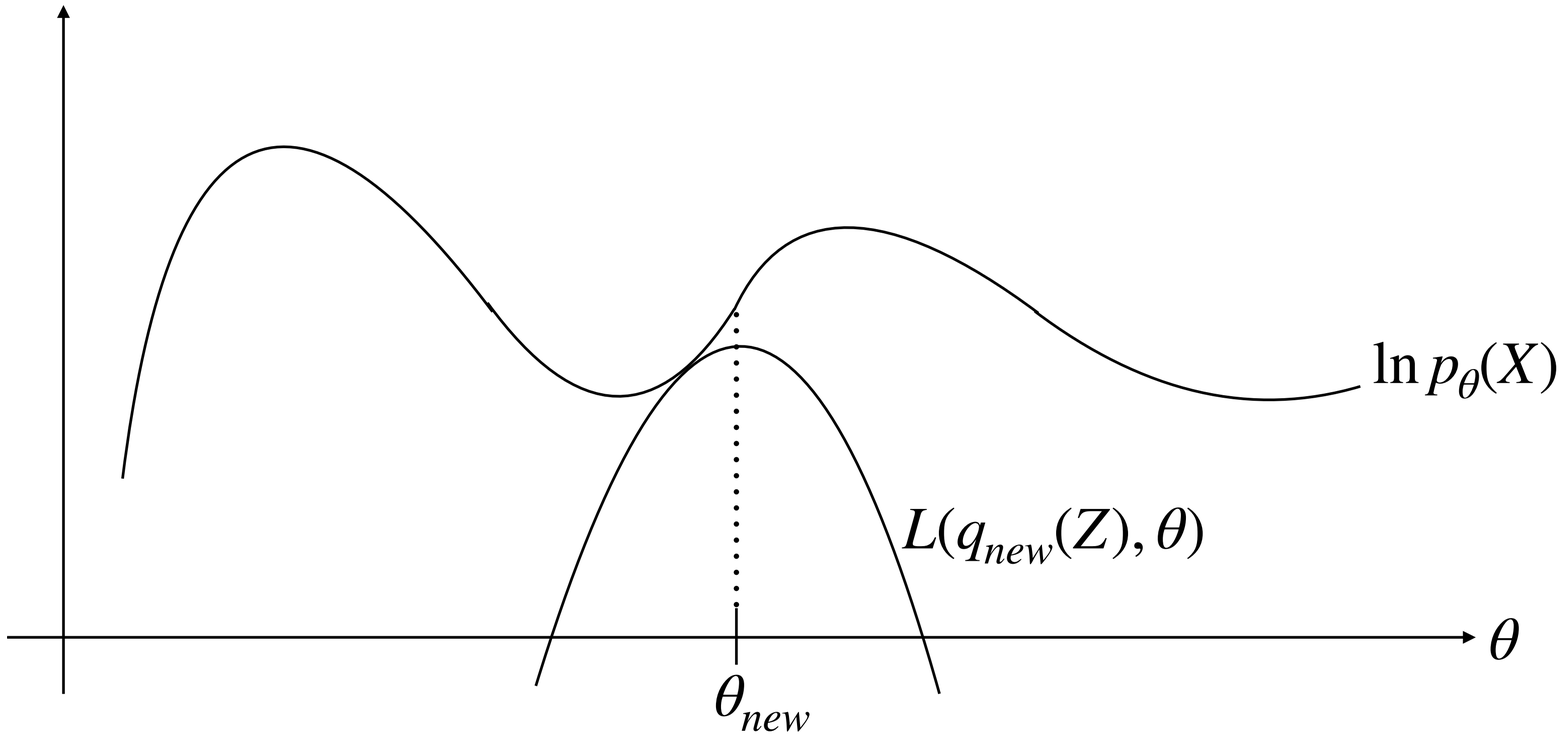
$Set\ q_{new}(Z) = p_{\theta_{old}}(Z|X)$



General EM

M-step

$$Find\ the\ \theta_{new}\ that\ maximize\ L(q_{new}(Z), \theta)$$

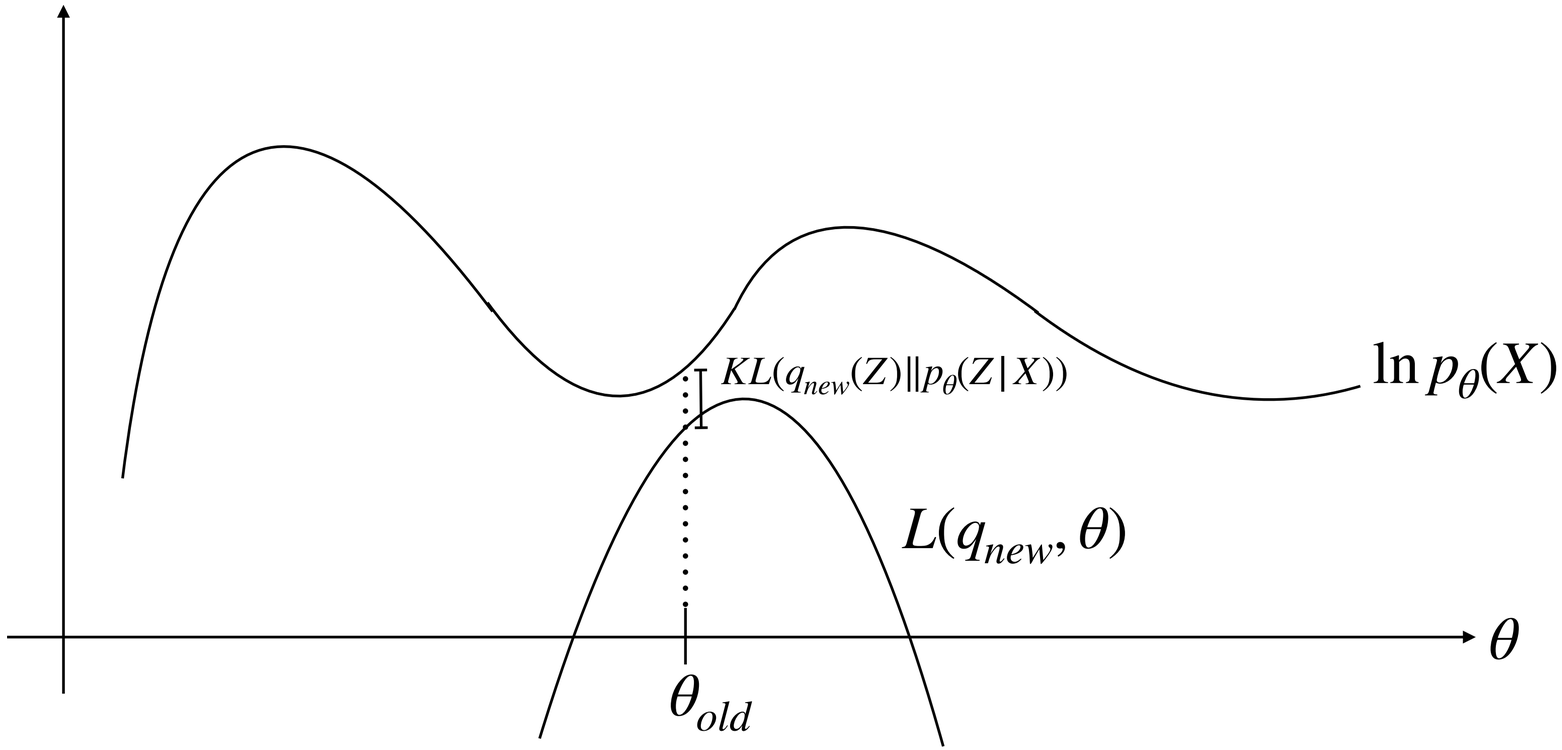


General EM

Variational E-step

~~Set $q_{new}(Z) = p_{\theta_{old}}(Z|X)$~~

Find the $q_{new}(Z)$ that maximize $L(q_{new}, \theta)$

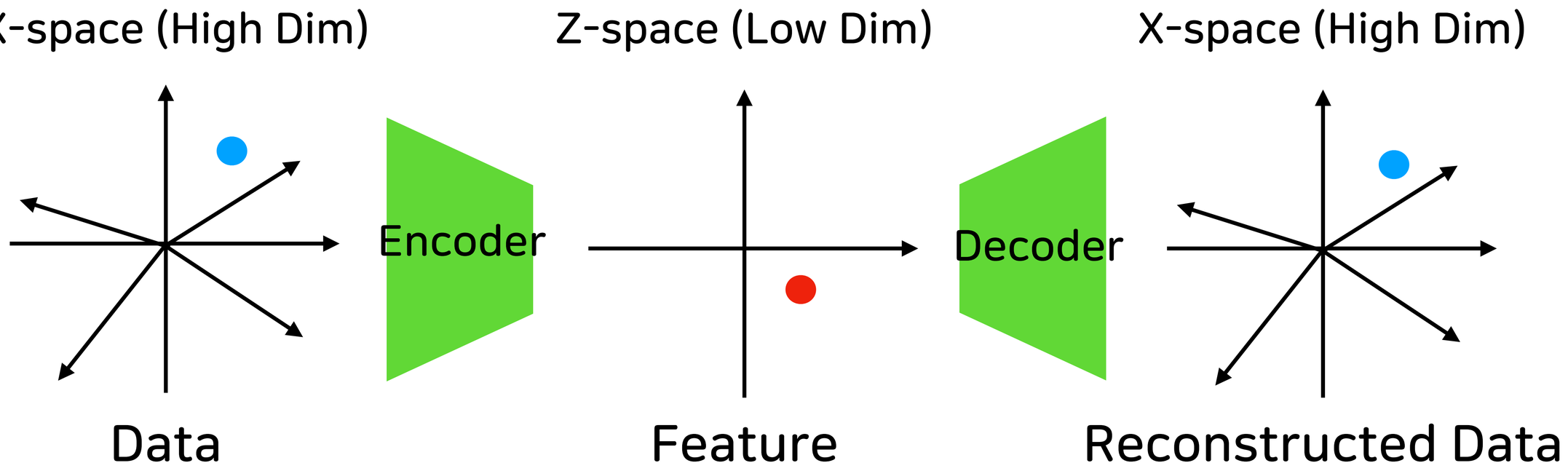


Auto-Encoder

Auto-Encoder(개념)

- Auto-Encoder는 Encoder를 통해 고차원의 데이터를 저차원으로 맵핑하고, Decoder를 통해 다시 복원하는 네트워크 입니다.

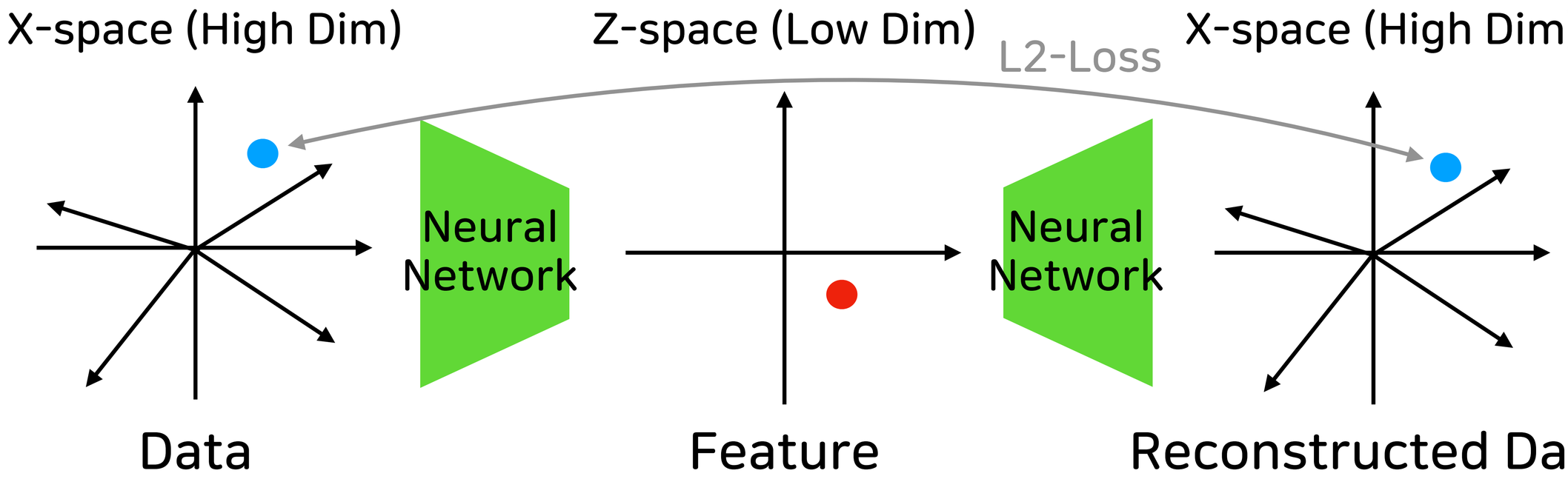
- 이렇게 저차원으로 맵핑된 정보는 데이터를 나타내는 feature로 활용될 수 있습니다.



Auto-Encoder(구성)

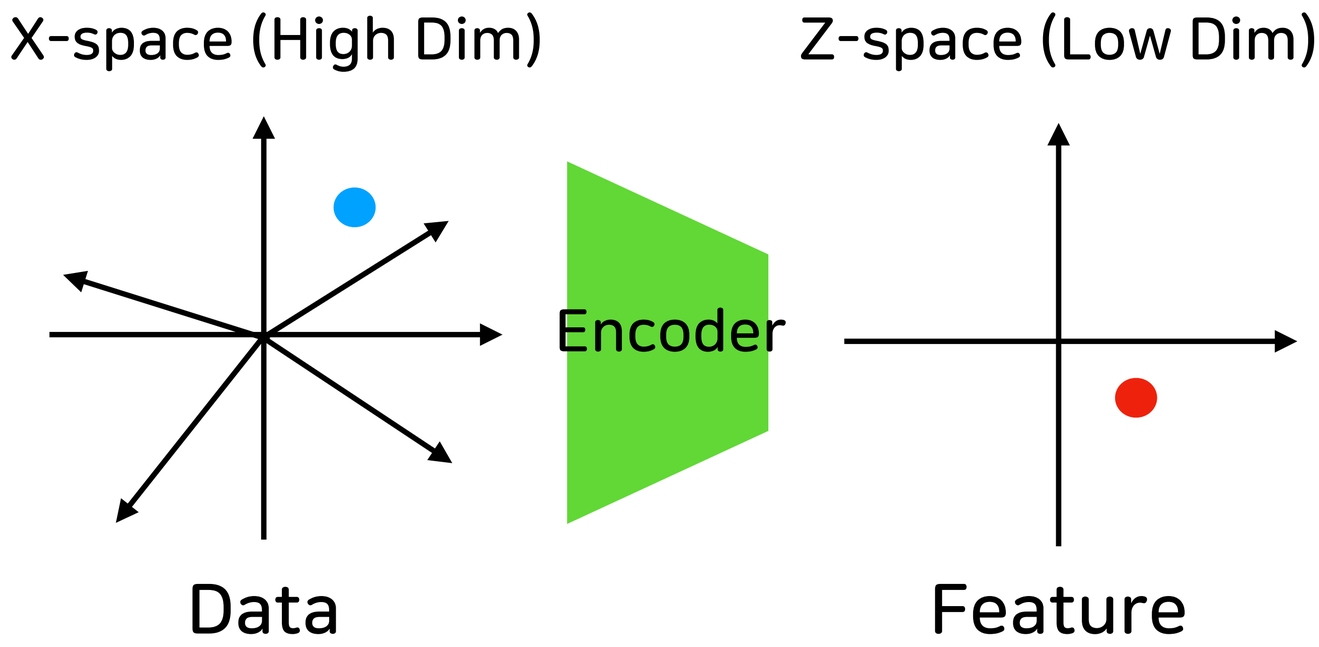
- Encoder와 Decoder는 Fully-Connected Layers 또는 Convolution Layers 등 Neural Networks로 구성합니다.

- input과 output 간에 L2 Loss를 취해 reconstruction을 할 수 있도록 합니다.



Auto-Encoder(Encoder)

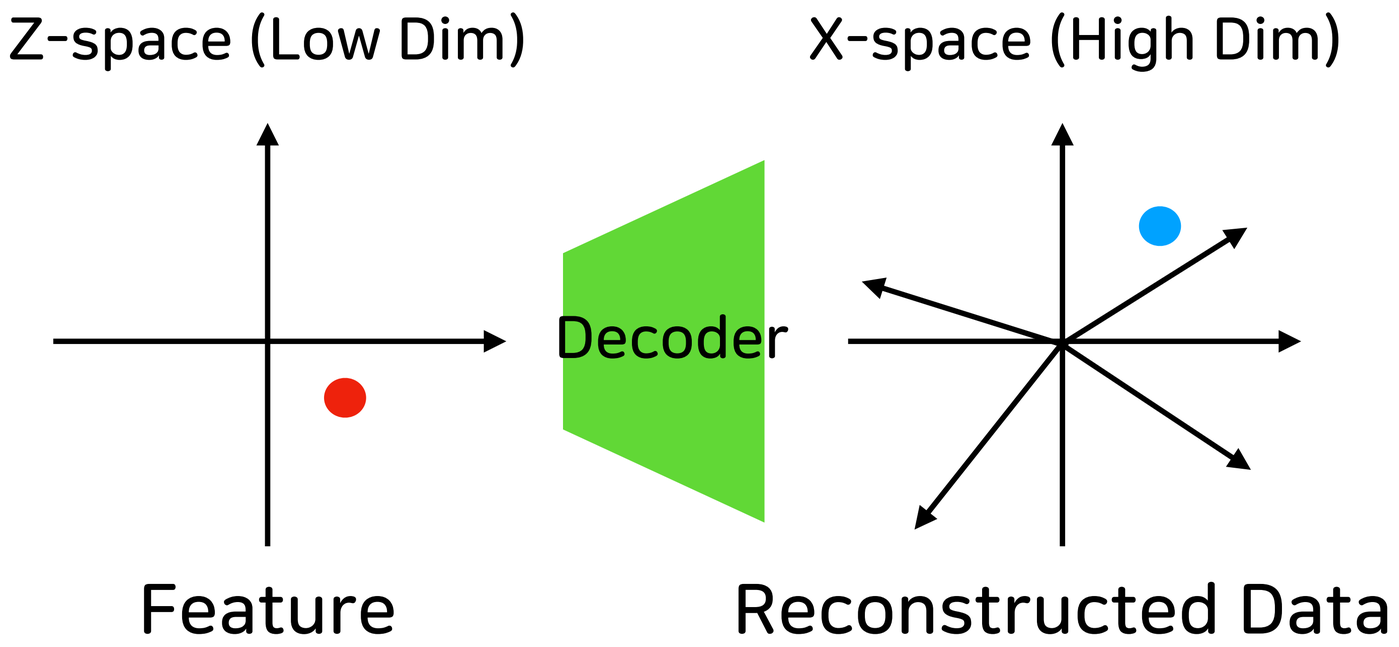
- 트레이닝 후 Encoder는 Feature Extraction의 용도로 활용될 수 있습니다.
- 이 때, Encoder는 고차원의 데이터를 압축하여 feature를 추출해내는 역할을 합니다.



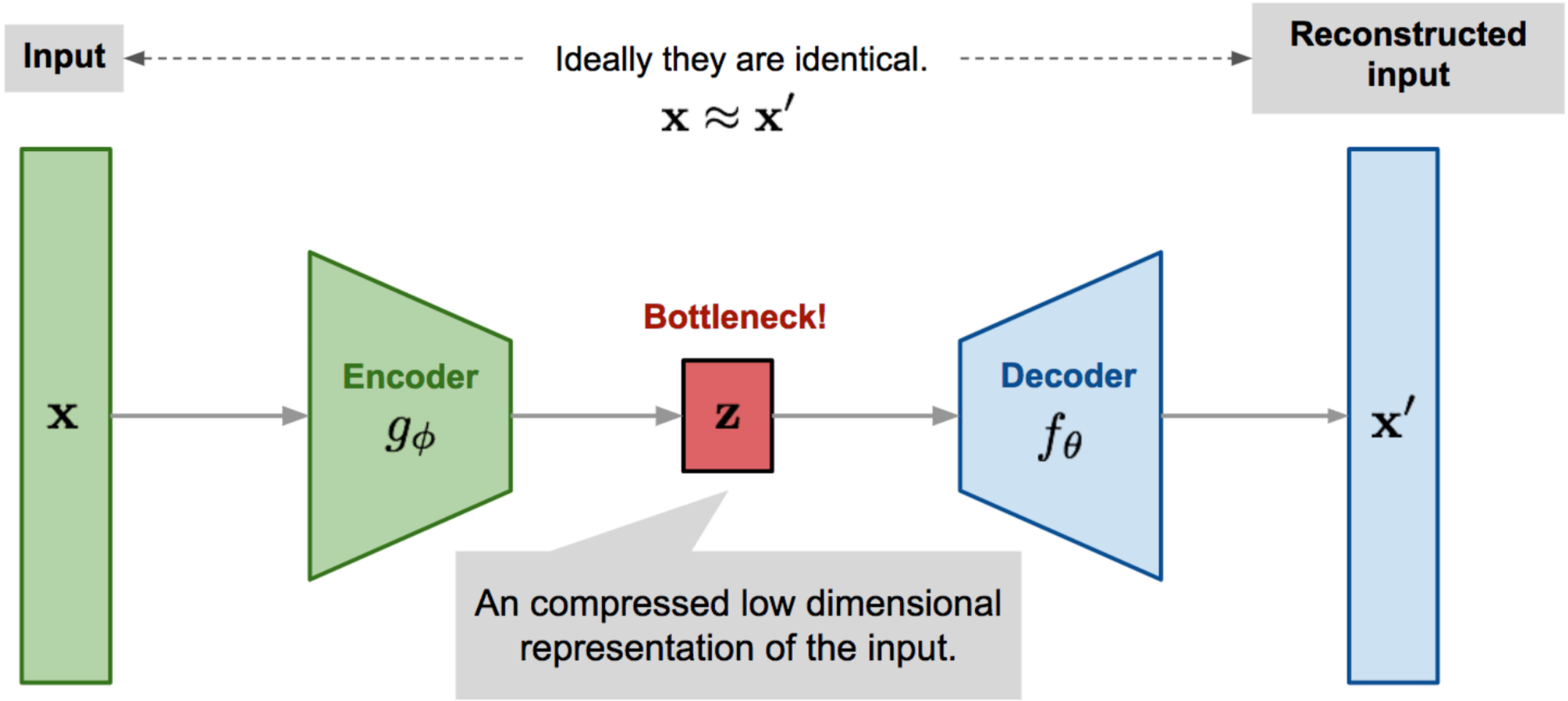
Auto-Encoder(Decoder)

- Z-space에서 벡터 하나를 샘플링하여 Decoder를 통해 X-space의 벡터로 복원하면 데이터를 생성해 낼 수 있습니다.

- 하지만 Z-space에서 벡터들의 집합이 어떤 분포를 가지고 있는지 Auto-Encoder 만으로는 알 수 없으므로 생성모델이라 부를 수 없습니다.

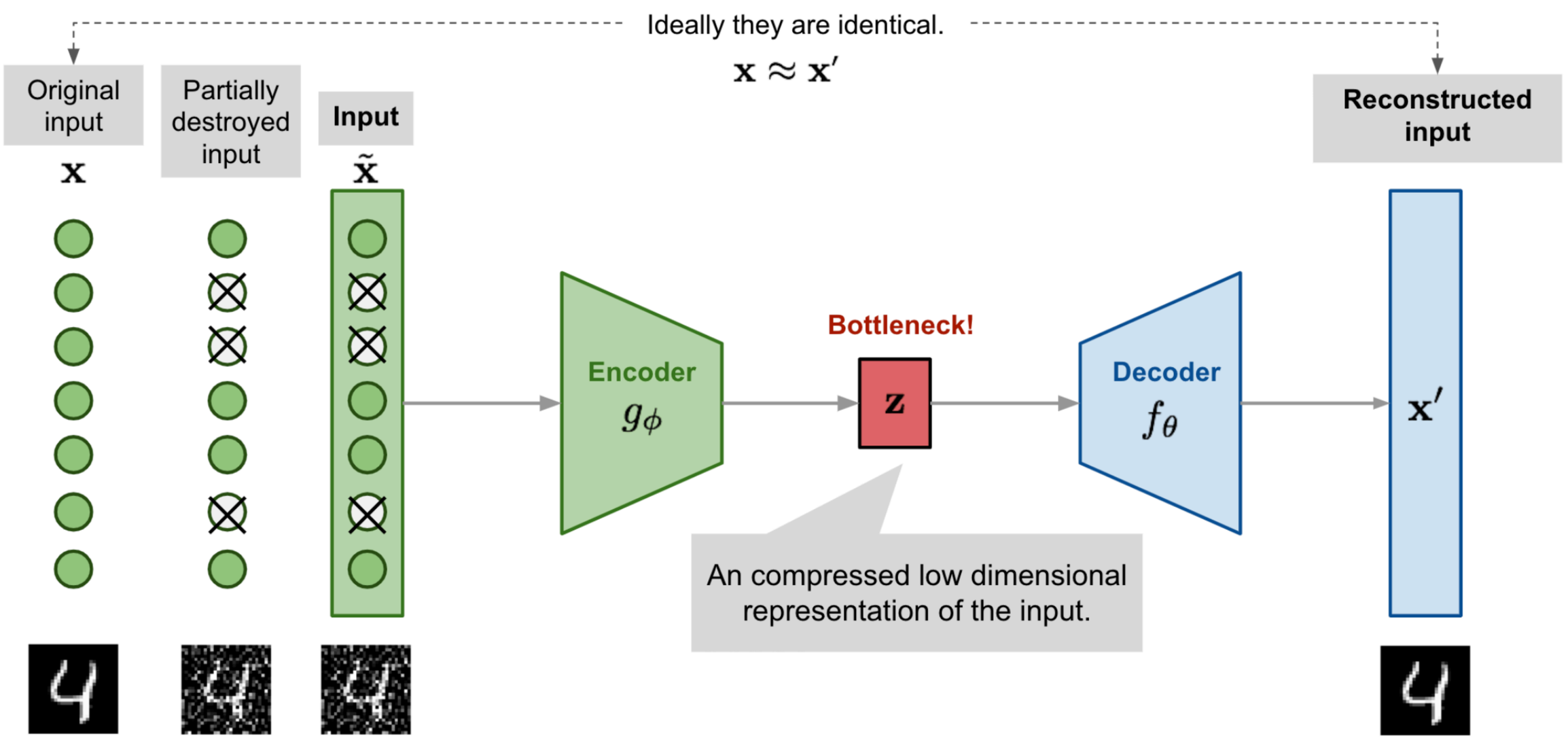


Auto-Encoder



lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Auto-Encoder



lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

딥러닝 에스프레소

딥러닝을 위한 베이지안 통계 Day3

VAE, BBB, MCDropout, Uncertainty, GAN

2022

멀티캠퍼스

박수철

Variational Auto-Encoders

Variational Auto-Encoders

- VAE (Variational Auto-Encoders)는 GAN과 더불어 딥러닝의 대표적인 생성모델(Generative Models)입니다.
- Auto-Encoder가 Z-variables의 분포를 알 수 없는 단점을 개선한 모델입니다.
- Z-variables의 분포를 Istropic Gaussian 등으로 제한하여 자유롭게 컨트롤할 수 있다는 특징을 가집니다.
- VAE는 또한 Probabilistic PCA의 conditional distribution이 linear transform 모델에 제한된다는 단점을 개선한 모델로 생각할 수 있습니다.
- Neural Networks의 non-linearity를 posterior와 conditional distribution을 형성하는데 사용해 기존 머신러닝에서 보여줄 수 없었던 유연한 표현 능력을 가집니다.

Variational Auto-Encoders

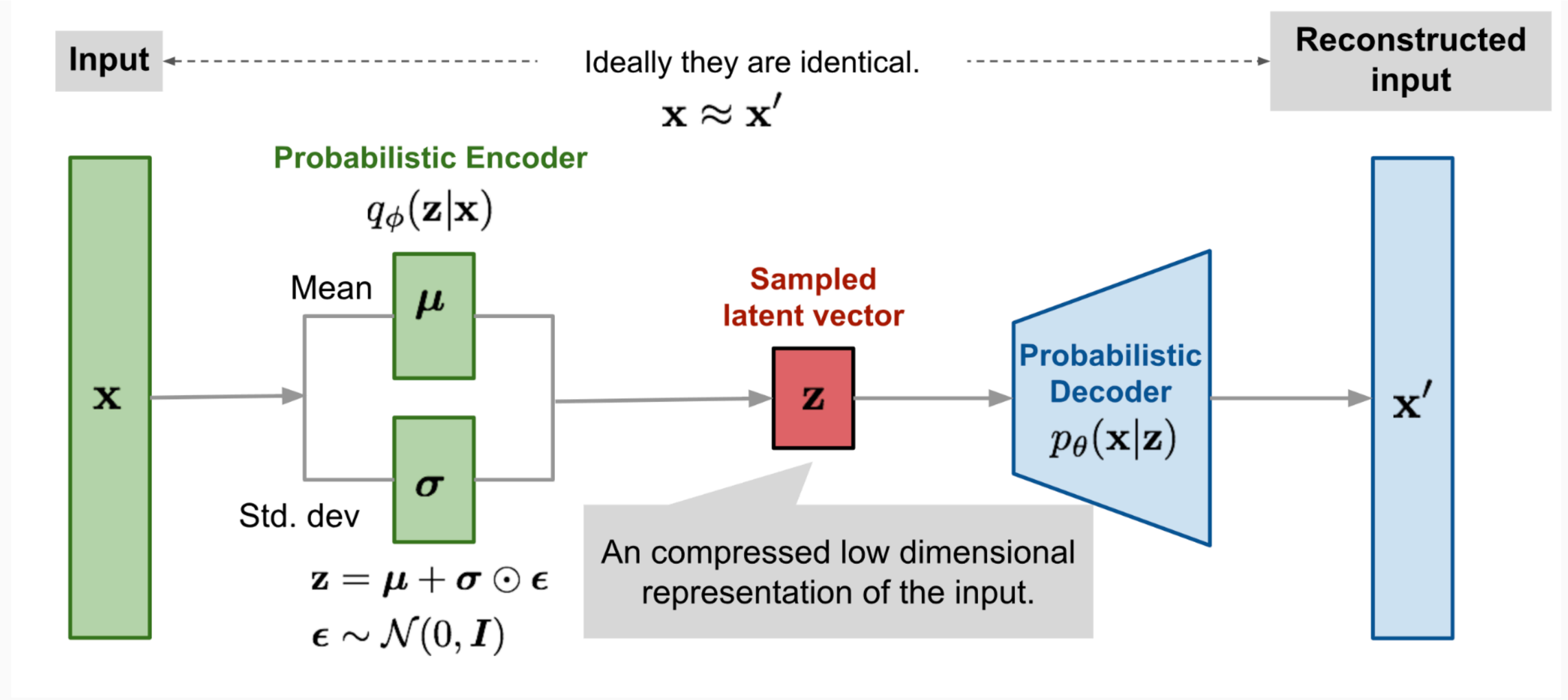


Fig. 9. Illustration of variational autoencoder model with the multivariate Gaussian assumption.

https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Variational Auto-Encoders

Called a variational method because it derives from the **Calculus of Variations**.

Functions:

- Variables as input, output is a value.
- Full and partial derivatives $\frac{df}{dx}$
- E.g., Maximise likelihood p(x|θ) w.r.t. parameters θ

Functionals:

- Functions as input, output is a value.
- Functional derivatives $\frac{\delta F}{\delta f}$
- E.g., Maximise the entropy H[*p(x)*] w.r.t. *p(x)*

We exploit both types of derivatives in variational inference.

Shakir Mohamed. Tutorial on Variational Inference for Machine Learning. https://www.shakirm.com/papers/VITutorial.pdf

Variational Auto-Encoders

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

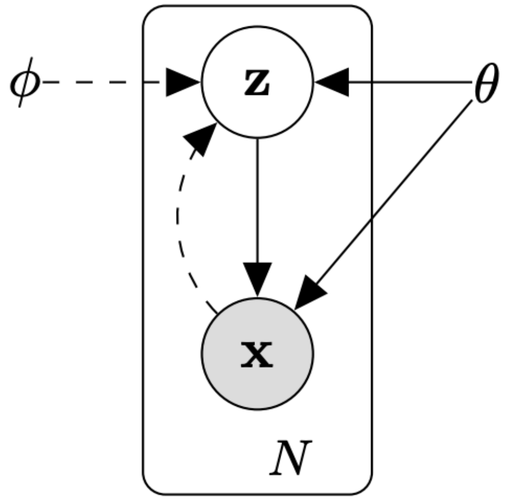


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE에서 해결하고자 하는 문제의 정의

2.1 Problem scenario

Let us consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable $\mathbf{x}$. We assume that the data are generated by some random process, involving an unobserved continuous random variable $\mathbf{z}$. The process consists of two steps: (1) a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p_{\boldsymbol{\theta}^*}(\mathbf{z})$; (2) a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$. We assume that the prior $p_{\boldsymbol{\theta}^*}(\mathbf{z})$ and likelihood $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$ come from parametric families of distributions $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, and that their PDFs are differentiable almost everywhere w.r.t. both $\boldsymbol{\theta}$ and $\mathbf{z}$. Unfortunately, a lot of this process is hidden from our view: the true parameters $\boldsymbol{\theta}^*$ as well as the values of the latent variables $\mathbf{z}^{(i)}$ are unknown to us.

Very importantly, we *do not* make the common simplifying assumptions about the marginal or posterior probabilities. Conversely, we are here interested in a general algorithm that even works efficiently in the case of:

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE에서 문제를 해결함에 있어 부딪히는 난관

1. *Intractability*: the case where the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is intractable (so we cannot evaluate or differentiate the marginal likelihood), where the true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable (so the EM algorithm cannot be used), and where the required integrals for any reasonable mean-field VB algorithm are also intractable. These intractabilities are quite common and appear in cases of moderately complicated likelihood functions $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, e.g. a neural network with a nonlinear hidden layer.

2. *A large dataset*: we have so much data that batch optimization is too costly; we would like to make parameter updates using small minibatches or even single datapoints. Sampling-based solutions, e.g. Monte Carlo EM, would in general be too slow, since it involves a typically expensive sampling loop per datapoint.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- 문제를 해결하기 위한 방법의 개요

For the purpose of solving the above problems, let us introduce a recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$: an approximation to the intractable true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Note that in contrast with the approximate posterior in mean-field variational inference, it is not necessarily factorial and its parameters $\boldsymbol{\phi}$ are not computed from some closed-form expectation. Instead, we'll introduce a method for learning the recognition model parameters $\boldsymbol{\phi}$ jointly with the generative model parameters $\boldsymbol{\theta}$.

From a coding theory perspective, the unobserved variables $\mathbf{z}$ have an interpretation as a latent representation or *code*. In this paper we will therefore also refer to the recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$ as a probabilistic *encoder*, since given a datapoint $\mathbf{x}$ it produces a distribution (e.g. a Gaussian) over the possible values of the code $\mathbf{z}$ from which the datapoint $\mathbf{x}$ could have been generated. In a similar vein we will refer to $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ as a probabilistic *decoder*, since given a code $\mathbf{z}$ it produces a distribution over the possible corresponding values of $\mathbf{x}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- ELBO(Evidence Lower BOund) 설명

2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

which can also be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \tag{3}$$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- 왜 ϕ에 대해 ELBO를 최대화하면 variational posterior $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$가 true posterior $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$를 근사하게 되는가?
-

$$p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) = \frac{p_\theta(\mathbf{z})p_\theta(\mathbf{x}^{(i)}|\mathbf{z})}{p_\theta(\mathbf{x}^{(i)})}$$

-

$$\arg\max_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log \frac{p_\theta(\mathbf{z})p_\theta(\mathbf{x}^{(i)}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right]$$

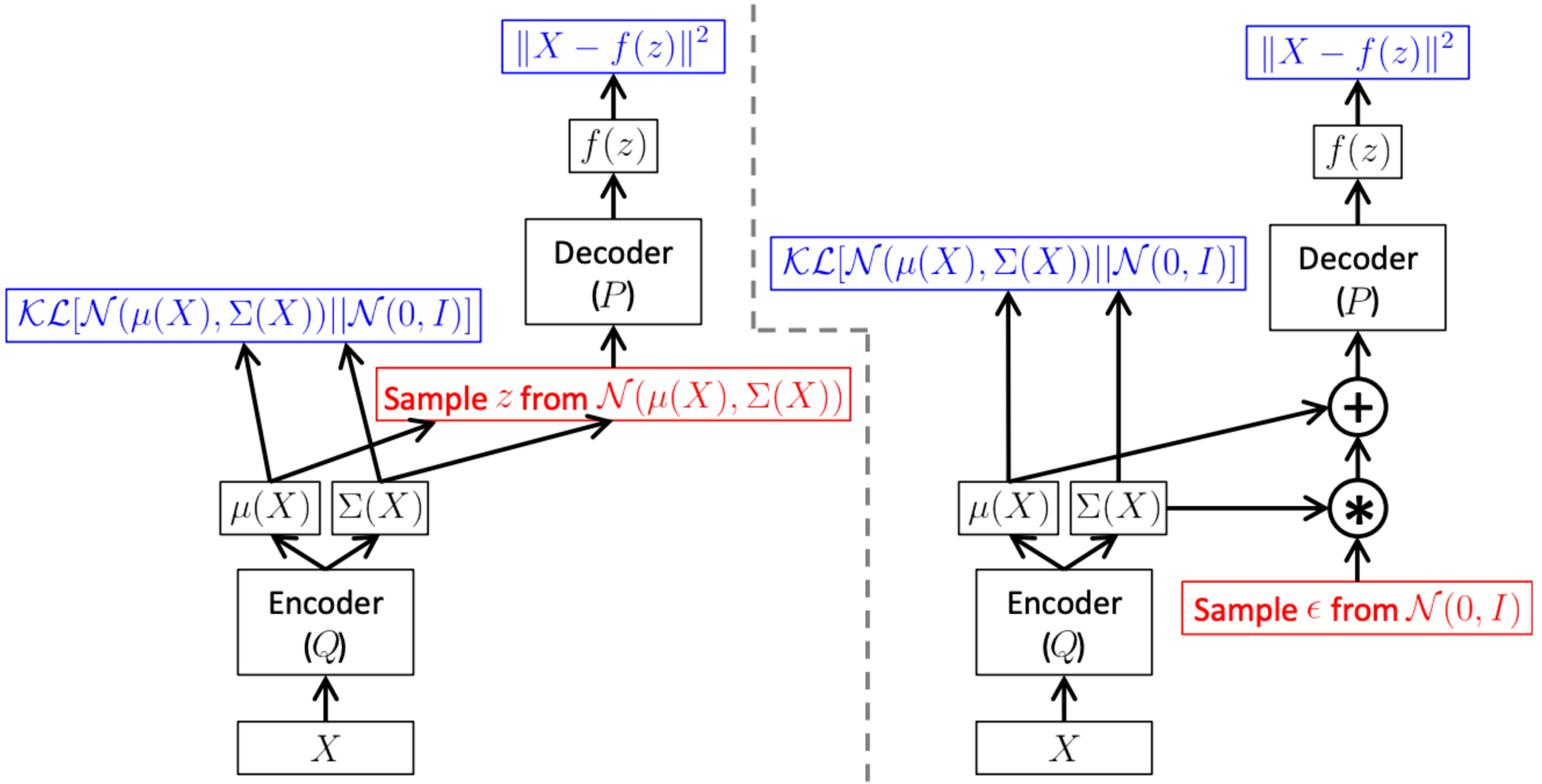
Variational Auto-Encoders

- ELBO를 ϕ에 대해 미분하고자 할 때 생기는 문제점 지적

We want to differentiate and optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. However, the gradient of the lower bound w.r.t. $\boldsymbol{\phi}$ is a bit problematic. The usual (naïve) Monte Carlo gradient estimator for this type of problem is: $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$ where $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. This gradient estimator exhibits exhibits very high variance (see e.g. [BJP12]) and is impractical for our purposes.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders



Carl Doersch, Tutorial on Variational Autoencoders

Variational Auto-Encoders

- Reparameterization trick과 Monte Carlo estimation을 이용하여 ELBO 계산

2.3 The SGVB estimator and AEVB algorithm

In this section we introduce a practical estimator of the lower bound and its derivatives w.r.t. the parameters. We assume an approximate posterior in the form $q_\phi(\mathbf{z}|\mathbf{x})$, but please note that the technique can be applied to the case $q_\phi(\mathbf{z})$, i.e. where we do not condition on $\mathbf{x}$, as well. The fully variational Bayesian method for inferring a posterior over the parameters is given in the appendix.

Under certain mild conditions outlined in section 2.4 for a chosen approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ we can reparameterize the random variable $\widetilde{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ of an (auxiliary) noise variable $\boldsymbol{\epsilon}$:

$$\widetilde{\mathbf{z}} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{4}$$

See section 2.4 for general strategies for chosing such an approriate distribution $p(\boldsymbol{\epsilon})$ and function $g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$. We can now form Monte Carlo estimates of expectations of some function $f(\mathbf{z})$ w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$ as follows:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_\phi(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{5}$$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB(Stochastic Gradient Variational Bayes) A type

We apply this technique to the variational lower bound (eq. (2)), yielding our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{6}$$

-> Bayes By Backprop.에서도 등장

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB(Stochastic Gradient Variational Bayes) B type
  posterior와 prior 간의 KL-Divergence를 closed form으로 계산

Often, the KL-divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))$ of eq. (3) can be integrated analytically (see appendix B), such that only the expected reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$ requires estimation by sampling. The KL-divergence term can then be interpreted as regularizing $\boldsymbol{\phi}$, encouraging the approximate posterior to be close to the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$. This yields a second version of the SGVB estimator $\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$, corresponding to eq. (3), which typically has less variance than the generic estimator:

$$\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \tag{7}$$

-> VAE에서 사용하는 일반적 방식

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB를 이용한 AEVB(Auto-Encoding Variational Bayes) 알고리즘 pseudo code

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- Dataset 전체의 likelihood, batch size와 sampling 갯수

Given multiple datapoints from a dataset $\mathbf{X}$ with N datapoints, we can construct an estimator of the marginal likelihood lower bound of the full dataset, based on minibatches:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{8}$$

where the minibatch $\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn sample of M datapoints from the full dataset $\mathbf{X}$ with N datapoints. In our experiments we found that the number of samples L per datapoint can be set to 1 as long as the minibatch size M was large enough, e.g. $M = 100$. Derivatives $\nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}; \mathbf{X}^M)$ can be taken, and the resulting gradients can be used in conjunction with stochastic optimization methods such as SGD or Adagrad [DHS10]. See algorithm 1 for a basic approach to compute the stochastic gradients.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- SGVB B-type의 직관적 해석

A connection with auto-encoders becomes clear when looking at the objective function given at eq. (7). The first term is (the KL divergence of the approximate posterior from the prior) acts as a regularizer, while the second term is a an expected negative reconstruction error. The function $g_{\boldsymbol{\phi}}(.)$ is chosen such that it maps a datapoint $\mathbf{x}^{(i)}$ and a random noise vector $\boldsymbol{\epsilon}^{(l)}$ to a sample from the approximate posterior for that datapoint: $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})$ where $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. Subsequently, the sample $\mathbf{z}^{(i,l)}$ is then input to function $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$, which equals the probability density (or mass) of datapoint $\mathbf{x}^{(i)}$ under the generative model, given $\mathbf{z}^{(i,l)}$. This term is a negative *reconstruction error* in auto-encoder parlance.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- VAE 모델의 구체적 예시

3 Example: Variational Auto-Encoder

In this section we'll give an example where we use a neural network for the probabilistic encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (the approximation to the posterior of the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$) and where the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are optimized jointly with the AEVB algorithm.

Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters. We let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be a multivariate Gaussian (in case of real-valued data) or Bernoulli (in case of binary data) whose distribution parameters are computed from $\mathbf{z}$ with a MLP (a fully-connected neural network with a single hidden layer, see appendix C). Note the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is in this case intractable. While there is much freedom in the form $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, we'll assume the true (but intractable) posterior takes on a approximate Gaussian form with an approximately diagonal covariance. In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure[2]:

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}) \tag{9}$$

where the mean and s.d. of the approximate posterior, $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$, are outputs of the encoding MLP, i.e. nonlinear functions of datapoint $\mathbf{x}^{(i)}$ and the variational parameters $\boldsymbol{\phi}$ (see appendix C).

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders

- ELBO 함수의 구체적 예시

As explained in section 2.4, we sample from the posterior $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ using $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With $\odot$ we signify an element-wise product. In this model both $p_{\boldsymbol{\theta}}(\mathbf{z})$ (the prior) and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ are Gaussian; in this case, we can use the estimator of eq. (7) where the KL divergence can be computed and differentiated without estimation (see appendix B). The resulting estimator for this model and datapoint $\mathbf{x}^{(i)}$ is:

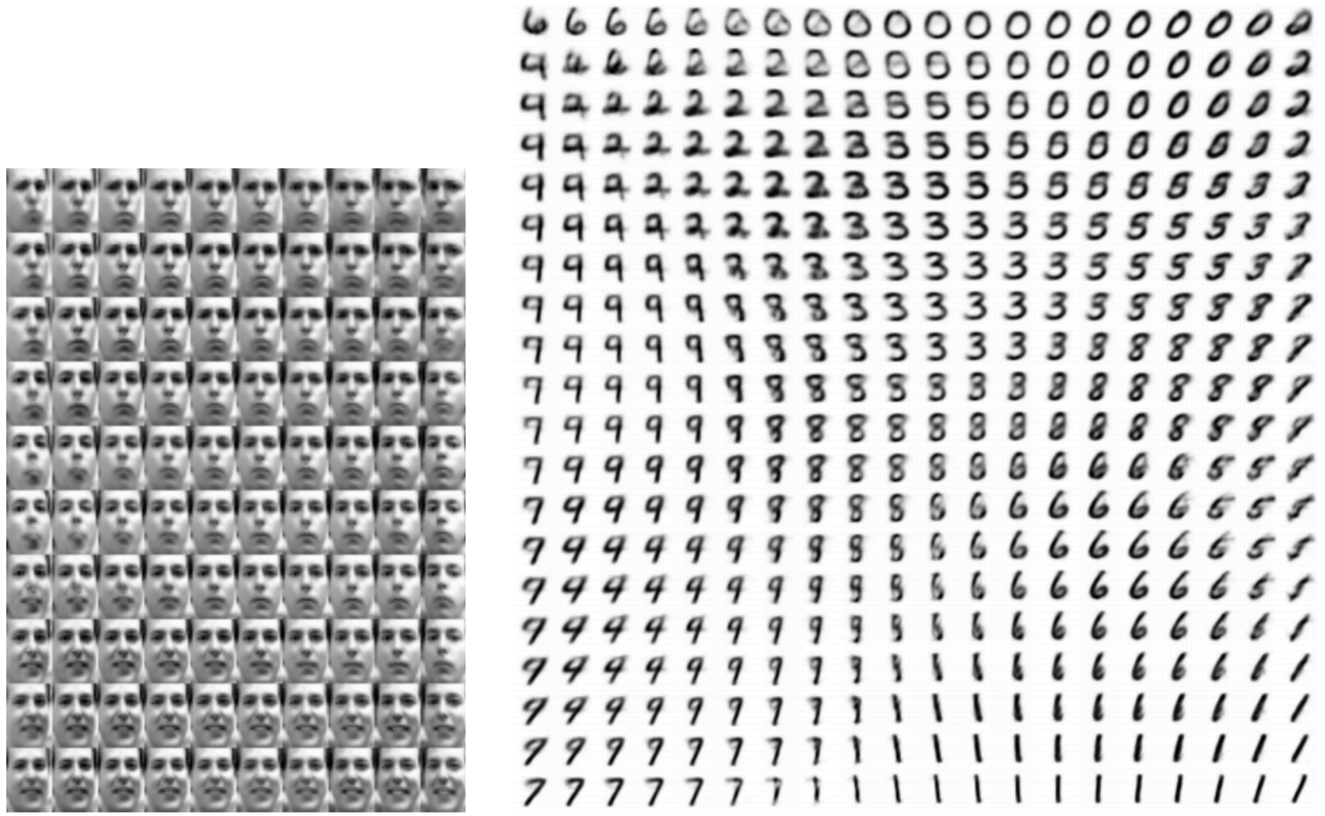
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

As explained above and in appendix C, the decoding term $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$ is a Bernoulli or Gaussian MLP, depending on the type of data we are modelling.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders



(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

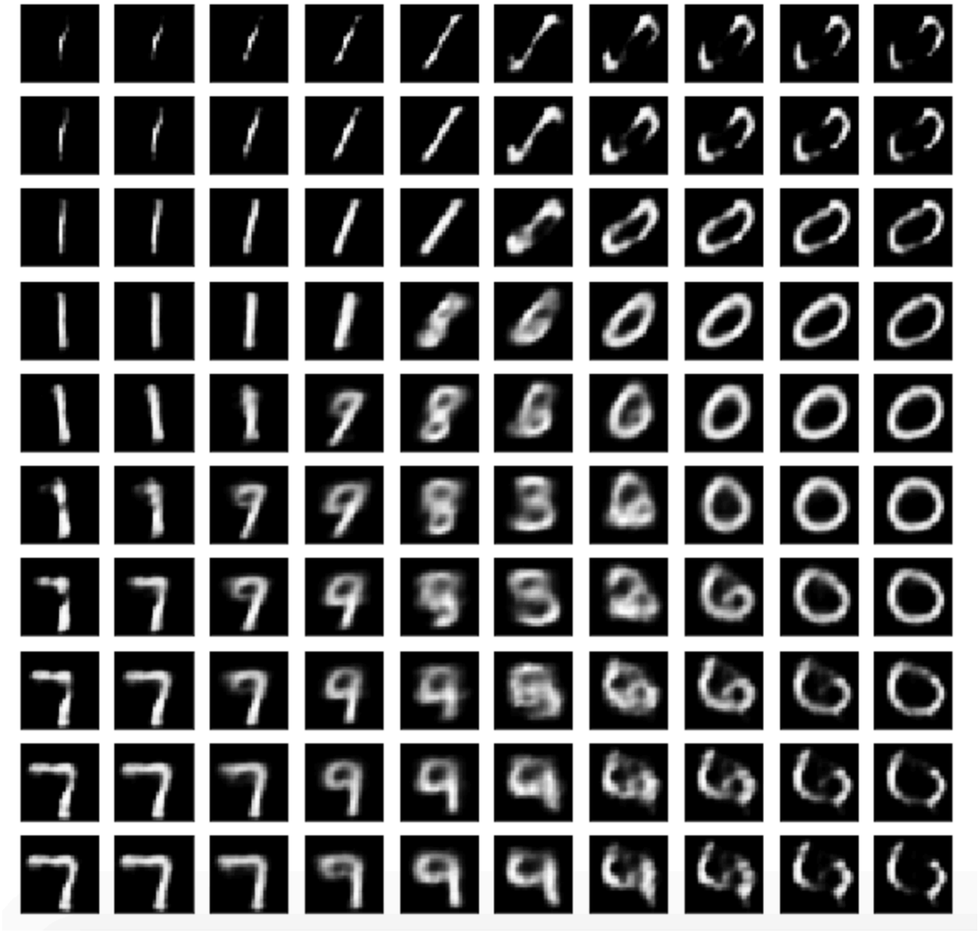
Variational Auto-Encoders

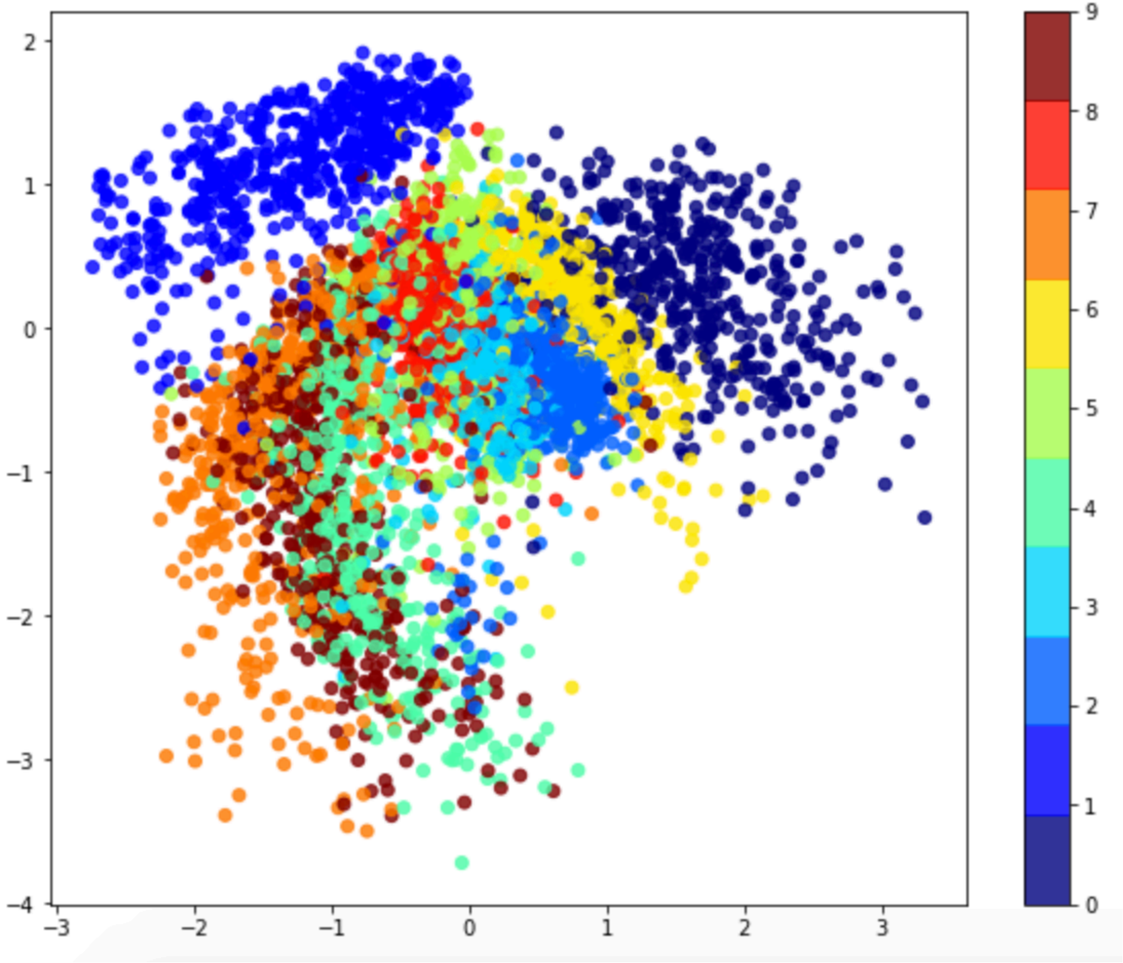
(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes

Variational Auto-Encoders





Bayes By Backprop.

Bayes By Backprop.

- VAE에서 latent variables Z의 posterior를 구하기 위해 variational inference를 사용했듯이 Bayes by Backprop.에서는 weight의 posterior를 구하기 위해 variational inference를 사용합니다.

- VAE에서와 마찬가지로 ELBO를 최대화하는 방식으로 트레이닝이 진행되며, neural networks에서 사용하는 standard backpropagation을 이용하기 위해 reparameterization trick을 사용합니다.

- weight의 posterior를 얻음으로써 predictive distribution을 얻을 수 있다는 장점이 있습니다. 하지만 Bayesian linear regression과 달리 analytical한 형태의 distribution을 얻을 수는 없고 많은 샘플링을 통해 mean과 variance를 측정할 수 있습니다.

- Predictive distribution을 얻음으로써 예측에 대한 uncertainty를 얻어낼 수 있습니다.

Bayes By Backprop.

Abstract

We introduce a new, efficient, principled and backpropagation-compatible algorithm for learning a probability distribution on the weights of a neural network, called *Bayes by Backprop*. It regularises the weights by minimising a compression cost, known as the variational free energy or the expected lower bound on the marginal likelihood. We show that this principled kind of regularisation yields comparable performance to dropout on MNIST classification. We then demonstrate how the learnt uncertainty in the weights can be used to improve generalisation in non-linear regression problems, and how this weight uncertainty can be used to drive the exploration-exploitation trade-off in reinforcement learning.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- 일반적인 deep learning model과의 차이점

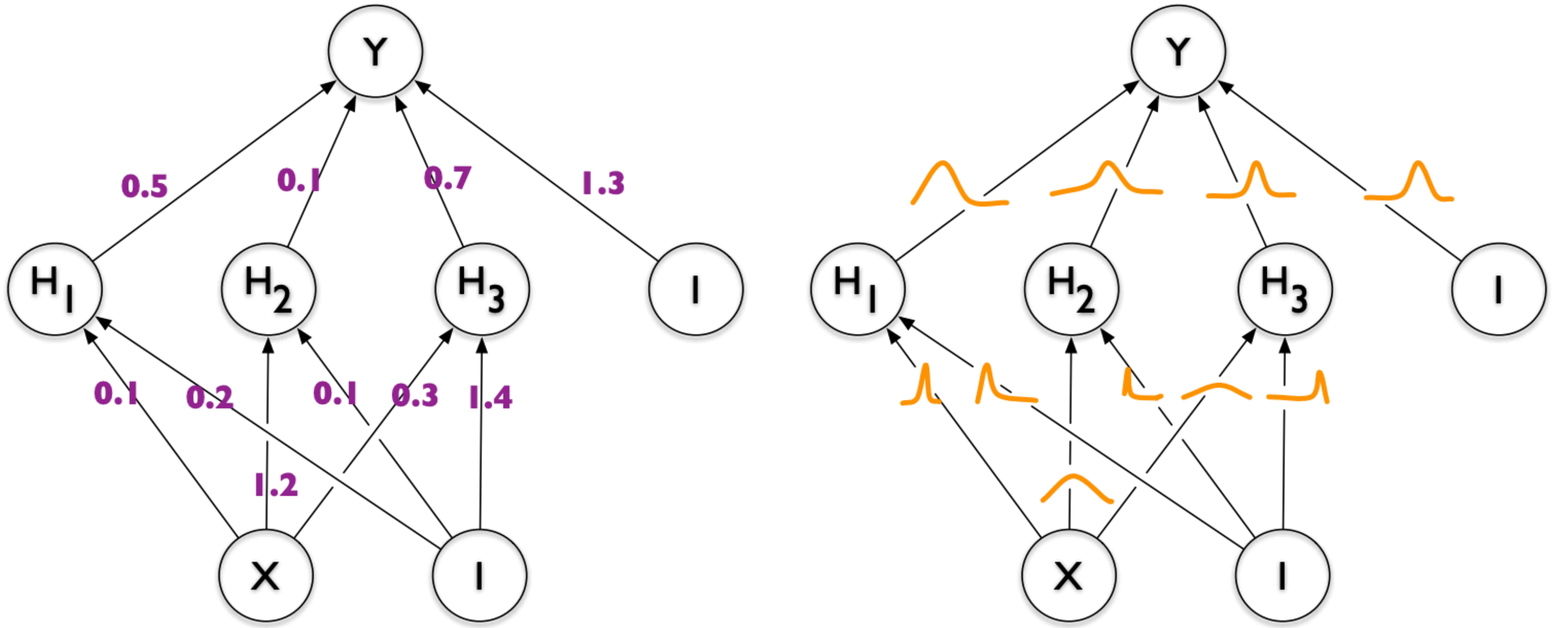


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Posterior, Predictive Distrubution, Ensemble

3. Being Bayesian by Backpropagation

Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data, $P(\mathbf{w}|\mathcal{D})$. This distribution answers predictive queries about unseen data by taking expectations: the predictive distribution of an unknown label $\hat{\mathbf{y}}$ of a test data item $\hat{\mathbf{x}}$, is given by $P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$. Each possible configuration of the weights, weighted according to the posterior distribution, makes a prediction about the unknown label given the test data item $\hat{\mathbf{x}}$. Thus taking an expectation under the posterior distribution on weights is equivalent to using an ensemble of an uncountably infinite number of neural networks. Unfortunately, this is intractable for neural networks of any practical size.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Optimal Paramaters

Previously Hinton and Van Camp (1993) and Graves (2011) suggested finding a variational approximation to the Bayesian posterior distribution on the weights. Variational learning finds the parameters θ of a distribution on the weights $q(\mathbf{w}|\theta)$ that minimises the Kullback-Leibler (KL)

divergence with the true Bayesian posterior on the weights:

variational posterior, true posterior

$$\theta^\star = \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})]$$

$$= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathbf{dw}$$

Negative ELBO

$$= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].$$

variational posterior; prior; log-likelihood

$$P(D) = ELBO + KL(q||p)$$

$$ELBO = \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log \frac{P(\mathbf{w})P(D|\mathbf{w})}{q(\mathbf{w}|\theta)}\right]$$

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- ELBO (Evidence Lower BOund)

The resulting cost function is variously known as the variational free energy (Neal and Hinton, 1998; Yedidia et al., 2000; Friston et al., 2007) or the expected lower bound (Saul et al., 1996; Neal and Hinton, 1998; Jaakkola and Jordan, 2000). For simplicity we shall denote it as

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]. \quad (1)$$

Negative ELBO

The cost function of (1) is a sum of a data-dependent part, which we shall refer to as the likelihood cost, and a prior-dependent part, which we shall refer to as the complexity cost. The cost function embodies a trade-off between satisfying the complexity of the data $\mathcal{D}$ and satisfying the simplicity prior $P(\mathbf{w})$. (1) is also readily given an information theoretic interpretation as a minimum description length cost (Hinton and Van Camp, 1993; Graves, 2011). Exactly minimising this cost naïvely is computationally prohibitive. Instead gradient descent and various approximations are used.

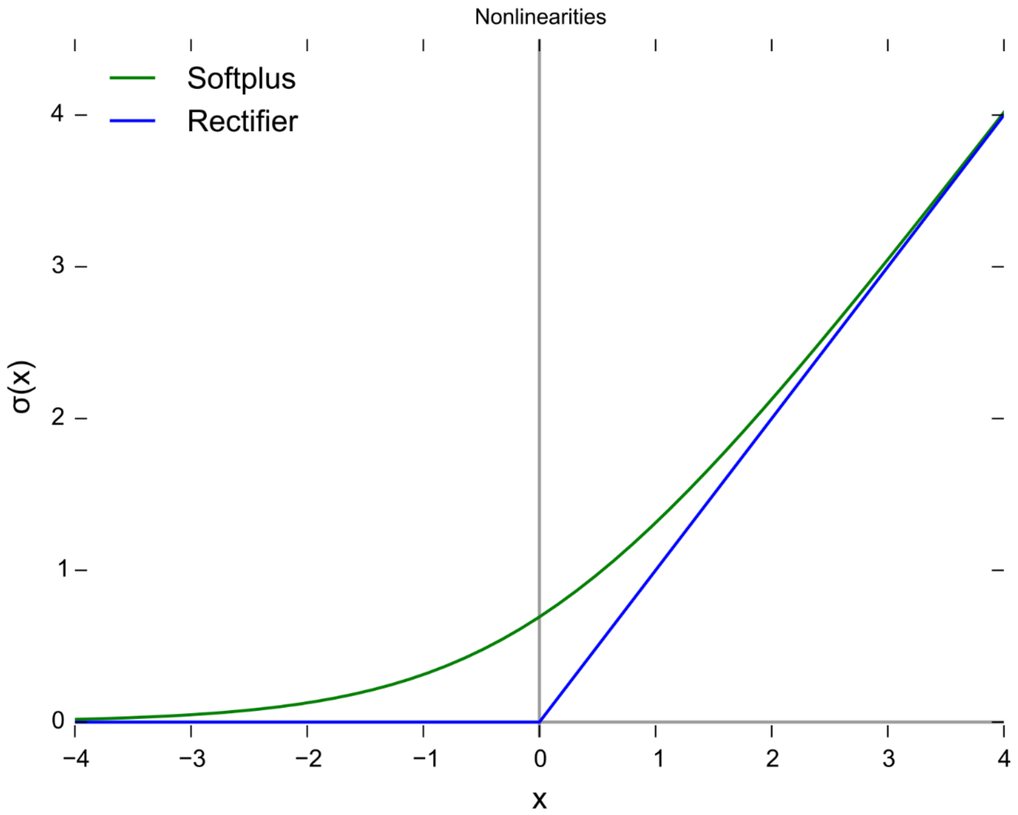
Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Reparameterization Trick

3.2. Gaussian variational posterior

Suppose that the variational posterior is a diagonal Gaussian distribution, then a sample of the weights $\mathbf{w}$ can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ. We parameterise the standard deviation pointwise as $\sigma = \log(1 + \exp(\rho))$ and so σ is always non-negative. The variational posterior parameters are $\theta = (\mu, \rho)$. Thus the transform from a sample of parameter-free noise and the variational posterior parameters that yields a posterior sample of the weights $\mathbf{w}$ is: $\mathbf{w} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$ where $\circ$ is pointwise multiplication. Each step of optimisation proceeds as follows:



https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Training Steps

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Let $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \tag{3}$$

6. Calculate the gradient with respect to the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \tag{4}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{5}$$
$$\rho \leftarrow \rho - \alpha\Delta_\rho. \tag{6}$$

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

- Scale mixture prior

We propose using a scale mixture of two Gaussian densities as the prior. Each density is zero mean, but differing variances:

$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}(\mathbf{w}_j|0,\sigma_1^2) + (1-\pi)\mathcal{N}(\mathbf{w}_j|0,\sigma_2^2), \quad (7)$$

where $\mathbf{w}_j$ is the jth weight of the network, $\mathcal{N}(x|\mu,\sigma^2)$ is the Gaussian density evaluated at x with mean μ and variance σ^2 and σ_1^2 and σ_2^2 are the variances of the mixture components. The first mixture component of the prior is given a larger variance than the second, $\sigma_1 > \sigma_2$, providing a heavier tail in the prior density than a plain Gaussian prior. The second mixture component has a small variance $\sigma_2 \ll 1$ causing many of the weights to *a priori* tightly concentrate around zero. Our prior resembles a spike-and-slab prior (Mitchell and Beauchamp, 1988; George and McCul-

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

Table 1. Classification Error Rates on MNIST. $\star$ indicates result used an ensemble of 5 networks.

Method	# Units/Layer	# Weights	Test Error
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			$\approx$ 1.3%
SGD, dropconnect (Wan et al., 2013)	800	1.3m	**1.2**%*
SGD	400	500k	1.83%
	800	1.3m	1.84%
	1200	2.4m	1.88%
SGD, dropout	400	500k	1.51%
	800	1.3m	1.33%
	1200	2.4m	1.36%
Bayes by Backprop, Gaussian	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes by Backprop, Scale mixture	400	500k	1.36%
	800	1.3m	1.34%
	1200	2.4m	**1.32**%

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

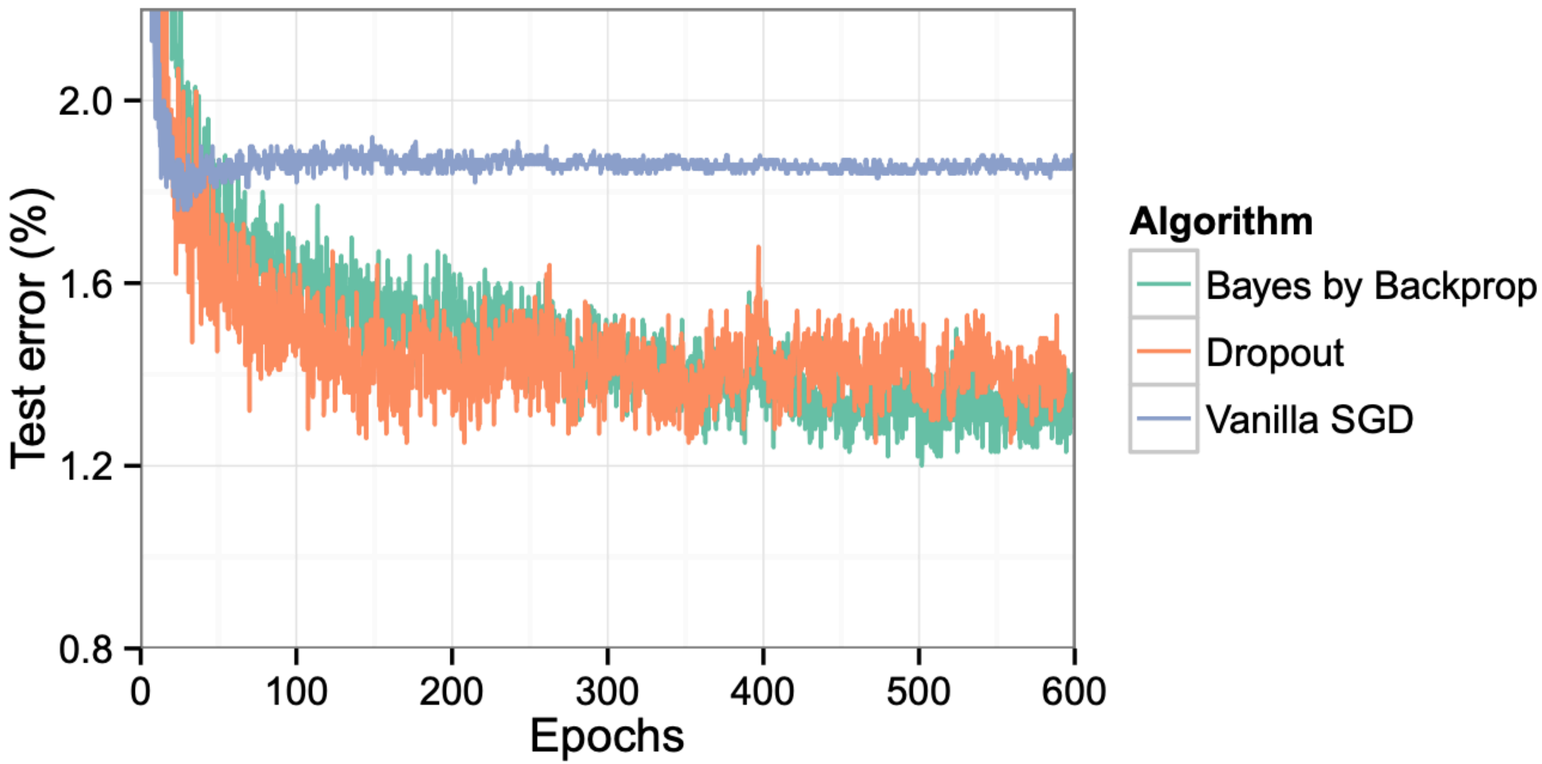


Figure 2. Test error on MNIST as training progresses.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

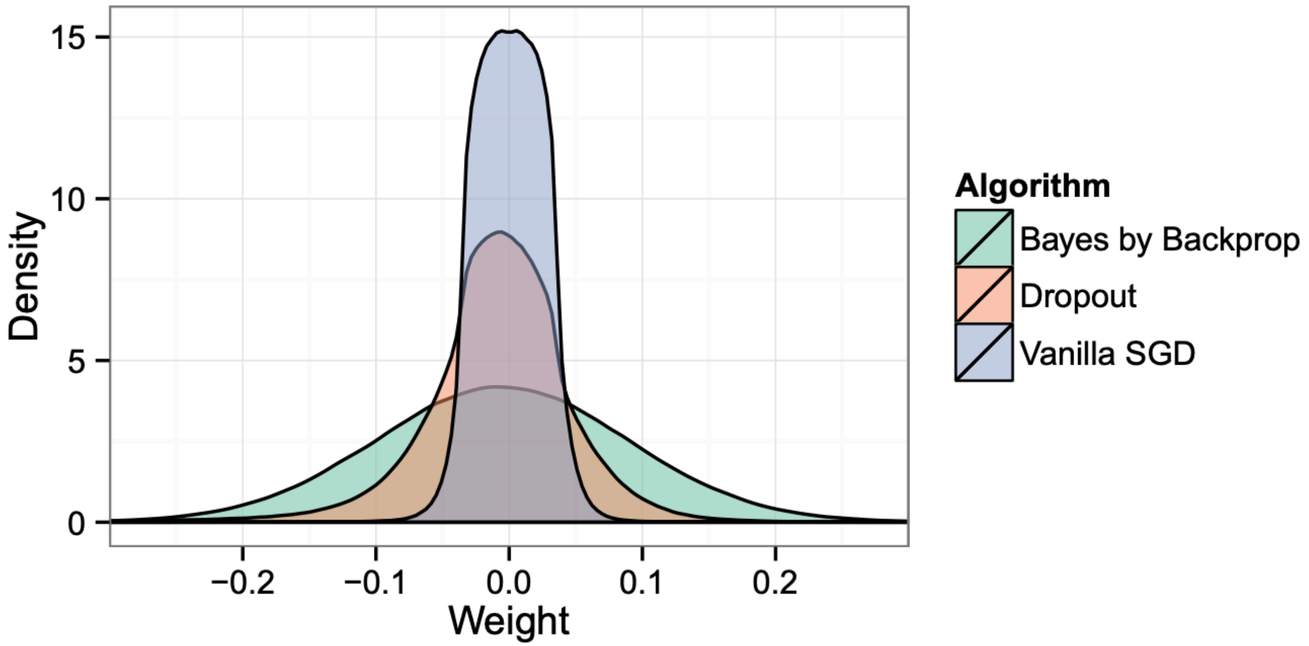


Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

"We took a Bayes by Backprop trained network with two layers of 1200 units and ordered the weights by their signal–to–noise ratio ($|\mu_i|/\sigma_i$). We removed the weights with the lowest signal to noise ratio."

Table 2. Classification Errors after Weight pruning

Proportion removed	# Weights	Test Error
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

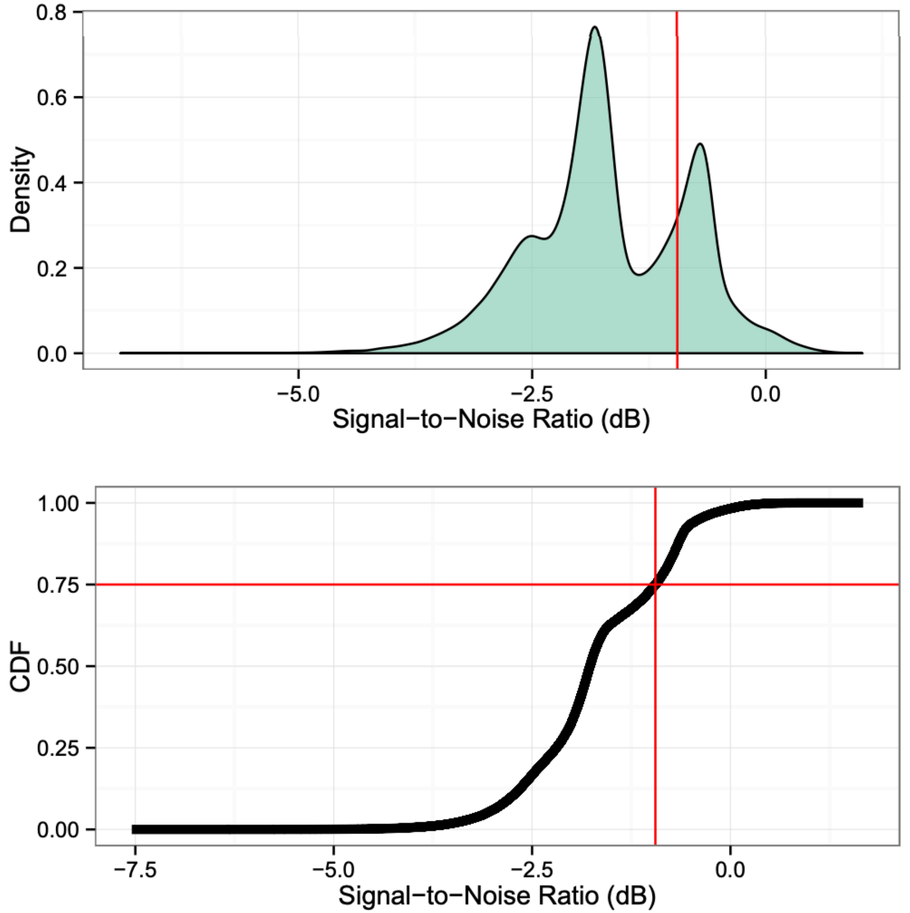


Figure 4. Density and CDF of the Signal-to-Noise ratio over all weights in the network. The red line denotes the 75% cut-off.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayes By Backprop.

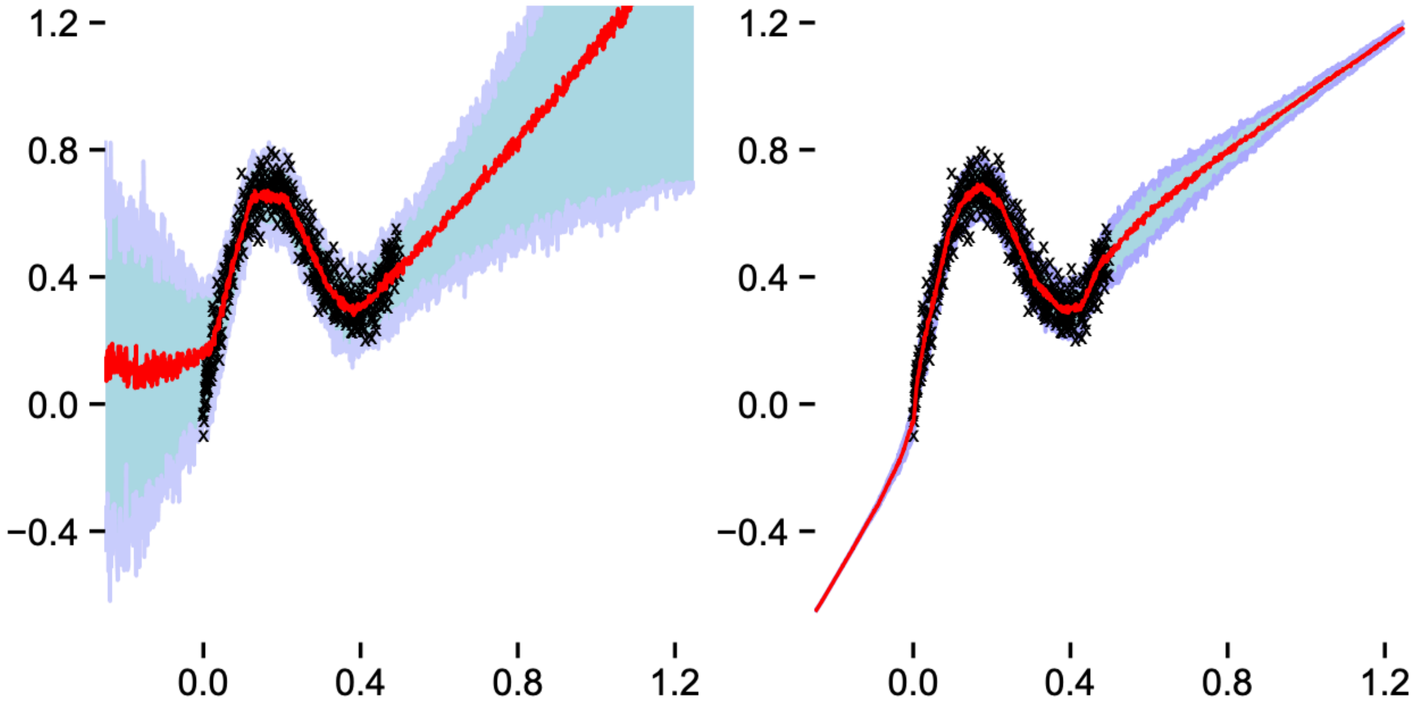
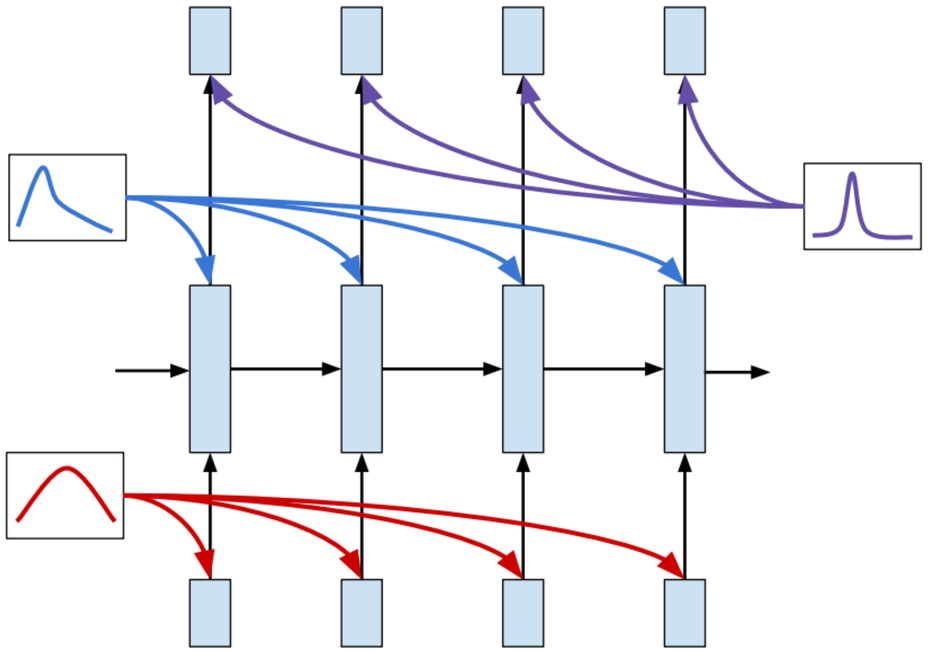


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Charles Blundell et al., Weight Uncertainty in Neural Networks

Bayesian RNN



Algorithm: Bayes by Backprop for RNNs

Sample $\epsilon \sim \mathcal{N}(0, I)$, $\epsilon \in \mathbb{R}^d$, and set network parameters to $\theta = \mu + \sigma\epsilon$.
Sample a minibatch of truncated sequences (x, y).
Do forward and backward propagation as normal, and let g be the gradient w.r.t θ.
Let $g_\theta^{KL}, g_\mu^{KL}, g_\sigma^{KL}$ be the gradients of $\log \mathcal{N}(\theta|\mu, \sigma^2) - \log p(\theta)$ w.r.t. θ, μ and σ respectively.
Update μ using the gradient $\frac{g + \frac{1}{C} g_\theta^{KL}}{B} + \frac{g_\mu^{KL}}{BC}$.
Update σ using the gradient $\left(\frac{g + \frac{1}{C} g_\theta^{KL}}{B}\right)\epsilon + \frac{g_\sigma^{KL}}{BC}$.

Figure 1: Illustration (left) and Algorithm (right) of Bayes by Backprop applied to an RNN.

Meire Fortunato et al., Bayesian Recurrent Neural Networks

Bayesian CNN

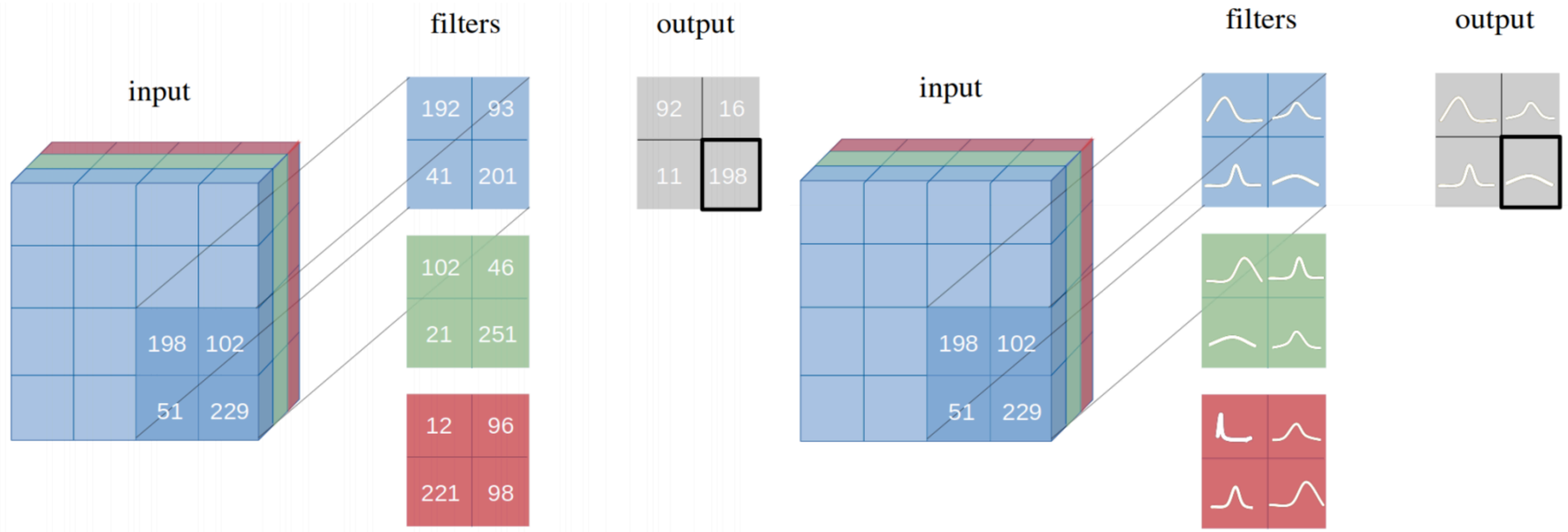
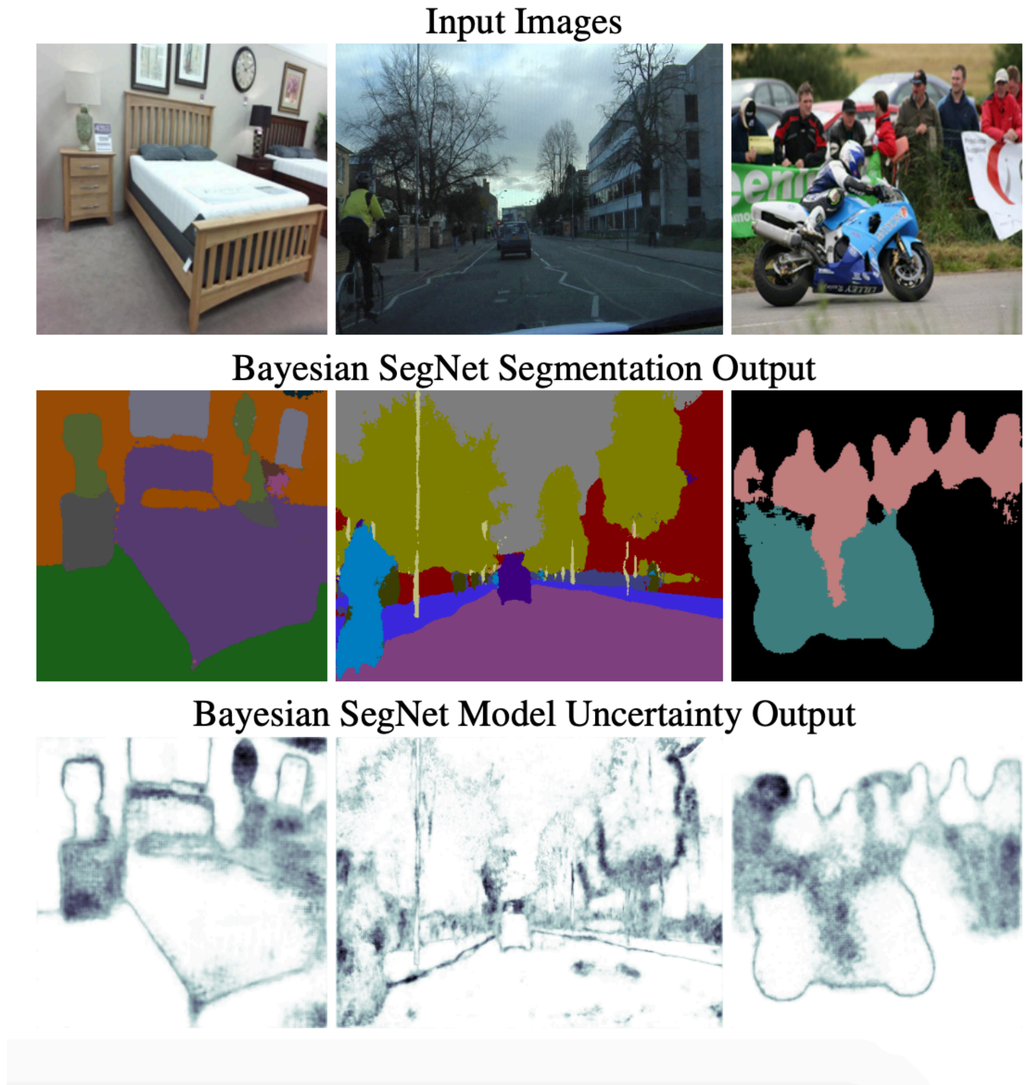


Figure 4: Input image with exemplary pixel values, filters, and corresponding output with point-estimates (top) and probability distributions (bottom) over weights.[55]

Kumar Shridhar et al., A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference

Bayesian SegNet



Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

Bayesian SegNet

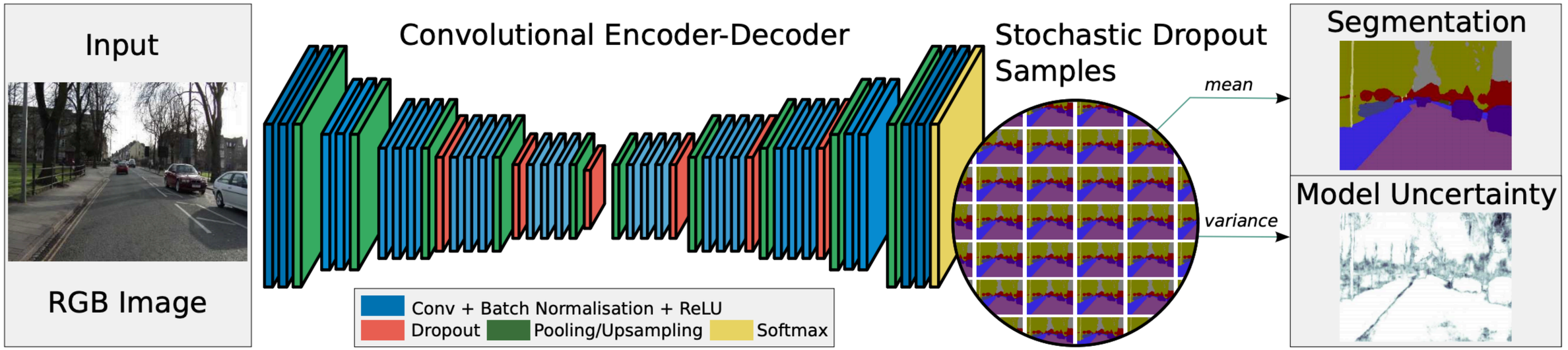
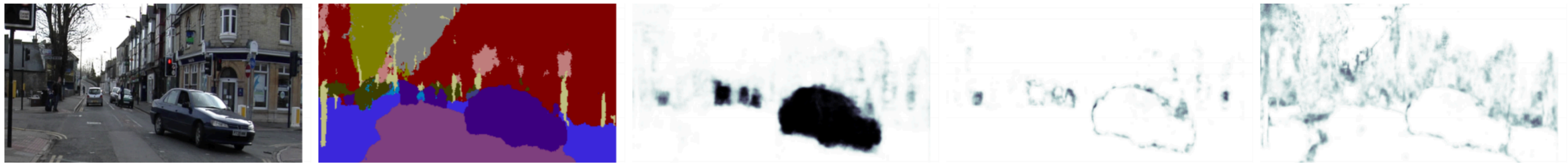


Figure 2: **A schematic of the Bayesian SegNet architecture.** This diagram shows the entire pipeline for the system which is trained end-to-end in one step with stochastic gradient descent. The encoders are based on the 13 convolutional layers of the VGG-16 network [34], with the decoder placing them in reverse. The probabilistic output is obtained from Monte Carlo samples of the model with dropout at test time. We take the variance of these softmax samples as the model uncertainty for each class.

Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

Bayesian SegNet



(a) Input Image (b) Semantic Segmentation (c) Softmax Uncertainty Car Class (d) Dropout Uncertainty Car Class (e) Dropout Uncertainty All Classes

Figure 3: **Comparison of uncertainty with Monte Carlo dropout and uncertainty from softmax regression (c-e: darker colour represents larger value).** This figure shows that softmax regression is only capable of inferring relative probabilities between classes. In contrast, dropout uncertainty can produce an estimate of absolute model uncertainty.

Alex Kendall et al., Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding

MCDropout

Dropout as a Bayesian Approximation

MCDropout

Abstract

Deep learning tools have gained tremendous attention in applied machine learning. However such tools for regression and classification do not capture model uncertainty. In comparison, Bayesian models offer a mathematically grounded framework to reason about model uncertainty, but usually come with a prohibitive computational cost. In this paper we develop a new theoretical framework casting dropout training in deep neural networks (NNs) as approximate Bayesian inference in deep Gaussian processes. A direct result of this theory gives us tools to model uncertainty with dropout NNs – extracting information from existing models that has been thrown away so far. This mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy. We perform an extensive study of the properties of dropout's uncertainty. Various network architectures and nonlinearities are assessed on tasks of regression and classification, using MNIST as an example. We show a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods, and finish by using dropout's uncertainty in deep reinforcement learning.

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

MCDropout

- Variational inference를 이용한 Bayesian neural networks는 구현하는데 번거로운 점이 있습니다.
- MCDropout (Monte Carlo Dropout)은 dropout을 이용하여 Bayesian neural networks와 같은 효과를 내도록합니다.
- Posterior에서 sampling하는 작업을 dropout으로 대신하고, Prior를 설정하던 것을 L2 regularisation으로 대체합니다.
- Loss function은 다음과 같이 Error term과 regularization term으로 구성됩니다.

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N}\sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L}\left(\|\mathbf{W}_i\|_2^2 + \|\mathbf{b}_i\|_2^2\right)$$

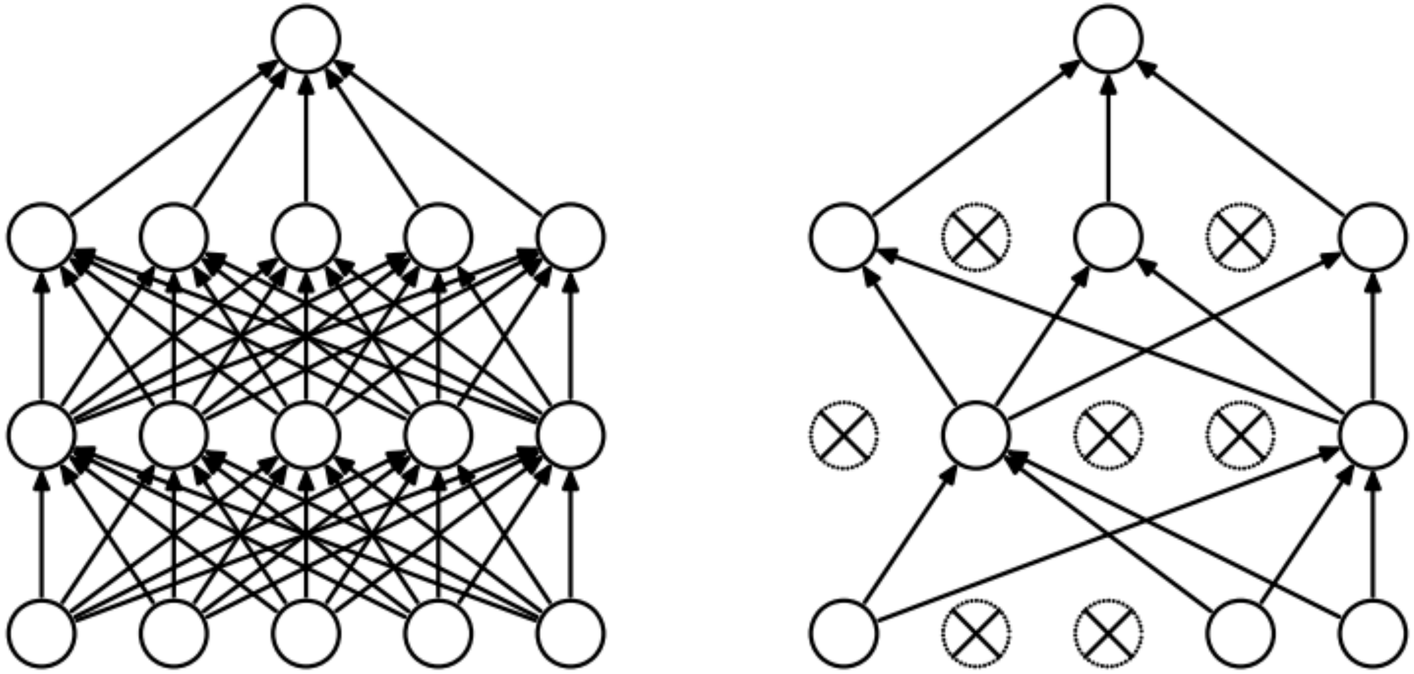
N: 데이터 갯수, $E(\cdot, \cdot)$: softmax loss or square loss

$\mathbf{y}_i$: target, $\hat{\mathbf{y}}_\mathbf{i}$: network output, λ: weight decay, L: # of layers

$\mathbf{W}$: weight, $\mathbf{b}$:bias

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

MCDropout

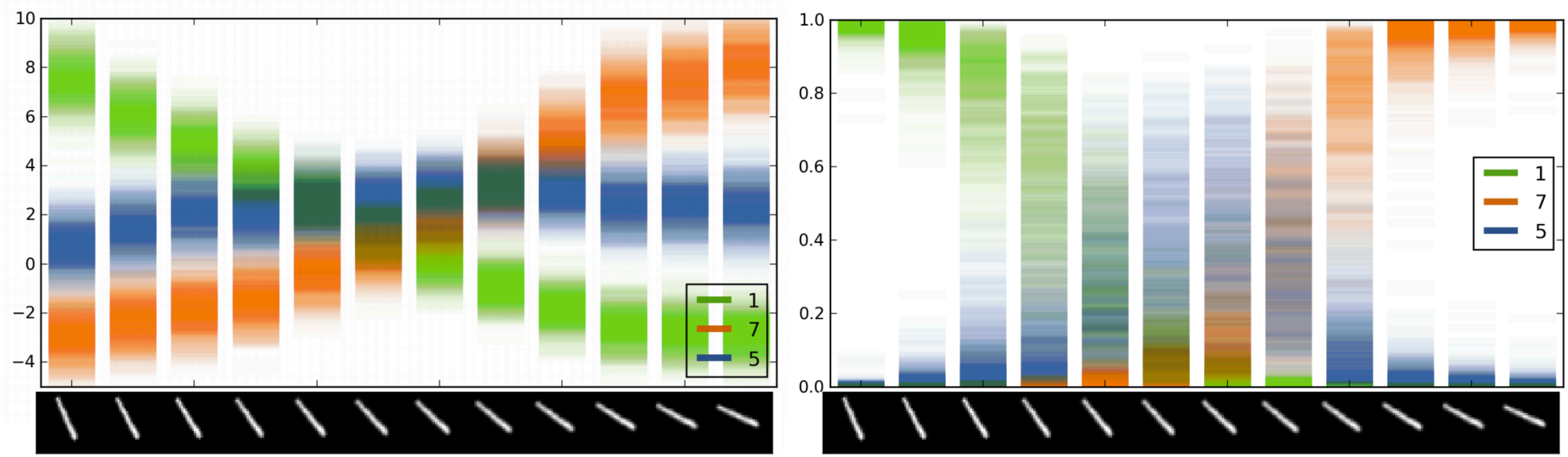


(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

MCDropout



(a) Softmax *input* scatter

(b) Softmax *output* scatter

Figure 4. **A scatter of 100 forward passes of the softmax input and output for dropout LeNet.** On the X axis is a rotated image of the digit 1. The input is classified as digit 5 for images 6-7, even though model uncertainty is extremly large (best viewed in colour).

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

Uncertainty

What uncertainties do we need in bayesian deep learning for computer vision?

Uncertainty

- Uncertainty는 크게 epistemic uncertainty와 aleatoric uncertainty로 구분합니다.

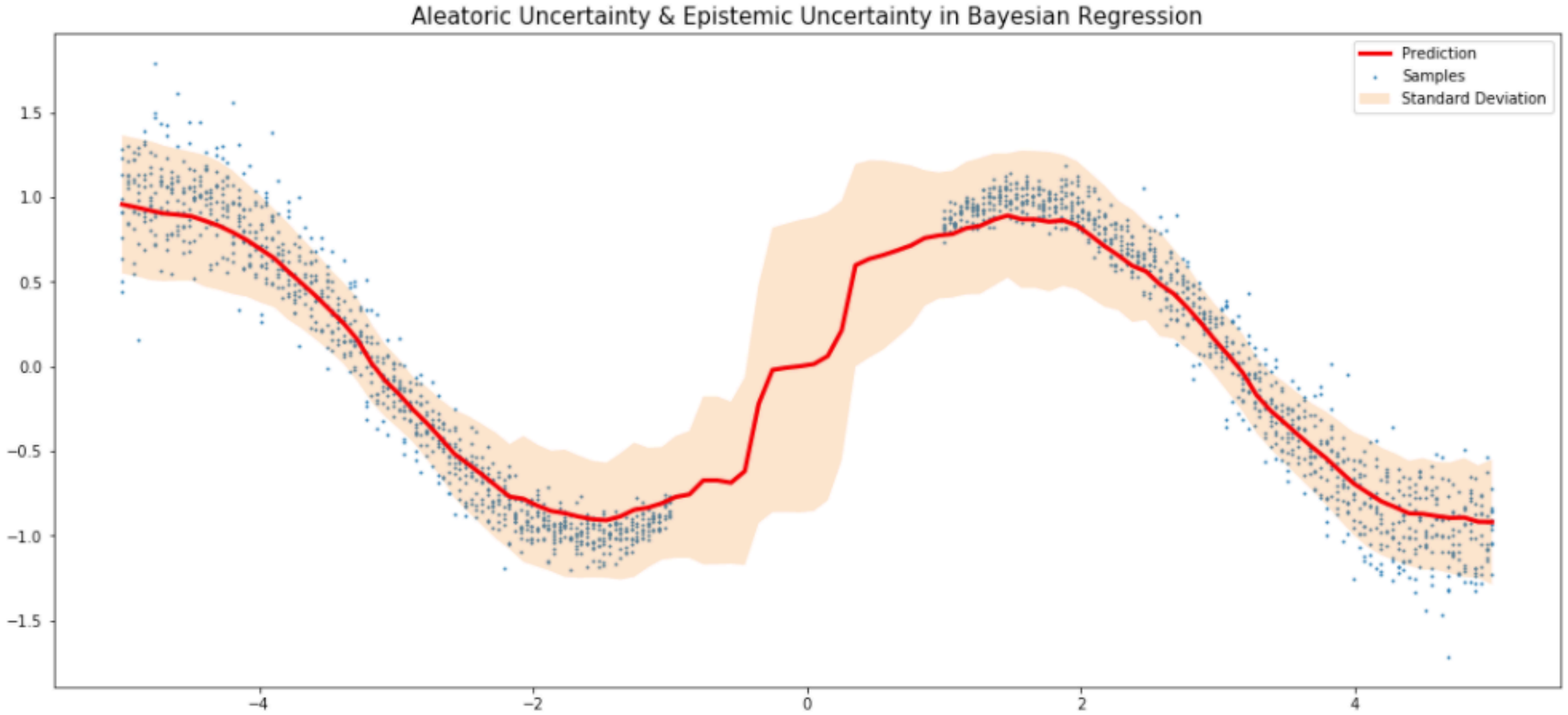
- Epistemic uncertainty는 model uncertainty라고도 불리며, 모델의 파라메터의 불확실성에 의해서 일어나는 불확실성을 뜻합니다. 충분히 많은 데이터를 통해 모델의 파라메터를 트레이닝하여 극복할 수 있는 요소입니다.

- Aleatoric uncertainty는 data uncertainty라고도 불리며, 데이터가 가지는 불확실성을 의미합니다. 센서 데이터와 같이 데이터에 노이즈가 있는 경우 많은 많은 데이터가 공급되어도 불확실성을 줄일 수 없습니다.

- Aleatoric uncertainty는 homoscedastic, heteroscedastic 두종류로 분류할 수 있습니다. homoscedastic uncertainty는 서로 다른 입력 데이터에 대한 각각의 출력 데이터의 분산이 같은 경우이고, heteroscedastic uncertainty은 서로 다른 입력 데이터에 대한 각각의 출력 데이터의 분산이 다를 수 있는 경우입니다.

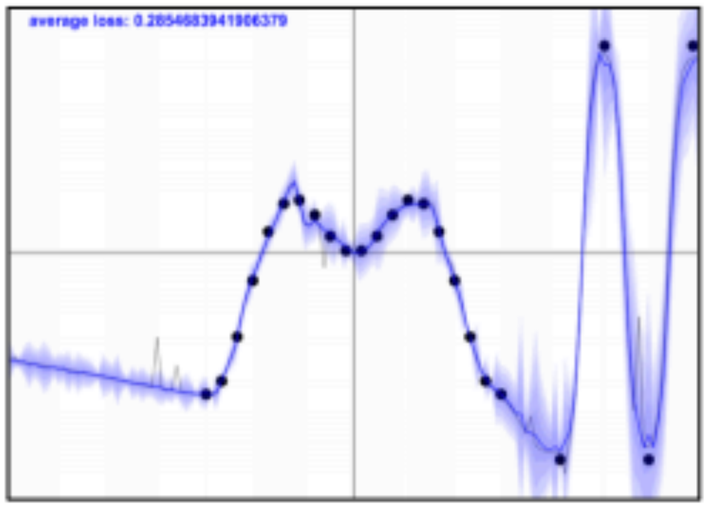
- GPS 장치 같이 지도의 어떤 지점을 인식하든 오차 범위가 일정하게 정해져 있는 경우 homoscedastic uncertainty를 갖는다 할 수 있습니다. 어떤 물체인식 모델이 특정 물체에 대해 정확하게 인식하고 다른 물체에 대해 부정확하게 인식한다면 heteroscedastic uncertainty를 갖는다 할 수 있습니다.

Uncertainty

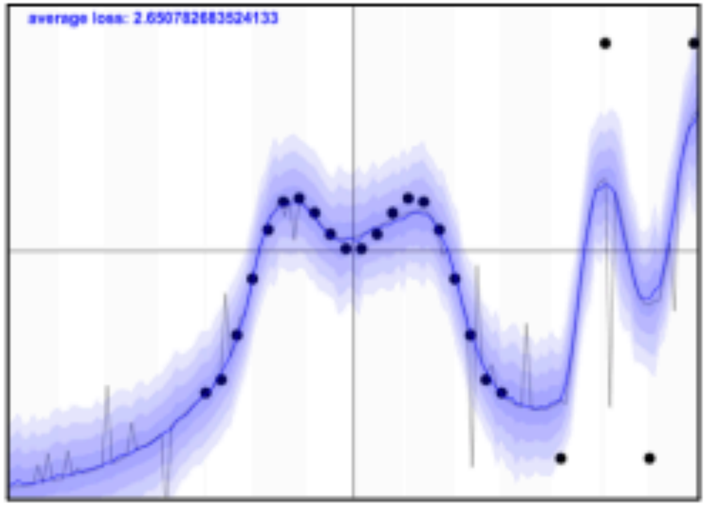


src/uncertainty.ipynb

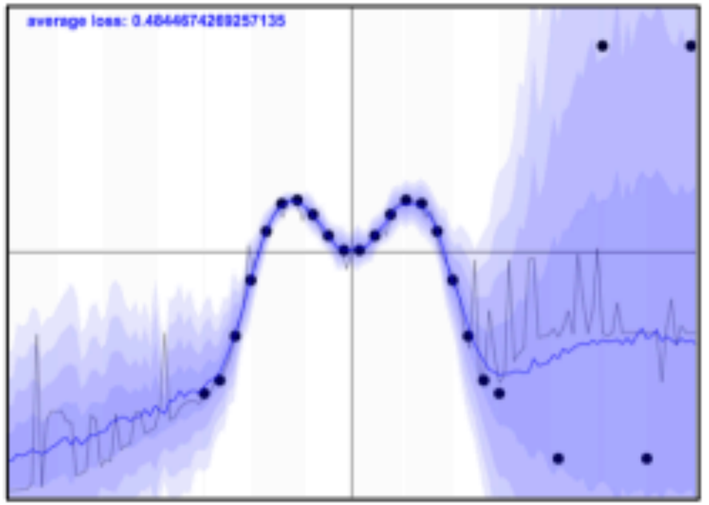
Uncertainty



(a) Homoscedastic model with small observation noise.



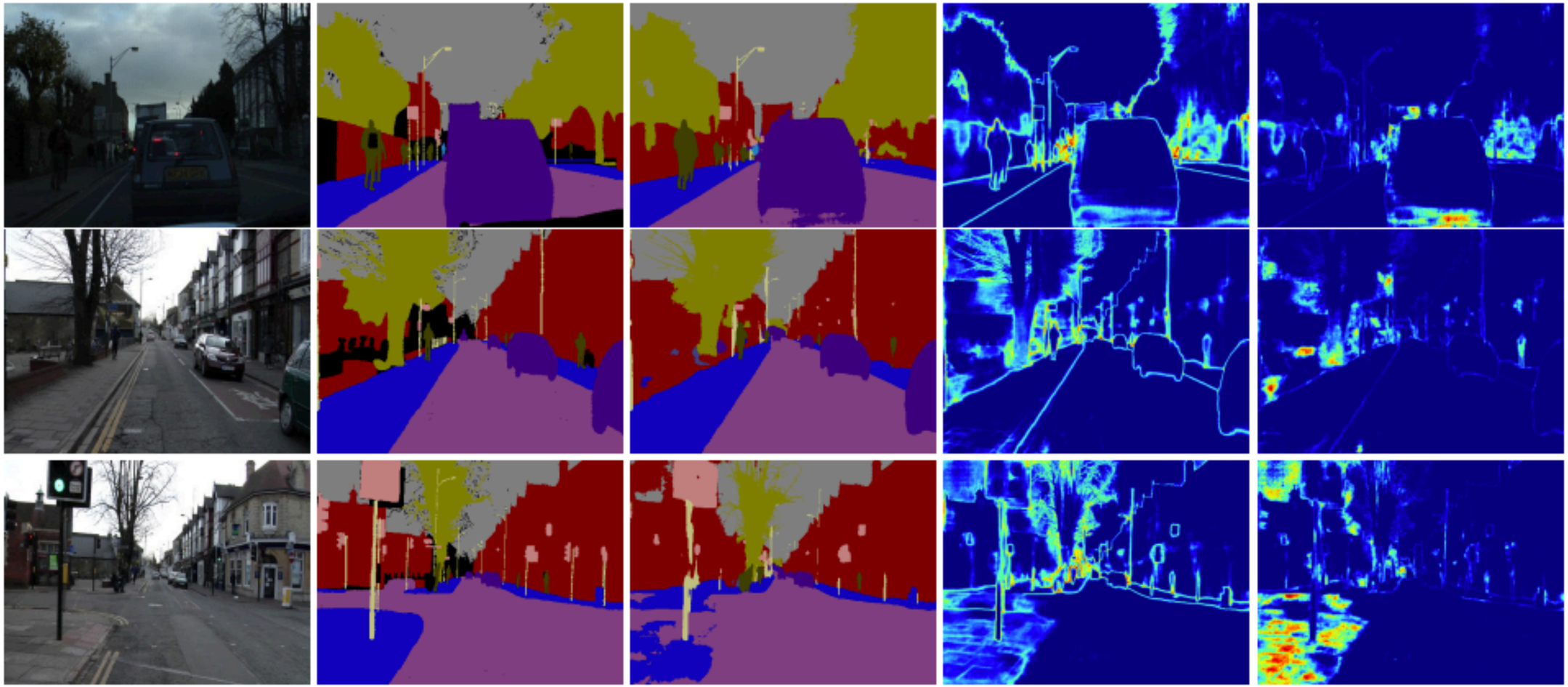
(b) Homoscedastic model with large observation noise.



(c) Heteroscedastic model with data-dependent observation noise.

Gal, Yarin. "Uncertainty in deep learning." University of Cambridge 1.3 (2016). p.85

Uncertainty



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

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Aleatoric Uncertainty In Bayesian Regression

- Bayesian Neural Networks에서 regression의 경우 추가적으로 variance를 나타내는 값 σ^2를 출력값으로 설정하여 data에 대한 uncentainty를 나타내는데 사용할 수 있습니다.

- 이 때, σ^2를 모든 데이터에 대해 동일하도록 설정하면 homoschedastic model이 되고, 각각의 입력 데이터에 대한 함수 $\sigma^2(\mathbf{x}_i)$로 만들면 heteroschedastic model이 됩니다.

- Predictive distribution을 Gaussian으로 설정했을 때, loss function을 다음과 같이 만들 수 있습니다.

$$\mathcal{L}_{\mathrm{NN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\sigma(\mathbf{x}_i)^2}\left\|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)\right\|^2 + \frac{1}{2}\log\sigma(\mathbf{x}_i)^2$$

N: # of the data points

$\mathbf{y}_i$: i-th target

$\mathbf{f}(\mathbf{x}_i)$: mean of output

$\sigma^2(\mathbf{x}_i)$: variance of output

Epistemic Uncertainty In Bayesian Regression

- Bayesian neural networks를 이용한 regression의 경우 epistemic uncentainty는 출력값의 분산을 구하여 측정할 수 있습니다.

$$\mathrm{Var}(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2$$

σ_t^2: t번째 시행시의 variance 결과값

$\hat{\mathbf{y}}_t$: t번째 시행시의 mean 결과값

분산 = 제곱의 평균 - 평균의 제곱

$$\begin{aligned}\mathrm{Var}(X) &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2 - 2X\,\mathrm{E}[X] + \mathrm{E}[X]^2\right] \\ &= \mathrm{E}\left[X^2\right] - 2\,\mathrm{E}[X]\,\mathrm{E}[X] + \mathrm{E}[X]^2 \\ &= \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2\end{aligned}$$

Aleatoric Uncertainty In Bayesian Classification

- Regression에서 aleatoric uncertainty를 구한 방법처럼 classification에서도 variance를 나타내는 출력값 $\sigma^2(\mathbf{x}_i)$을 두어 aleatoric uncertainty를 구할 수 있습니다.

- Regression의 경우 결과값의 범위가 실수값 $(-\infty, \infty)$으로 주어졌으나, classfication의 경우 출력값이 0보다 크고 각 클래스들의 확률의 합이 1이 되어야 합니다.

- 이와 같은 조건을 만족 시키기 위해 softmax layer를 지나기 전, logit space상에서 variance를 둡니다.

$$\hat{\mathbf{x}}_i \mid \mathbf{W} \sim \mathcal{N}\left(\mathbf{f}_i^{\mathbf{W}}, \left(\sigma_i^{\mathbf{W}}\right)^2\right)$$

$$\hat{\mathbf{p}}_i = \mathrm{Softmax}(\hat{\mathbf{x}}_i)$$

Aleatoric Uncertainty In Bayesian Classification

- Classification의 경우 aleatoric uncertainty를 측정하기 위한 loss function은 다음과 같이 나타낼 수 있습니다.

$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \sigma_i^{\mathbf{W}} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp\left(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'} \right)$$

$Softmax(\hat{\mathbf{x}}_{i,t})$

i: data point의 index

T: ϵ_t의 샘플링 횟수

$\hat{x}_{i,t,c}$: legit vector $\mathbf{x}_{i,t}$의 c번째 element

Epistemic Uncertainty In Bayesian Classification

- Bayesian Neural Networks에서 classification의 경우 predictive distribution은 다음과 같이 모델의 출력값을 평균내어 근사할 수 있습니다.

$$p(y = c \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}\left(\mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})\right)$$

$\mathbf{x}$: test시의 입력, y: test시의 출력, c: 출력 클래스

$\mathbf{X}$: 트레이닝시의 입력 데이터셋, $\mathbf{Y}$: 트레이닝시의 출력 데이터셋

T: test 횟수, $\hat{W}_t$: t회 test때 sampling된 weight, $\mathbf{f}^{\hat{W}_t}$: t회 test때의 neural network

- 이렇게 구한 categorical distribution의 entropy를 측정하여 uncertainty를 나타내는 대표값으로 둘 수 있습니다.

$$H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$$

$\mathbf{p}$: 출력 클래스들에 대한 확률값을 가진 벡터

GAN

Generative Adversarial Nets

- GAN (Generative Adversarial Nets)은 generator와 discriminator로 구분되어 있습니다.

- Generator는 discriminator가 구별을 하지 못하도록 트레이닝 됩니다. 따라서 생성된 결과가 노이즈에서 점차 트레이닝 데이터와 가깝게 변해갑니다.

- Discriminator는 트레이닝 데이터와 generator가 만든 결과를 구분하도록 트레이닝 됩니다. Generator가 만든 결과를 입력으로 받으면 출력을 0으로, 트레이닝 데이터를 입력으로 받으면 출력을 1로 내보내도록 학습합니다. 그러나 generator가 생성하는 이미지가 점차 트레이닝 데이터와 가까워짐에 따라 discriminator가 제대로 구별하지 못하게 됩니다.

- GAN은 VAE와 더불어 generative model로써 대표적인 모델입니다. VAE가 blurry한 결과를 내보내는데 반해 GAN는 매우 선명한 결과를 내보내는 장점을 가지고 있습니다. 하지만 GAN은 model collapse 현상이나 트레이닝이 매우 어렵다는 점 등 단점을 가지고 있습니다.

Generative Adversarial Nets

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- GAN, Value function $V(G, D)$ 정의

3 Adversarial nets

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution p_g over data $\boldsymbol{x}$, we define a prior on input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z}; \theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(\boldsymbol{x}; \theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Discriminator의 트레이닝 방법, Generator의 loss saturation 문제

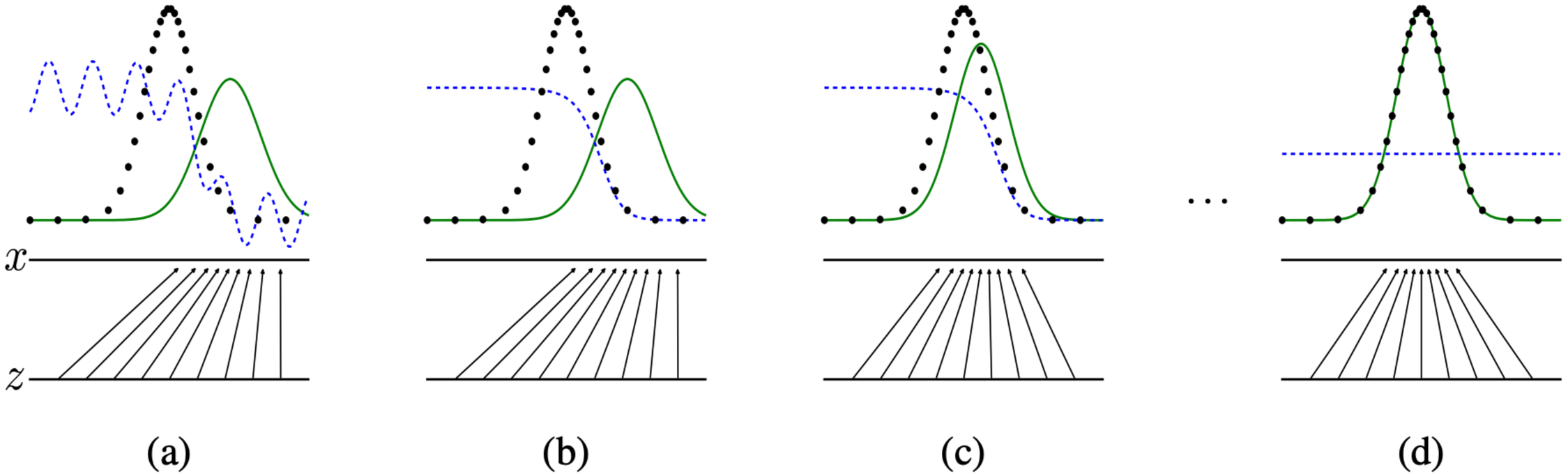
In the next section, we present a theoretical analysis of adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity, i.e., in the non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach. In practice, we must implement the game using an iterative, numerical approach. Optimizing D to completion in the inner loop of training is computationally prohibitive, and on finite datasets would result in overfitting. Instead, we alternate between k steps of optimizing D and one step of optimizing G. This results in D being maintained near its optimal solution, so long as G changes slowly enough. The procedure is formally presented in Algorithm 1.

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- 트레이닝 과정 도식화



I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- 트레이닝 Pseudo Code

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- GAN의 optimal solution이 $p_g = p_{data}$임을 증명, discriminator의 optimal solution 구함

4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator D for any given generator G.

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a\log(y) + b\log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Optimal discriminator에 대한 generator의 value function

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\boldsymbol{x})$, where Y indicates whether $\boldsymbol{x}$ comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \qquad (4) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

- Generator의 criterion은 JSD (Jensen-Shannon divergence)와 같음을 보임

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data distribution. □

Jensen Shannon Divergence

$$\text{JSD}(P\|Q) = \frac{1}{2}D(P\|M) + \frac{1}{2}D(Q\|M)$$

where $M = \frac{1}{2}(P + Q)$

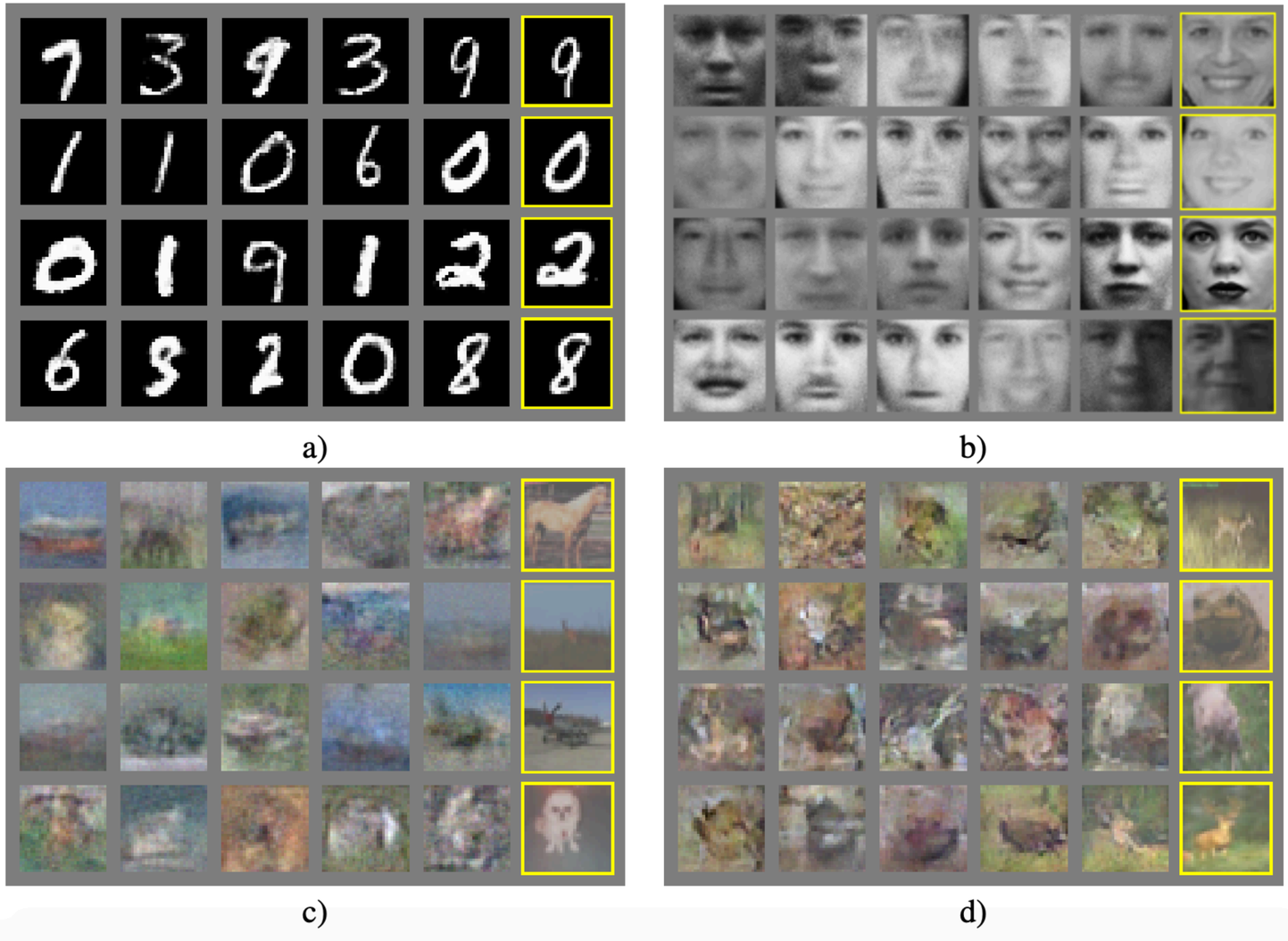
I. Goodfellow et al., Generative adversarial nets

Generative Adversarial Nets

$$\min_G \max_D V(D,G) = \min_G V(D^*,G) \qquad \text{for fixed optimal } \mathcal{D}$$

$$\begin{aligned}
V(D^*,G) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g}[\log(1-D^*(\mathbf{x}))] \\
&= \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -2\log 2 + 2\log 2 + \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -\log 4 + \int_{\mathbf{x}} d\mathbf{x}\; p_{\text{data}}(\mathbf{x}) \log \frac{2\cdot p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x}\; p_g(\mathbf{x}) \log \frac{2\cdot p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_g(\mathbf{x})} \\
&= -\log 4 + KL\left(p_{\text{data}} \middle|\middle| \frac{p_{\text{data}}+p_g}{2}\right) + KL\left(p_g \middle|\middle| \frac{p_{\text{data}}+p_g}{2}\right) \\
&= -\log 4 + 2\cdot JSD(p_{\text{data}}||p_g)
\end{aligned}$$

Generative Adversarial Nets



I. Goodfellow et al., Generative adversarial nets

Deep Convolutional GAN

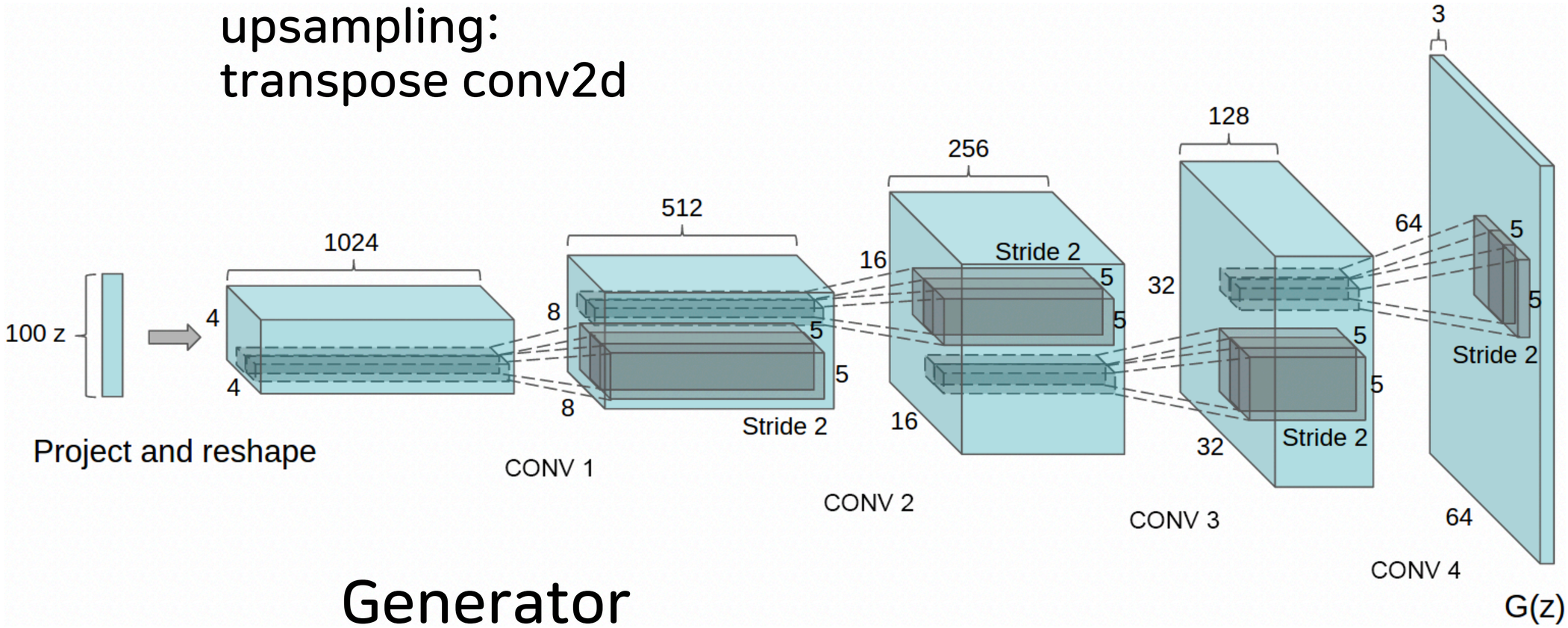
- Guidelines for Stable DCGAN

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

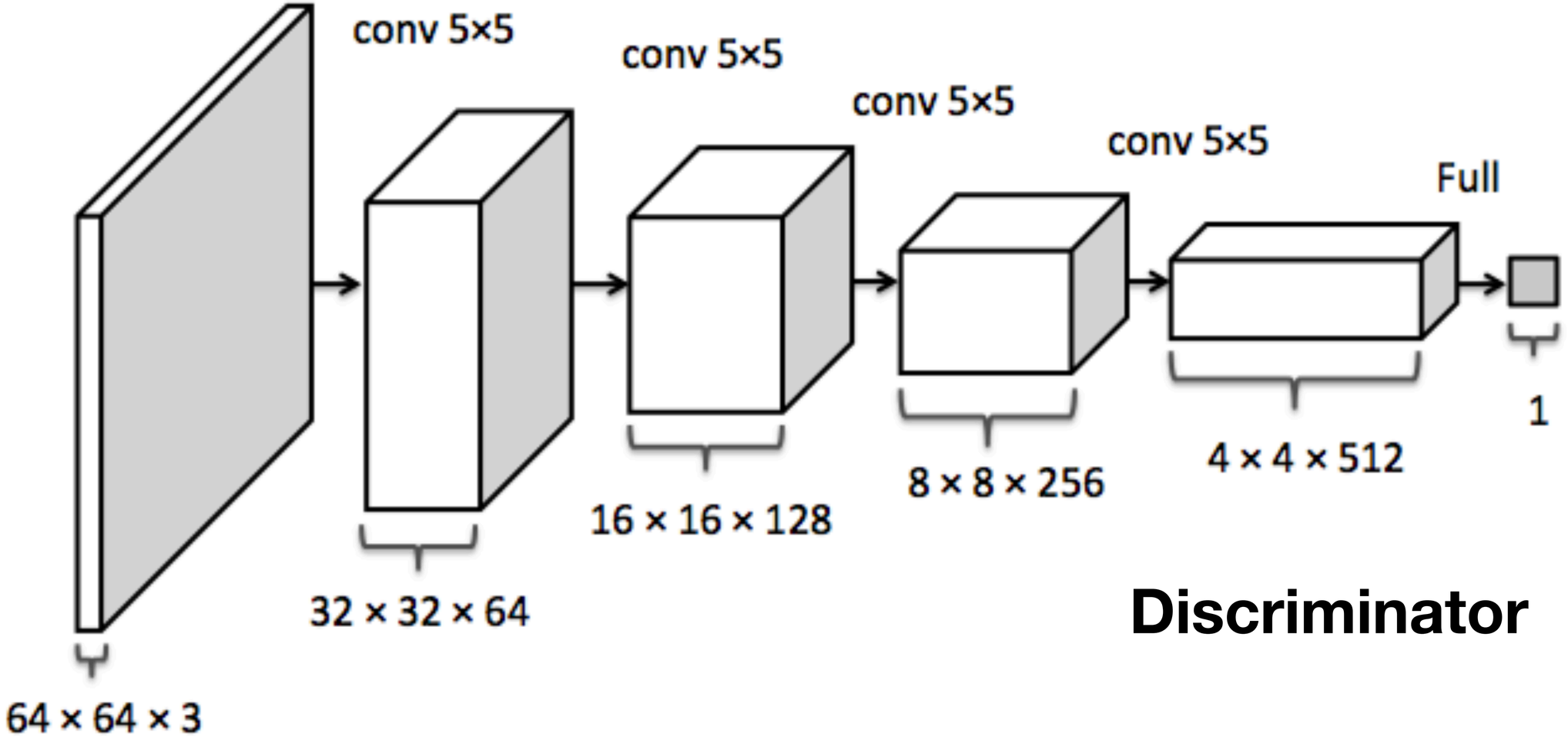
Deep Convolutional GAN



Generator

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Deep Convolutional GAN



R. Yeh, et al., Semantic Image Inpainting with Perceptual and Contextual Losses

Deep Convolutional GAN

- DCGAN

A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Generative Adversarial Nets

- WGAN

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

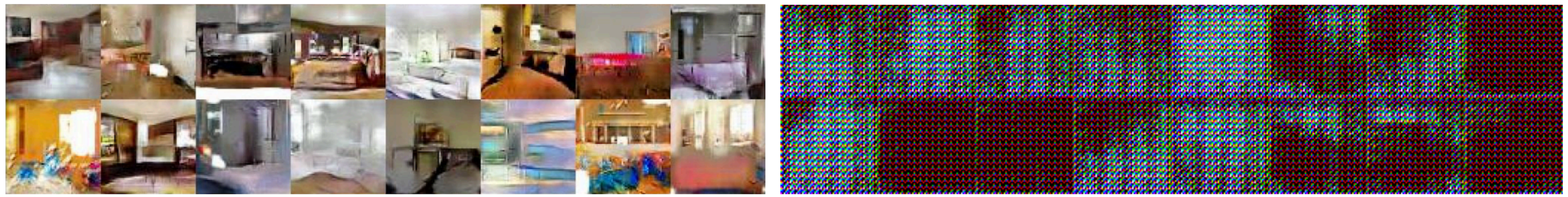


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]).

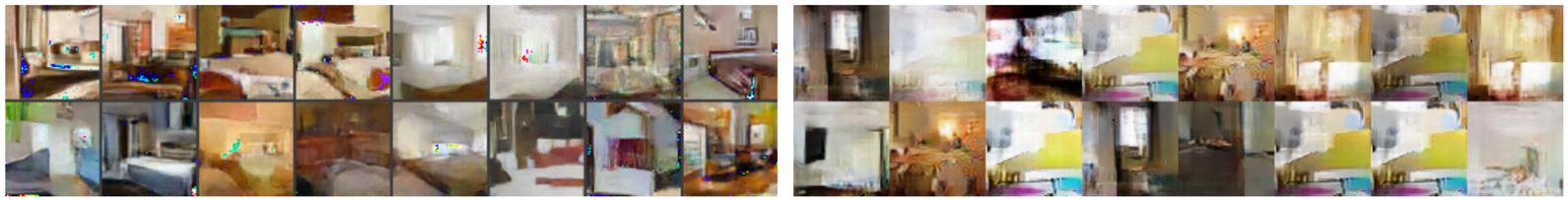


Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a

Martin Arjovsky, et al., Wasserstein GAN

Generative Adversarial Nets

- Progressive GAN

Tero Karras, et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

Generative Adversarial Nets

- Style GAN

Tero Karras, et al., A Style-Based Generator Architecture for Generative Adversarial Networks

더 읽을거리

- Meire Fortunato, Charles Blundell, Oriol Vinyals. Bayesian Recurrent Neural Networks
- Kumar Shridhar, Felix Laumann, Marcus Liwicki. A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference
- Yarin Gal. Uncertainty in Deep Learning
- Yarin Gal, Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning
- Yarin Gal, Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks
- Alex Kendall, Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?
- Yarin Gal, Zoubin Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference