Marrying Graphical Models & Deep Learning

Max Welling

University of Amsterdam

SCYFER
the science of prediction

QUVA
Uva-Qualcomm
Quva Lab

AMLAB
Amsterdam
Machine Learning Lab

UvA - BOSCH
DELTA LAB

Universiteit van Amsterdam

CIFAR
Canadian Institute for Advanced Research

Overview:

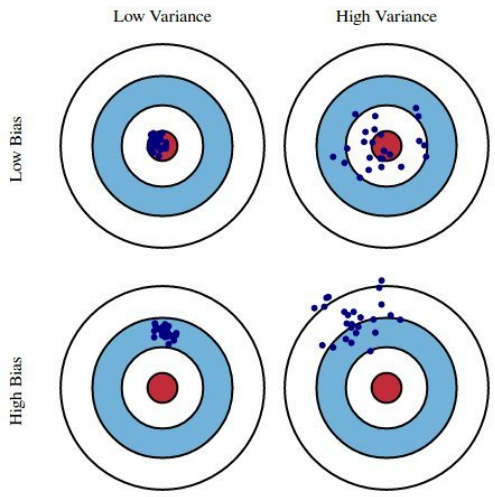
- Machine Learning as Computational Statistics
- Graphical Models:
  - Bayes nets
  - MRFs
  - Latent variable models
- Inference:
  - Variational inference
  - MCMC
- Learning:
  - EM
  - Amortized EM
  - Variational autoencoder
- Generative versus discriminative modeling
- Deep Learning:
  - CNN
  - Dropout
- Bayesian inference
  - Bayesian deep models
  - Compression

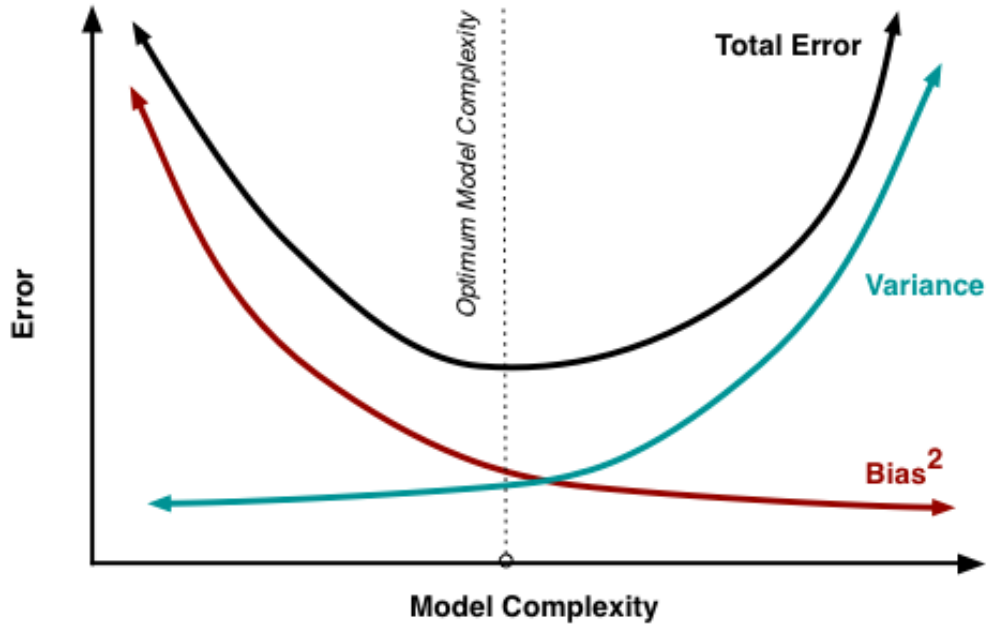
ML as Statistics

- Data: $\{X_1, ..X_n, Y_1, .., Y_n\} \sim P(X_1, ..X_n, Y_1, .., Y_n)$
- Optimize objective:
  - maximize log likelihood: $\max_{\Theta} \log P(X_1, ...X_n|\Theta)$ (unsupervised)

    $\max_{\Theta} \log P(Y_1, ..Y_n|X_1, ...X_n, \Theta)$ (supervised)
  - minimize loss: $\min_{\Theta} \sum_i Loss(Y_i, \hat{Y}(X_i, \Theta))$ (supervised)
- *ML is more than an optimization problem: it's a statistical inference problem*.
  - E.g.: you should not optimize parameters more precisely than the scale at which the MLE fluctuates under resampling the data: $\Theta_{mle}(X, Y) \approx \Theta'_{mle}(X', Y')$, or risk overfitting.

Bias Variance Tradeoff





$$Y = f(X) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon).$$

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

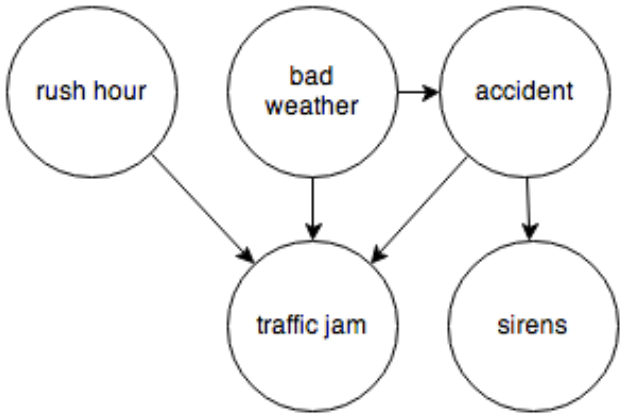
$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

http://scott.fortmann-roe.com/docs/BiasVariance.html

Graphical Models

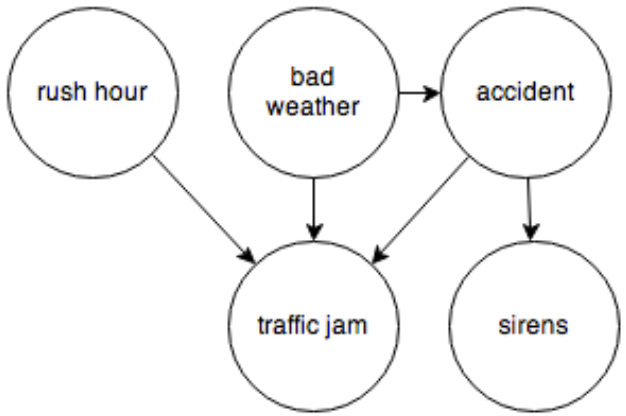
- A graphical representation to concisely represent (conditional) independence relations between variables.
- There is a one-to-one correspondence between the dependencies implied by the graph and the probabilistic model.

- E.g. Bayes Nets



P(all) = P(traffic-jam | rush-hour, bad-weather, accident) x
P(sirens | accident) x
P(accident | bad-weather) x
P(bad-weather) x P(rush-hour)

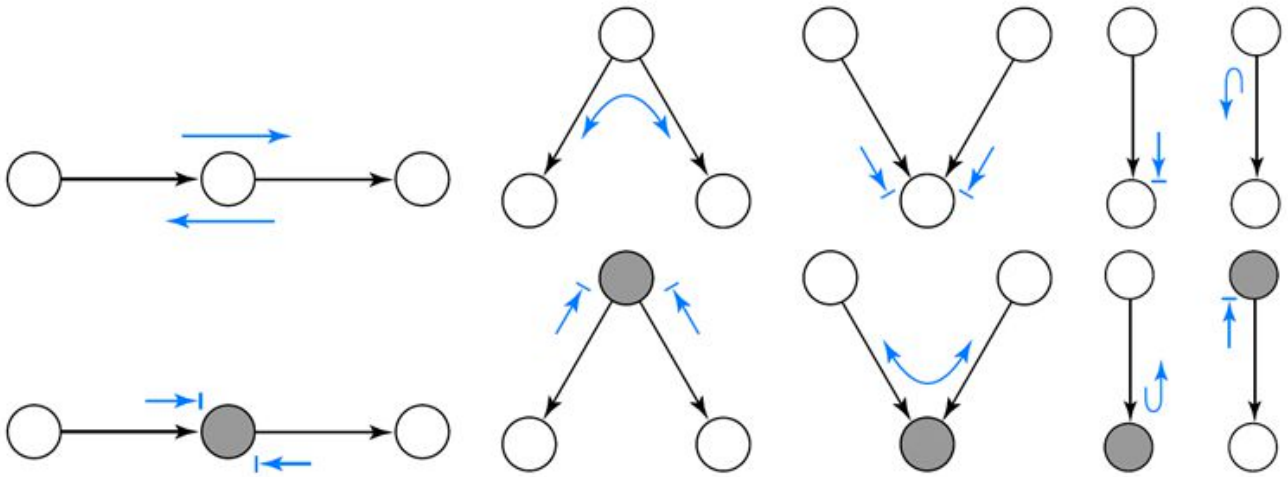
P(rush-hour) independent P(bad-weather) ←→ sum_{traffic-jam,sirens,accident) P(all) = P(rush-hour) P(bad-weather)



Rush-hour independent of bad-weather

Bayes ball algorithm

An undirected path is active if a Bayes ball travelling along it never encounters the "stop" symbol: $\rightarrow\!|$



If there are no active paths from X to Y when $\{Z_1, \ldots, Z_k\}$ are shaded, then $X \perp\!\!\!\perp Y \mid \{Z_1, \ldots, Z_k\}$.

Source:

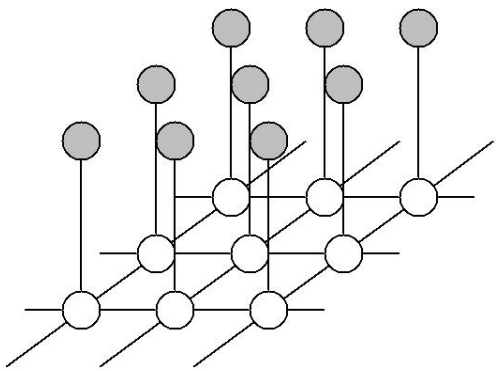
CS188: Computational Models of Human Behavior

INTRODUCTION TO GRAPHICAL MODELS

SLIDE CREDITS: KEVIN MURPHY, MARK PASHKIN, ZOUBIN GHAHRAMANI AND JEFF BILMES

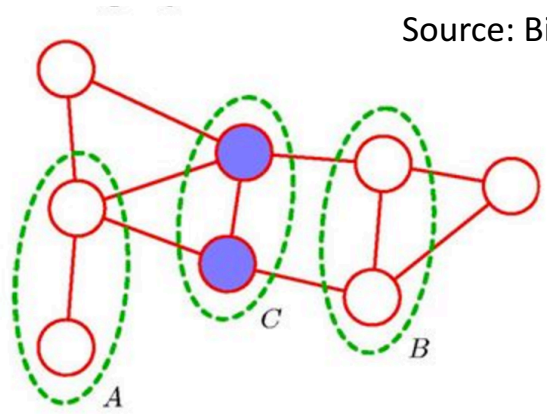
Markov Random Fields

Undirected edges



(Conditional) independence relationships easy:

Source: Bishop



A independent B given C
(for independence, all paths must be blocked)

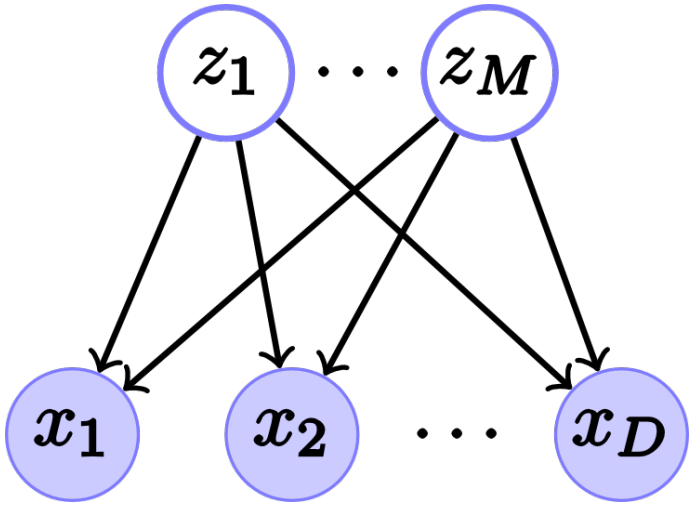
Probability distribution: $$P(X) = \frac{\prod_c \Psi_c(X_c)}{Z}$$

$\Psi_c(X_c)$: maximal clique = largest completely connected subgraphs

Hammersley-Clifford Theorem: if P>0 all x, then all (conditional) independencies in P match those of the graph.

Latent Variable Models

- Introducing latent (unobserved) variables will dramatically increase the capacity of a model.

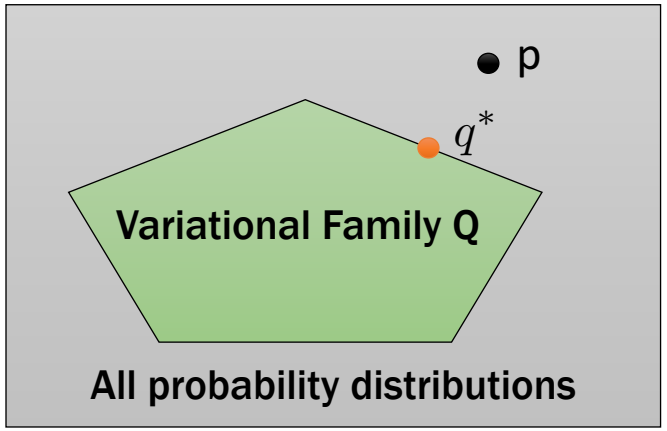


$$P(X) = \sum_Z P(X|Z)P(Z)$$

- Problem: P(Z|X) is intractable for most nontrivial models

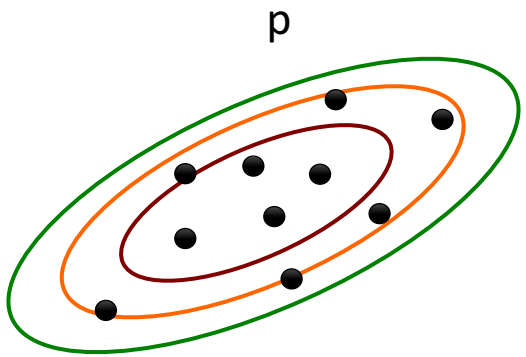
Approximate Inference

Variational Inference



- Deterministic
- Biased
- Local minima
- Easy to assess convergence

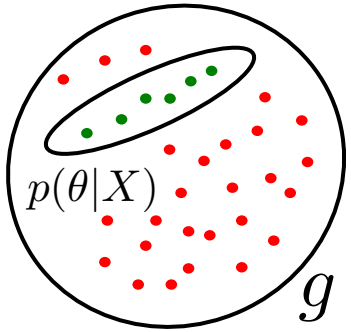
Sampling



- Stochastic (sample error)
- Unbiased
- Hard to mix between modes
- Hard to assess convergence

Independence Samplers & MCMC

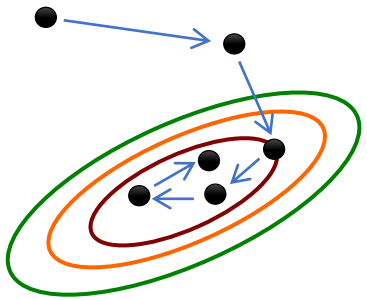
Generating Independent Samples



Sample from g and suppress samples with low p(θ|X)
e.g. a) Rejection Sampling b) Importance Sampling

- Does not scale to high dimensions

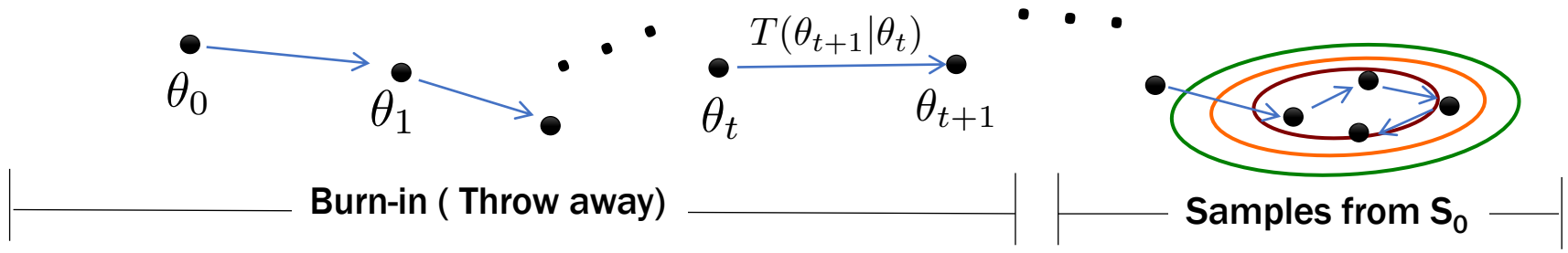
Markov Chain Monte Carlo



- Make steps by perturbing previous sample
- Probability of visiting a state is equal to P(θ|X)

Sampling 101 – What is MCMC?

Given target distribution S_0, design transitions s.t. $p_t(\theta_t) \to S_0$ as $t \to \infty$

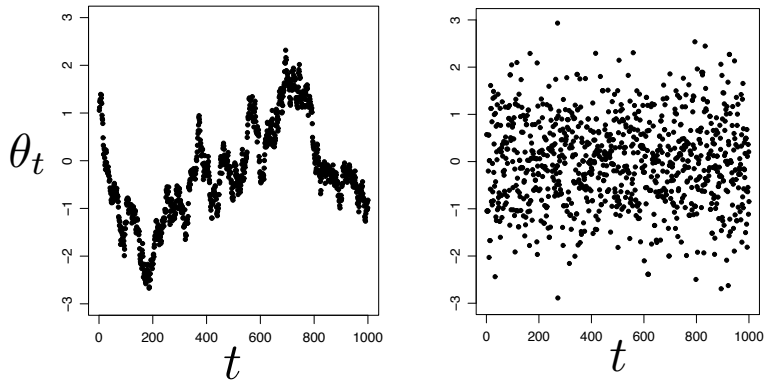


$$I = \langle f \rangle_{S_0} \approx \hat{I} = \frac{1}{T}\sum_{t=1}^{T} f(\theta_t)$$

$$\mathrm{Bias}(\hat{I}) = \mathbb{E}[\hat{I} - I] = 0$$

$$\mathrm{Var}(\hat{I}) = \tau \frac{\mathrm{Var}(f)}{T}$$

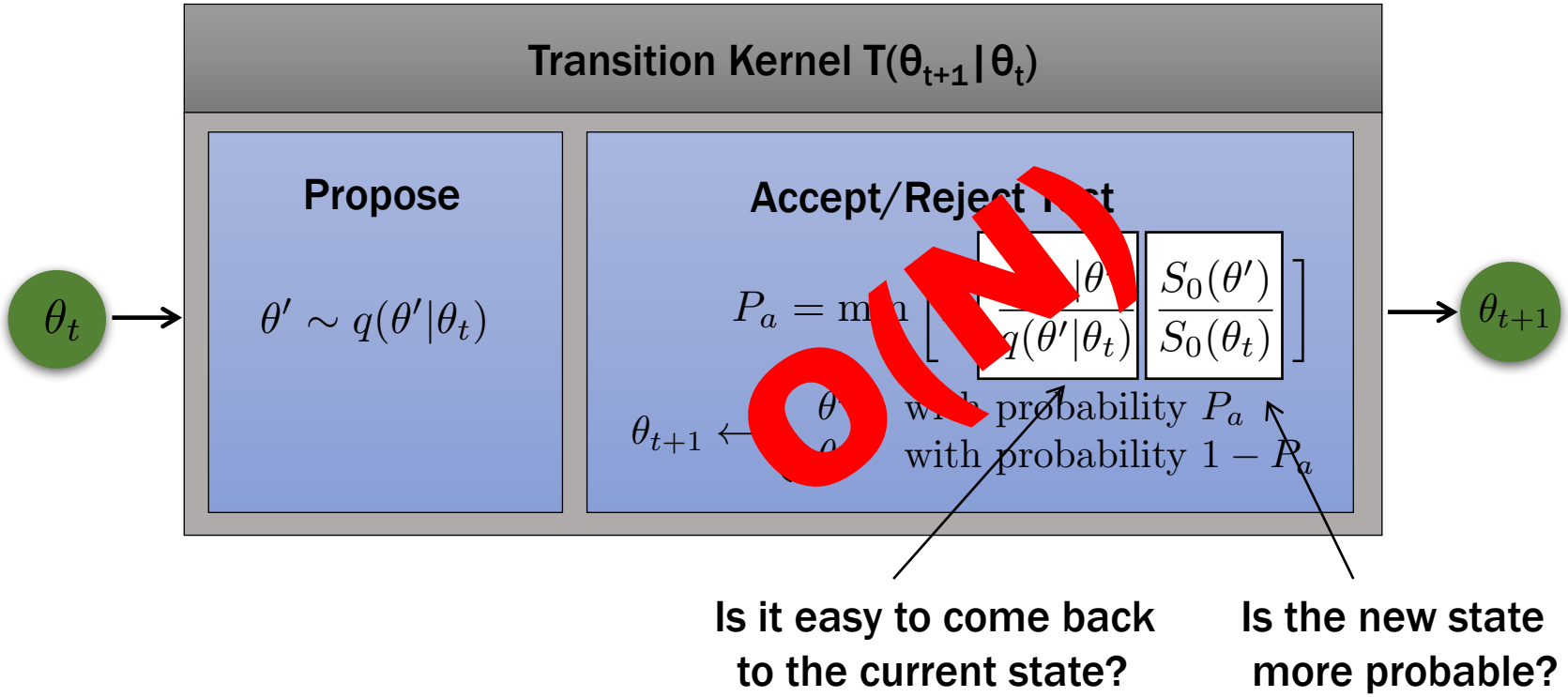
Auto correlation time



High τ

Low τ

Sampling 101 – Metropolis-Hastings

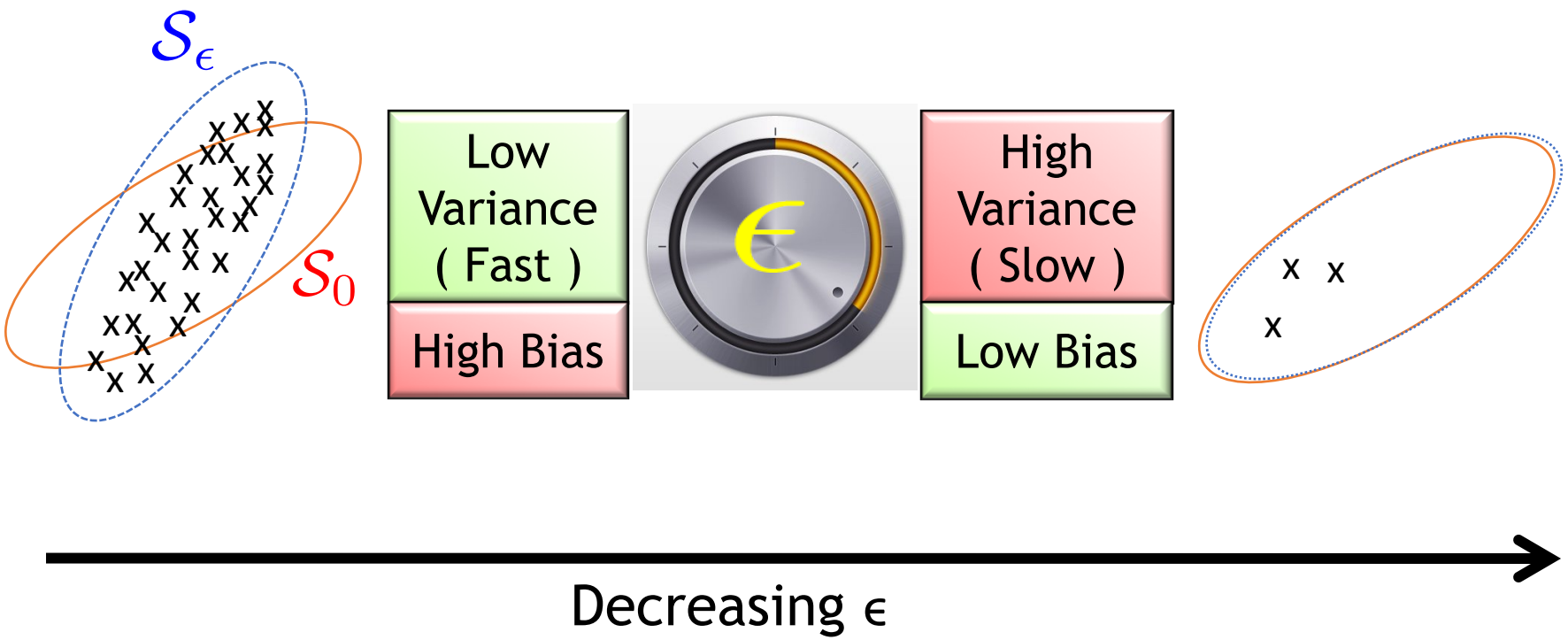


For Bayesian Posterior Inference, $S_0(\theta) \propto p(\theta) \prod_{i=1}^{N} p(x_i|\theta)$

1) Burn-in is unnecessarily slow.

2) $Var[\hat{I}] \propto \frac{1}{T}$ is too high.

Approximate MCMC

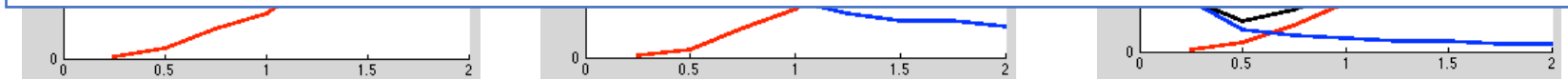


Minimizing Risk

2

Risk | Bias | Variance

Given finite sampling time, ϵ=0 is not the optimal setting.



X Axis – ϵ, Y Axis – Bias^2, Variance, Risk

Computational Time

Stochastic Gradient Langevin Dynamics

Welling & Teh 2011

Gradient Ascent

$$\Delta\theta_t = \frac{\epsilon}{2}\left(\nabla \log p(\theta_t) + \sum_{i=1}^{N} \nabla \log p(x_i;\theta_t)\right)$$

Langevin Dynamics

$$\Delta\theta_t = \frac{\epsilon}{2}\left(\nabla \log p(\theta_t) + \sum_{i=1}^{N} \nabla \log p(x_i;\theta_t)\right) + \mathbb{N}(0,\epsilon)$$

$\downarrow$

Metropolis-Hastings Accept Step

Stochastic Gradient Ascent

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{t_i};\theta_t)\right)$$

$$\sum_{t=1}^{\infty} \epsilon_t = \infty \qquad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty$$

e.g. $\epsilon_t = \frac{a}{(b+t)^{\gamma}}$

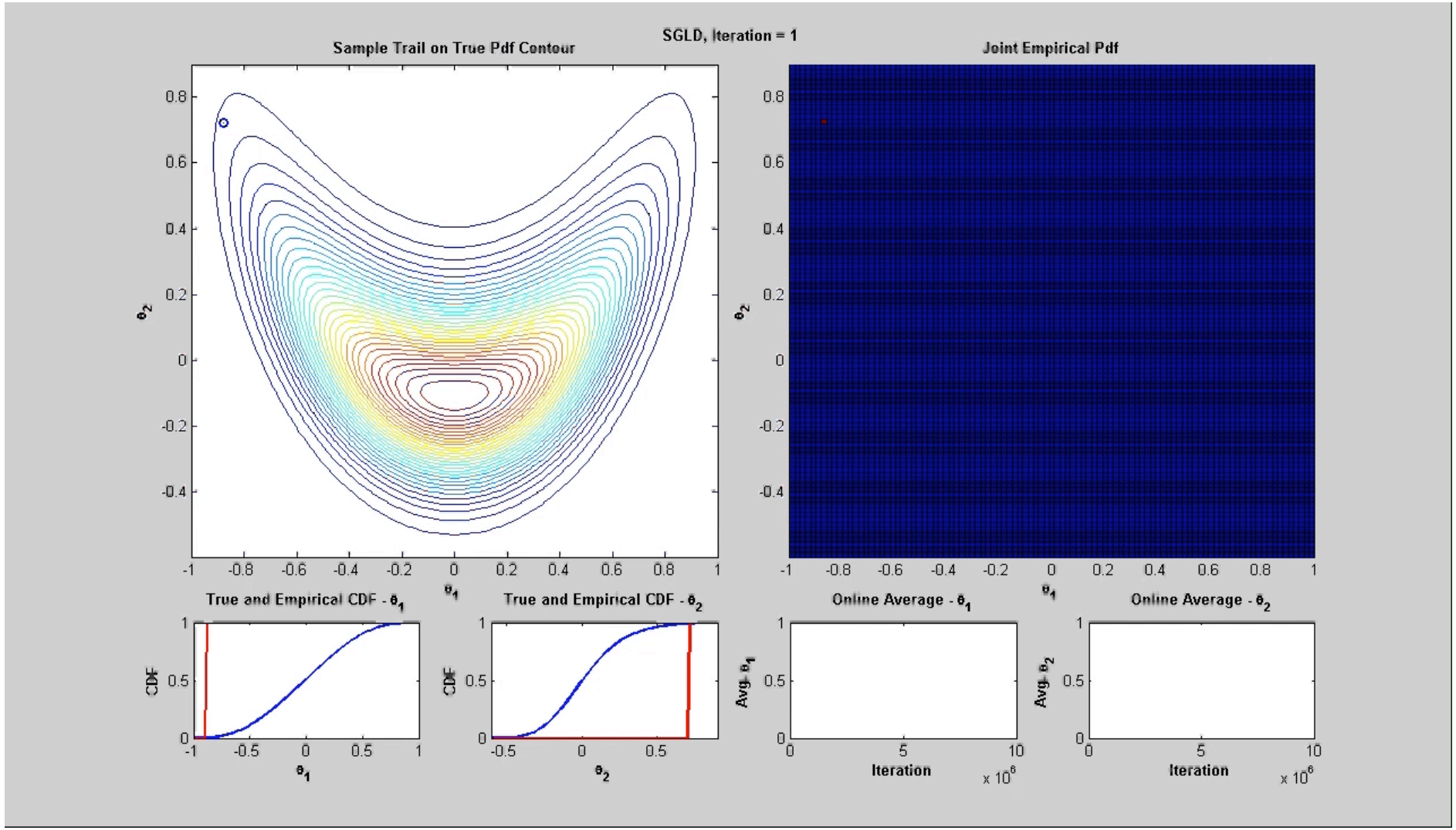
Stochastic Gradient Langevin Dynamics

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{t_i};\theta_t)\right) + \mathbb{N}(0,\epsilon_t)$$

$$\sum_{t=1}^{\infty} \epsilon_t = \infty \qquad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty$$

~~Metropolis-Hastings Accept Step~~

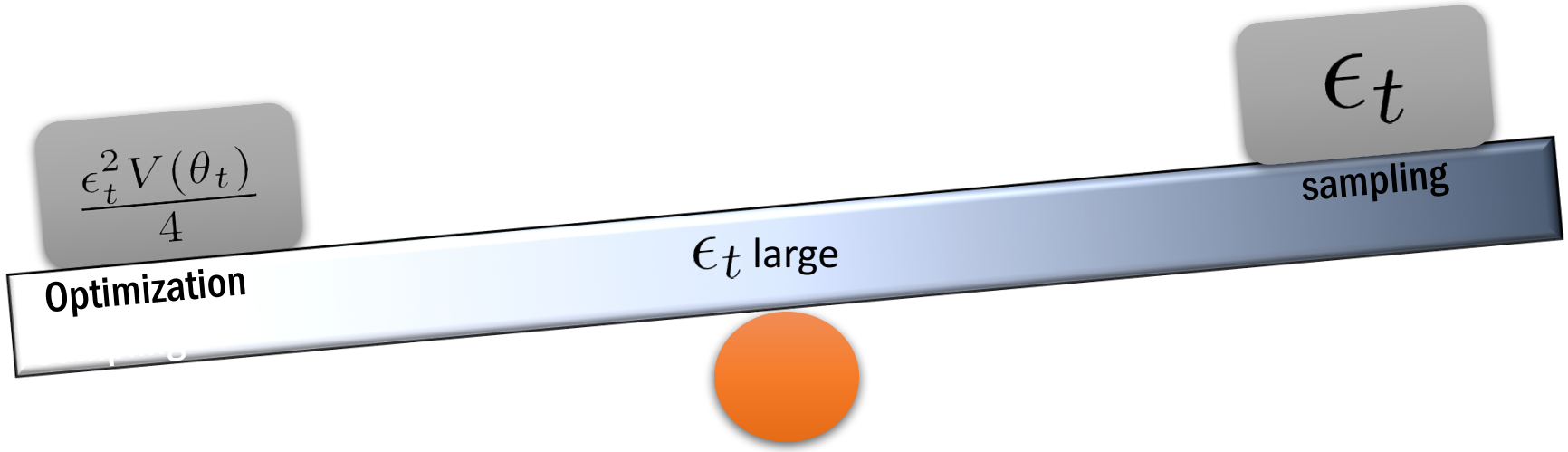
Demo: Stochastic Gradient LD



A Closer Look …

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{t_i};\theta_t)\right) + \mathbb{N}(0,\epsilon_t)$$

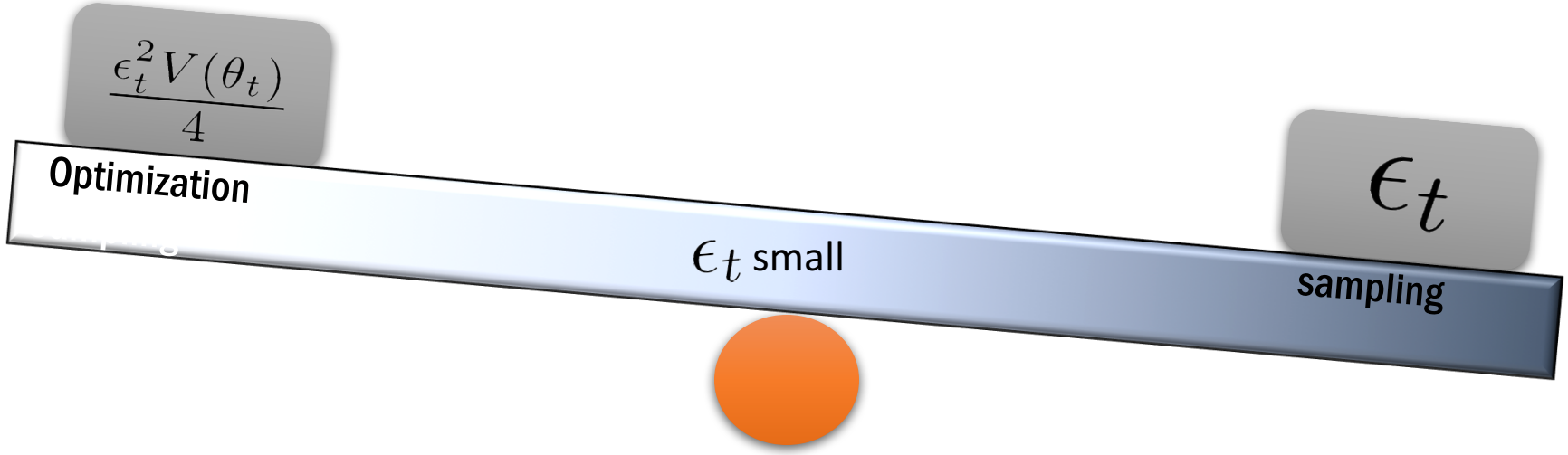
$$\mathcal{N}\left(\sum_{i=1}^{N} \nabla \log p(x_i;\theta_t), V(\theta_t, n)\right)$$



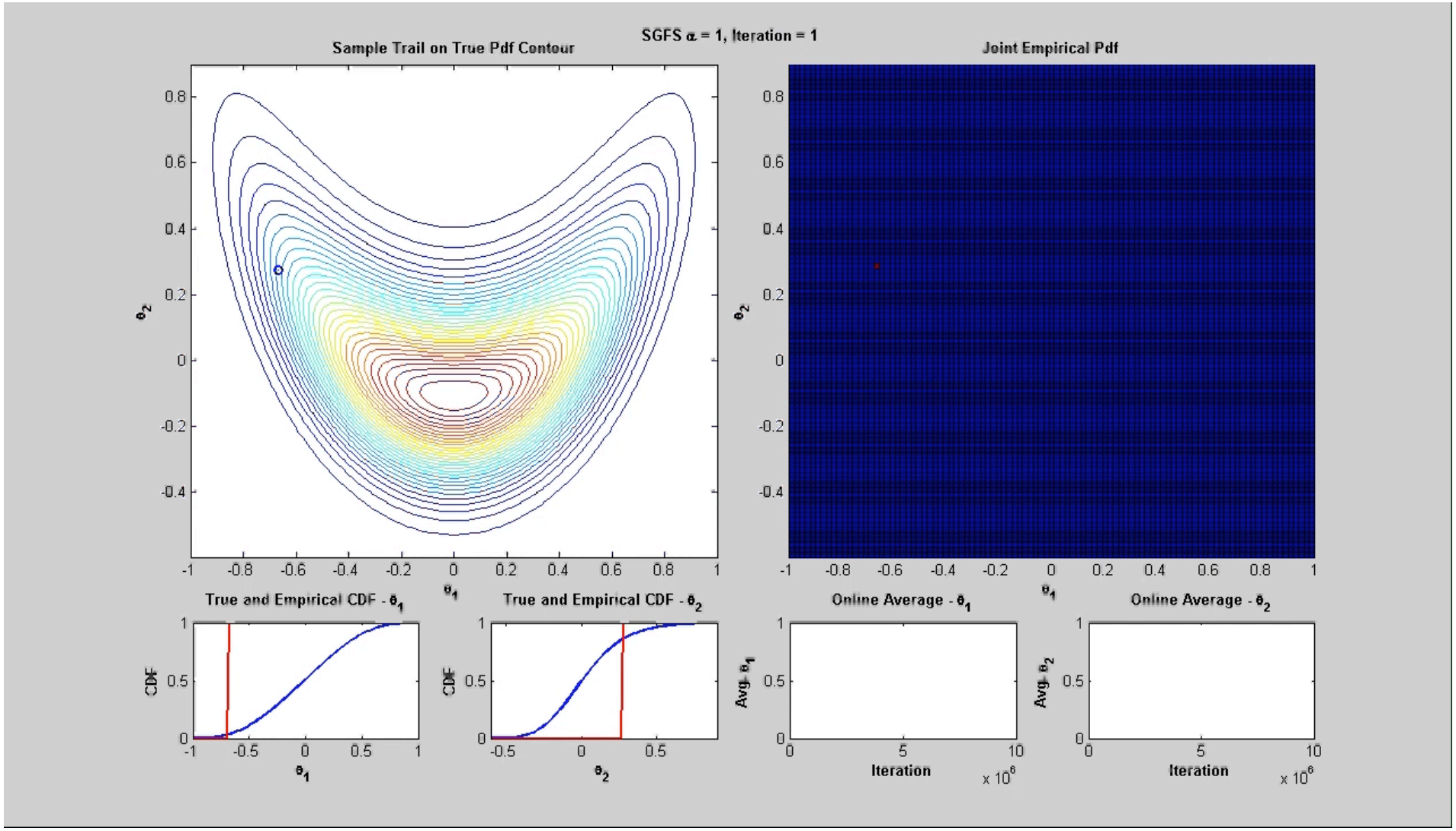
A Closer Look …

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{t_i};\theta_t)\right) + \mathbb{N}(0,\epsilon_t)$$

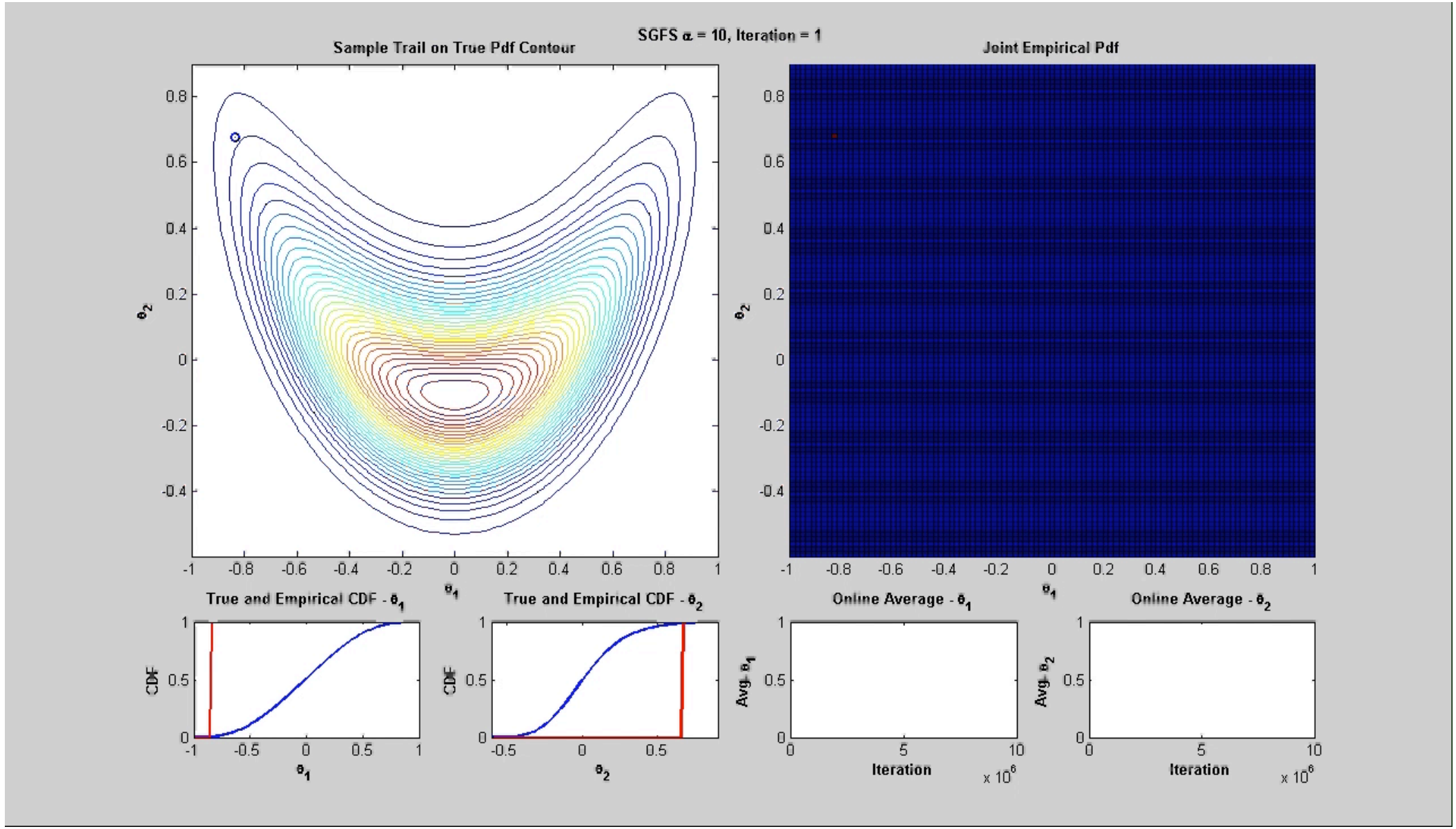
$$\mathcal{N}\left(\sum_{i=1}^{N} \nabla \log p(x_i;\theta_t), V(\theta_t, n)\right)$$



Demo SGLD: large stepsize

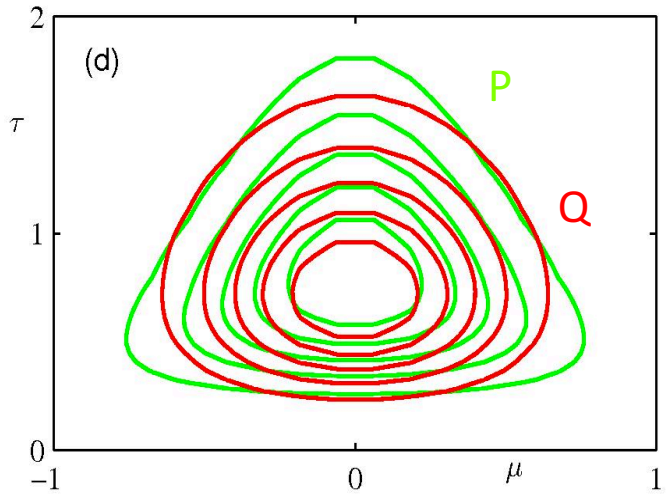


Demo SGLD: small stepsize



Variational Inference

- Choose tractable family of distributions (e.g. Gaussian, discrete)
- Minimize over Q: $KL[Q(Z|X)||P(Z||X)]$
- Equivalent to maximize over Φ : $\sum_Z Q(Z|X,\Phi)\left(\log P(X|Z,\Theta)P(Z) - \log Q(Z|X,\Phi)\right)$



Learning: Expectation Maximization

$$KL[Q(Z|X)||P(Z||X)]$$

Gap:

Bound $B(\Theta, \Phi)$

$$\log P(X|\Theta) =$$

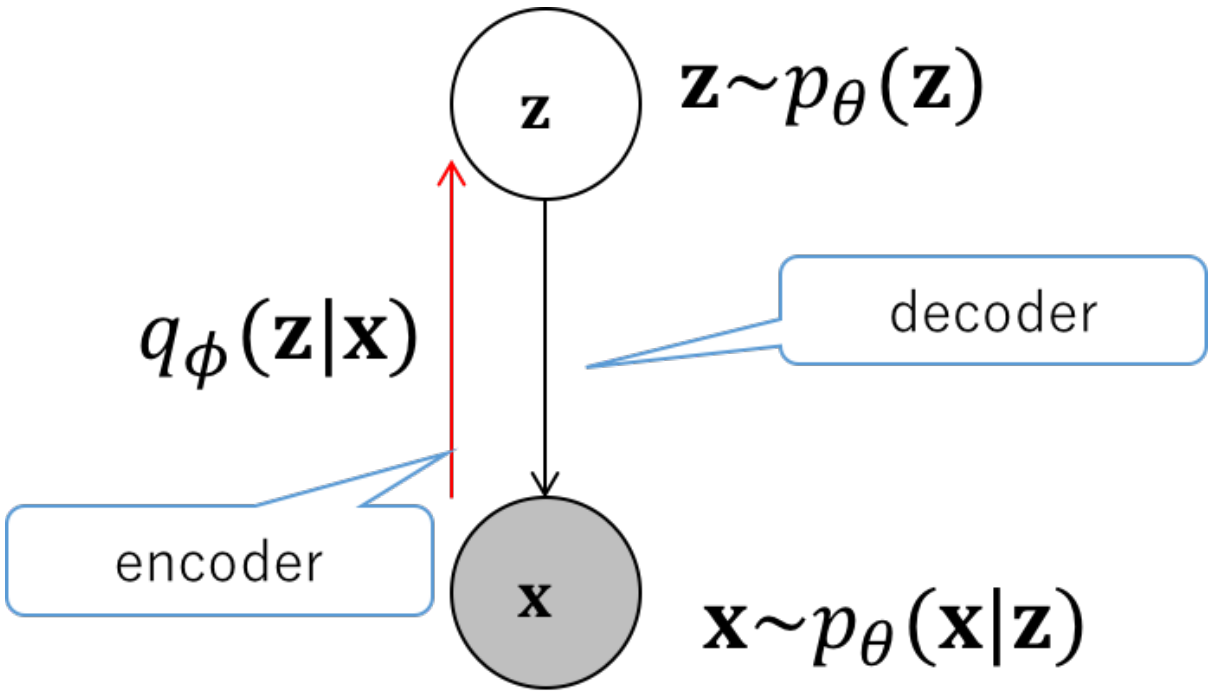
$$\log \sum_Z P(X|Z,\Theta)P(Z) \geq$$

$$\sum_Z Q(Z|X,\Phi)\left(\log P(X|Z,\Theta)P(Z) - \log Q(Z|X,\Phi)\right)$$

E-step: $\arg\max_{\Phi} \; B(\Theta, \Phi)$ (variational inference)

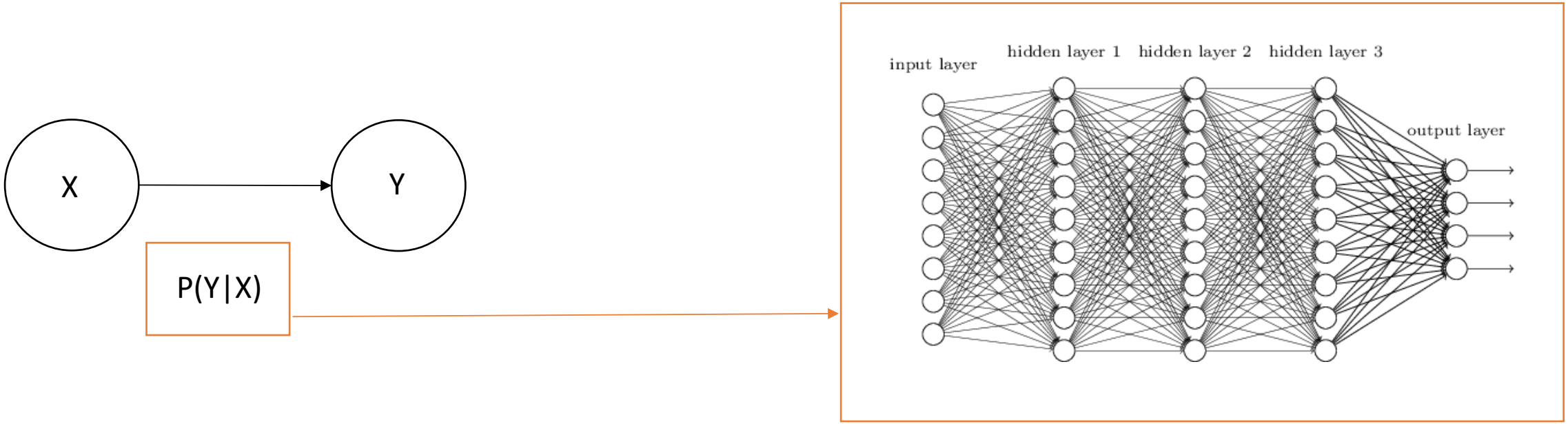
M-step: $\arg\max_{\Theta} \; B(\Theta, \Phi)$ (approximate learning)

Amortized Inference



- Bij making q(z|x) a function of x and sharing parameters ϕ , we can do very fast inference at test time (i.e. avoid iterative optimization of $q_{test}(z)$)

Deep NN as a glorified conditional distribution



The “Deepify” Operator

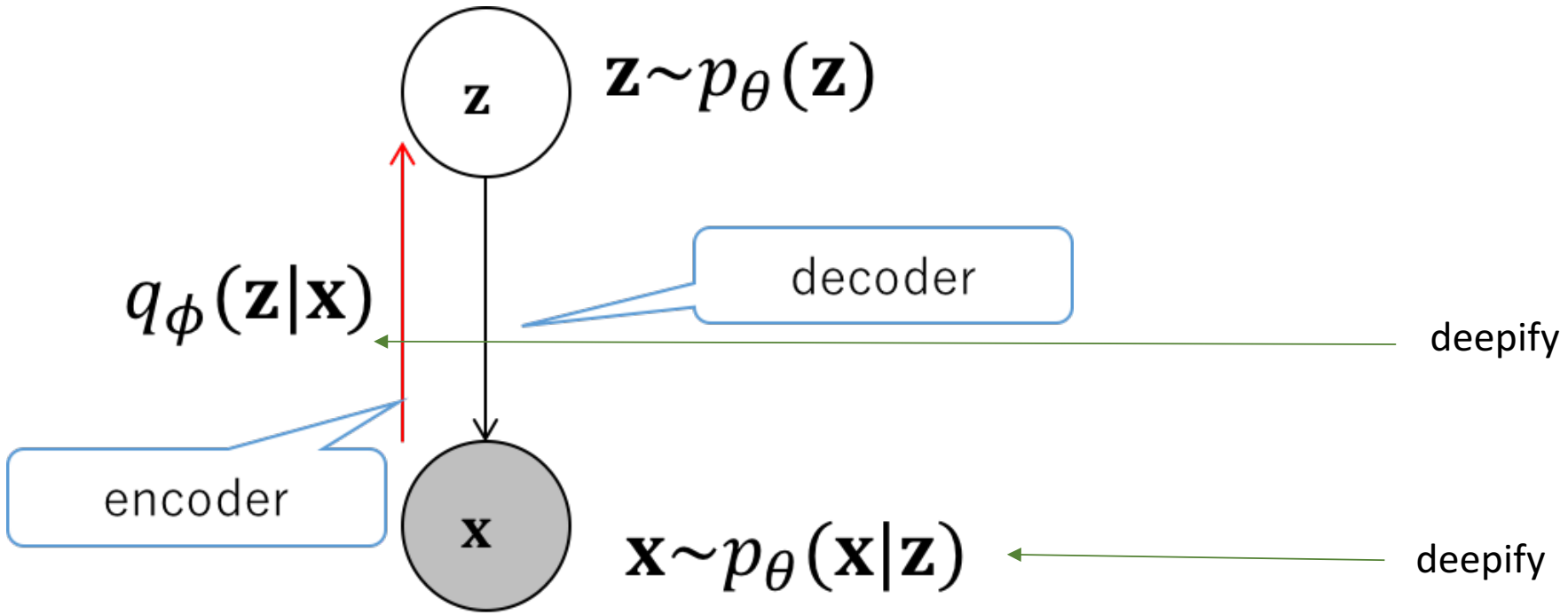
- Find a graphical model with conditional distributions and replace those with a deep NN.
- Logistic regression → deep NN.
- “deep survival analysis”. Cox’s proportional hazard function:

$$\lambda_i(t|\mathbf{x}_i) = \lambda_0(t)\exp\{\mathbf{x}_i'\boldsymbol{\beta}\}.$$

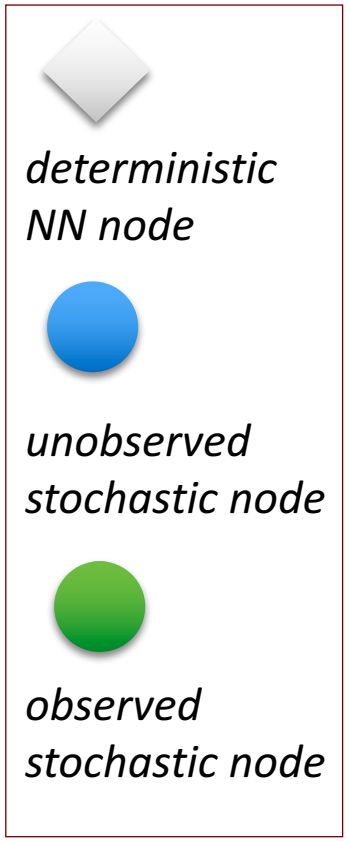
Replace with deep NN!

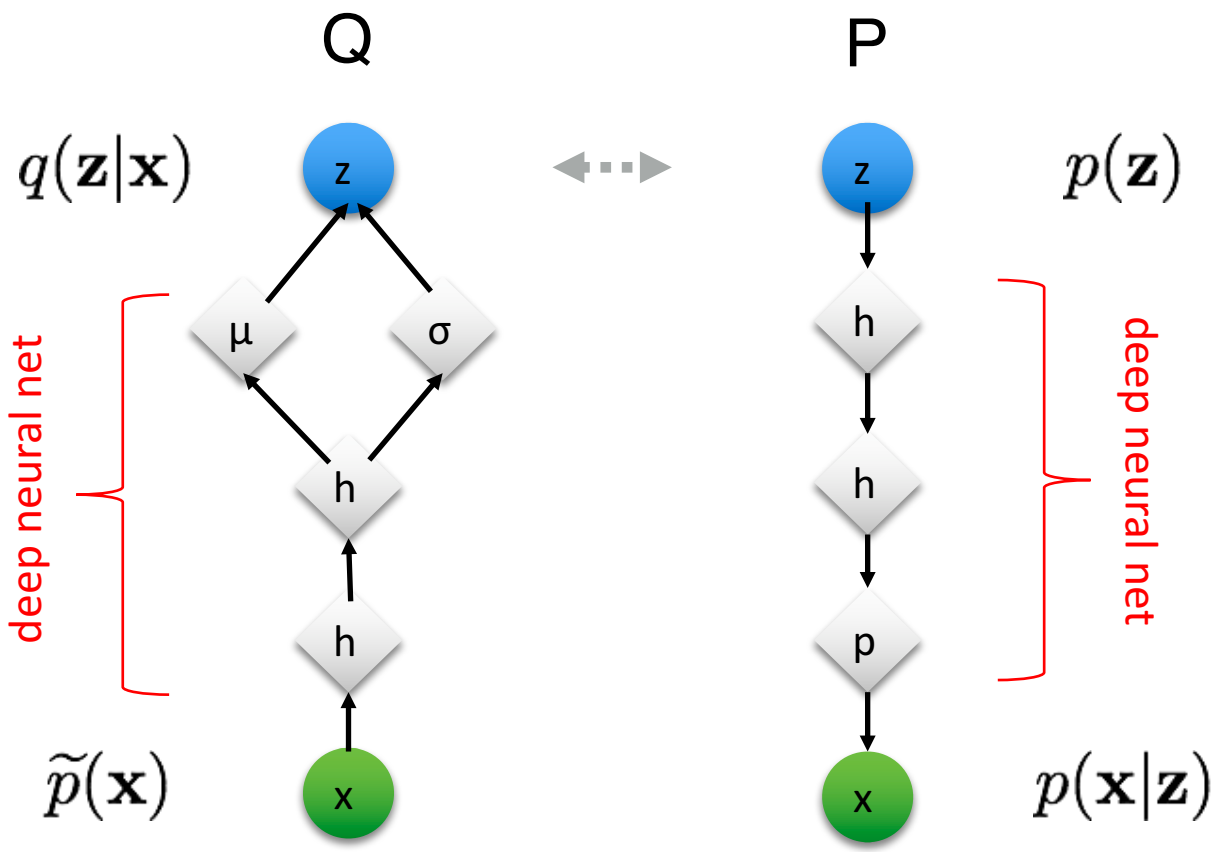
- Latent variable model: replace generative and recognition models with deep NNs:
  → ”Variational Autoencoder” (VAE).

Variational Autoencoder



Deep Generative Model: The Variational Auto-Encoder





Stochastic Variational Bayesian Inference

$$B(Q) = \sum_Z Q(Z|X,\Phi)(\log P(X|Z,\Theta) + \log P(Z) - \log Q(Z|X,\Phi))$$

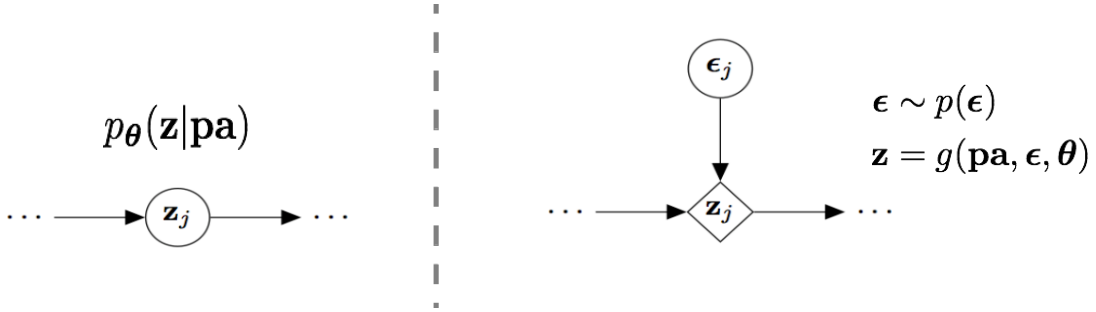
$$\nabla_\Phi B(Q) = \sum_Z \underbrace{Q(Z|X,\Phi)}_{\text{Sample Z}} \nabla_\Phi \log Q(Z|X,\Phi)(\log \underbrace{P(X|Z,\Theta)}_{\text{subsample mini-batch X}} + \log P(Z) - \log Q(Z|X,\Phi))$$

$$\nabla_\Phi B(Q) = \frac{1}{N}\frac{1}{S}\sum_{i=1}^{N}\sum_{s=1}^{S} \underbrace{\nabla_\Phi \log Q(Z_{is}|X_i,\Phi)(\log P(X_i|Z_{is},\Theta) + \log P(Z_{is}) - \log Q(Z_{is}|X_i,\Phi))}_{\text{very high variance}}$$

Reducing the Variance: The Reparametrization Trick

Kingma 2013, Bengio 2013, Kingma & Welling 2014

- Reparameterization:



- Applied to VAE:

$$\nabla_\Phi B(\Theta, \Phi) = \nabla_\Phi \int dz \; Q_\Phi(z|x)[\log P_\Theta(x, z) - \log Q_\Phi(z|x)]$$

$$\approx \nabla_\Phi [\log P_\Theta(x, z_s) - \log Q_\Phi(z_s|x)]_{z_s = g(\epsilon_s, \Phi)}, \qquad \epsilon_s \sim P(\epsilon)$$

- Example:

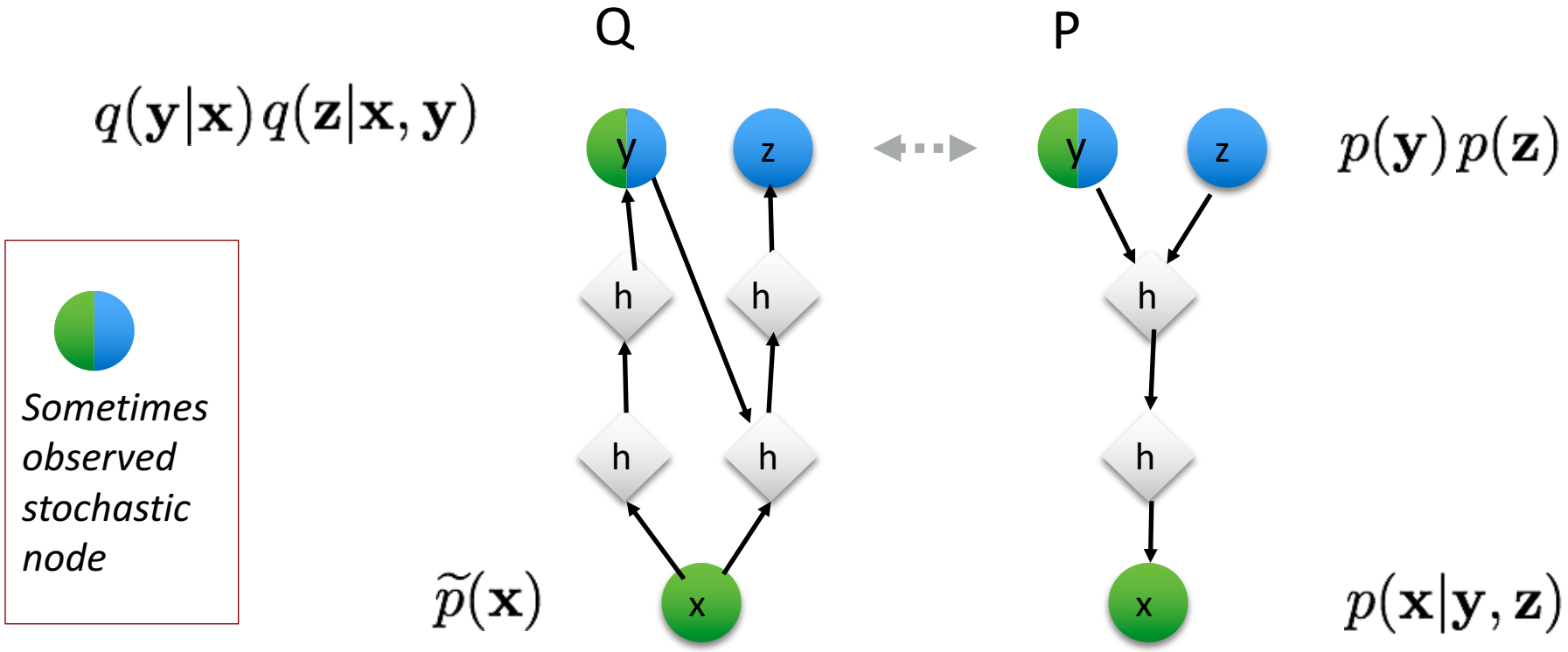
$$\nabla_\mu \int dz \mathcal{N}_z(\mu, \sigma) z$$

$$= \frac{1}{S} \sum_s z_s (z_s - \mu)/\sigma^2, \quad z_s \sim \mathcal{N}_z(\mu, \sigma)$$

$$\text{or } \frac{1}{S} \sum_s 1, \quad \epsilon_s \sim \mathcal{N}_\epsilon(0, 1), \quad z = \mu + \sigma\epsilon$$

Semi-Supervised VAE I

D.P. Kingma, D.J. Rezende, S. Mohamed, M. Welling, NIPS 2014

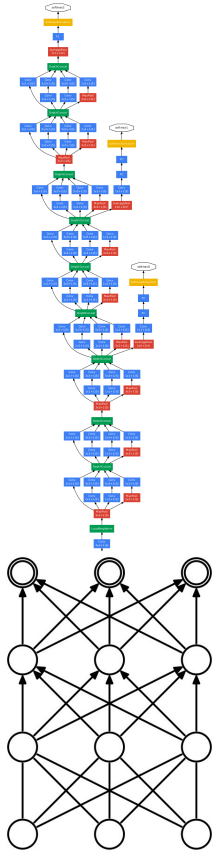


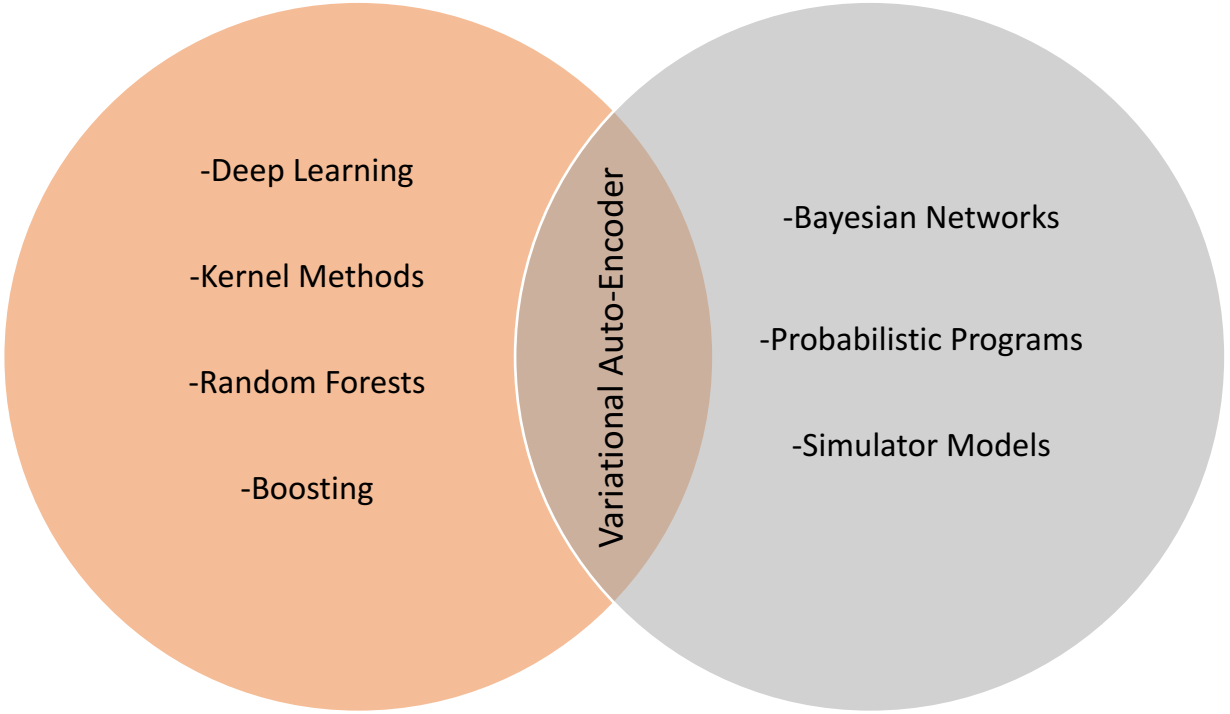
$$\mathcal{J} = \sum_{(\mathbf{x},y)\sim\widetilde{p}_l} \mathcal{L}(\mathbf{x},y) + \sum_{\mathbf{x}\sim\widetilde{p}_u} \mathcal{U}(\mathbf{x})$$ (normal VB objective)

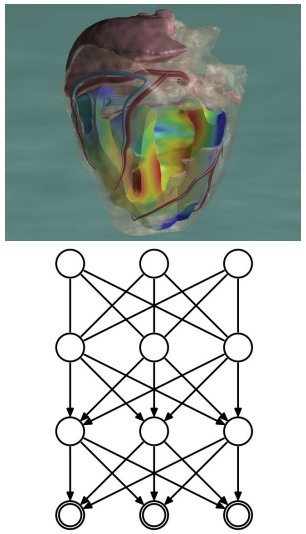
$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \mathbb{E}_{\widetilde{p}_l(\mathbf{x},y)}\left[-\log q_\phi(y|\mathbf{x})\right]$$ (boosting influence q(y|x))


0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

Discriminative or Generative?







- Advantages discriminative models:
  - Flexible map from input to target (low bias)
  - Efficient training algorithms available
  - Solve the problem you are evaluating on.
  - Very successful and accurate!

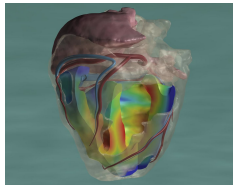
- Advantages generative models:
  - Inject expert knowledge
  - Model causal relations
  - Interpretable
  - Data efficient
  - More robust to domain shift
  - Facilitate un/semi-supervised learning

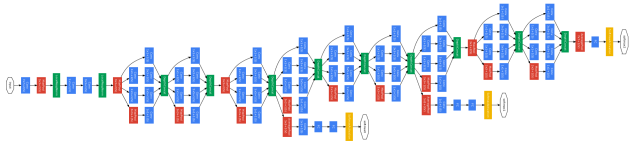
Big N vs. Small N?

We need *statistical efficiency*

We need *computational efficiency*

N = 100-1000

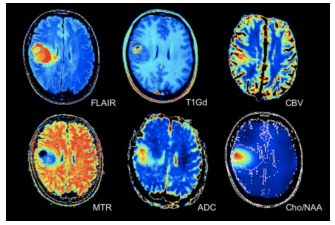




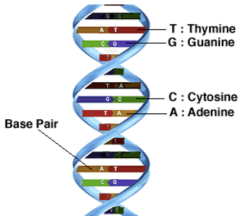
N=10^8-10^9

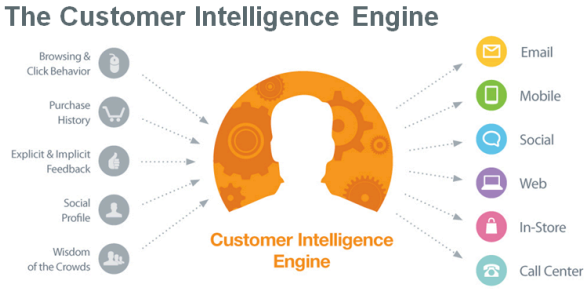
-Healthcare (p>>N)
-Generative, causal models generalize much better to new unknown situation (domain invariance)

-Customer Intelligence
-Finance
-Video/image
-Internet of Things

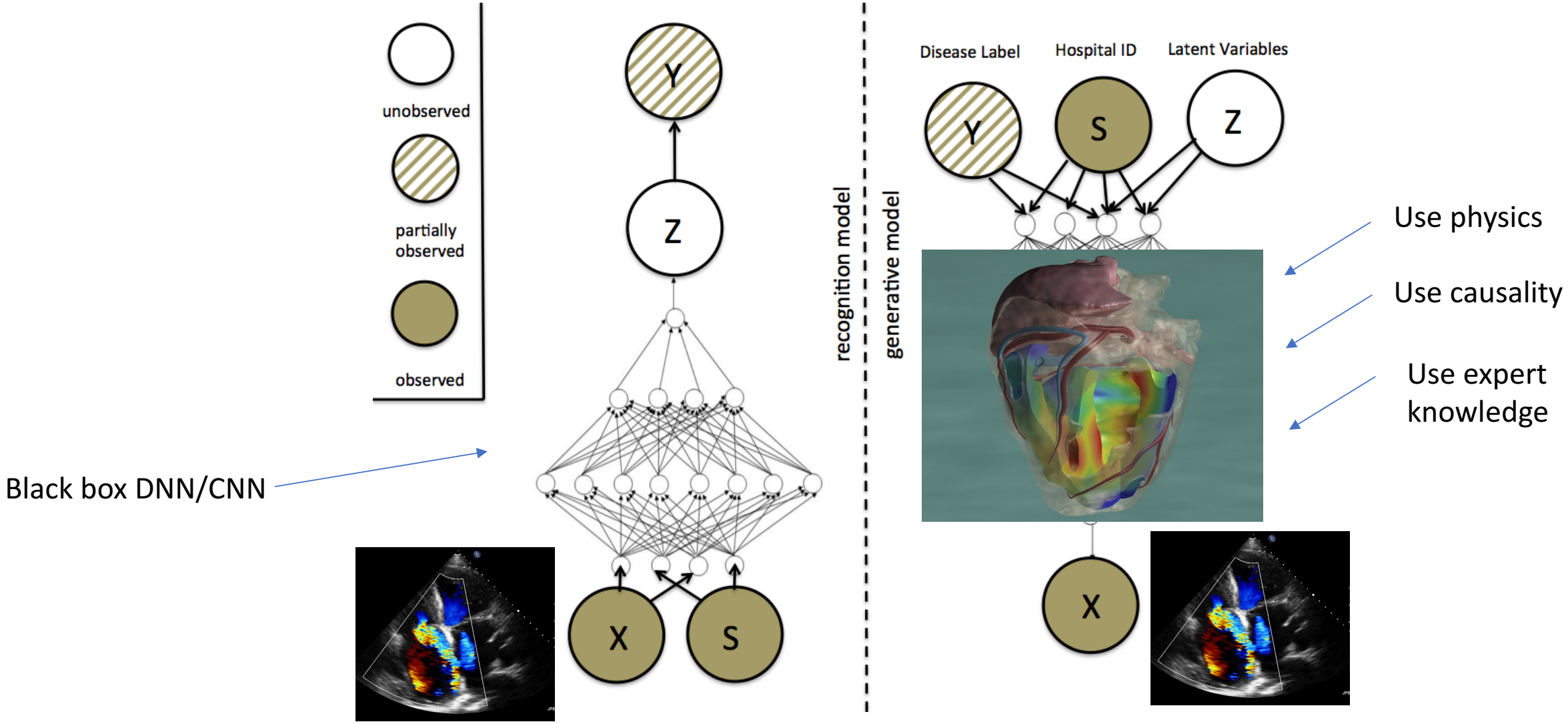


3 Billion ($3x10^9$) base pairs (DNA)





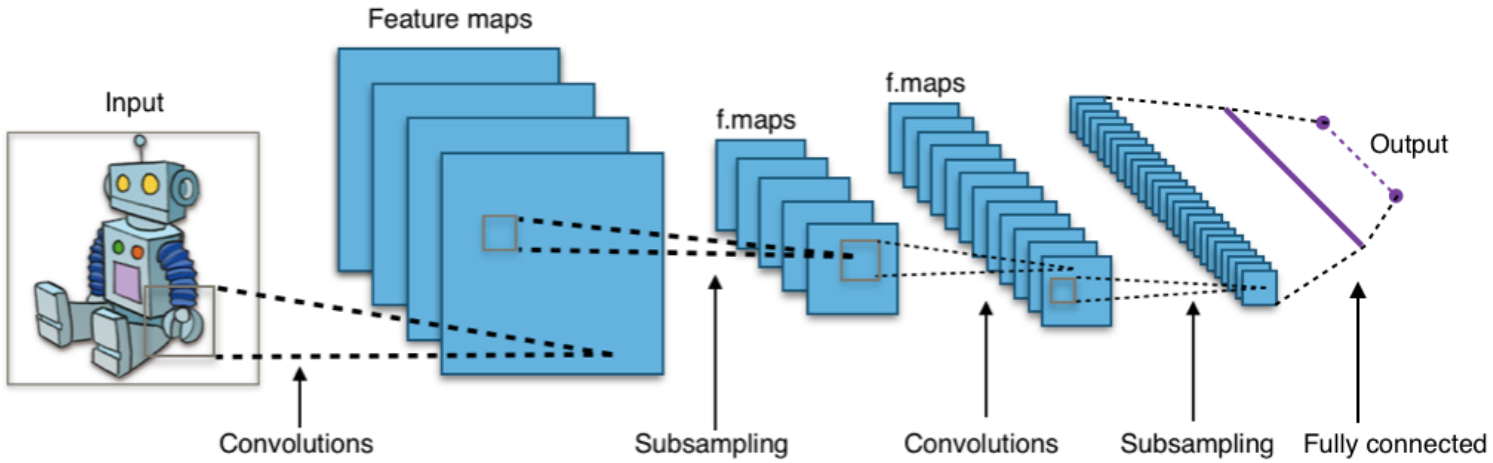
Combining Generative and Discriminative Models



Deep Convolutional Networks

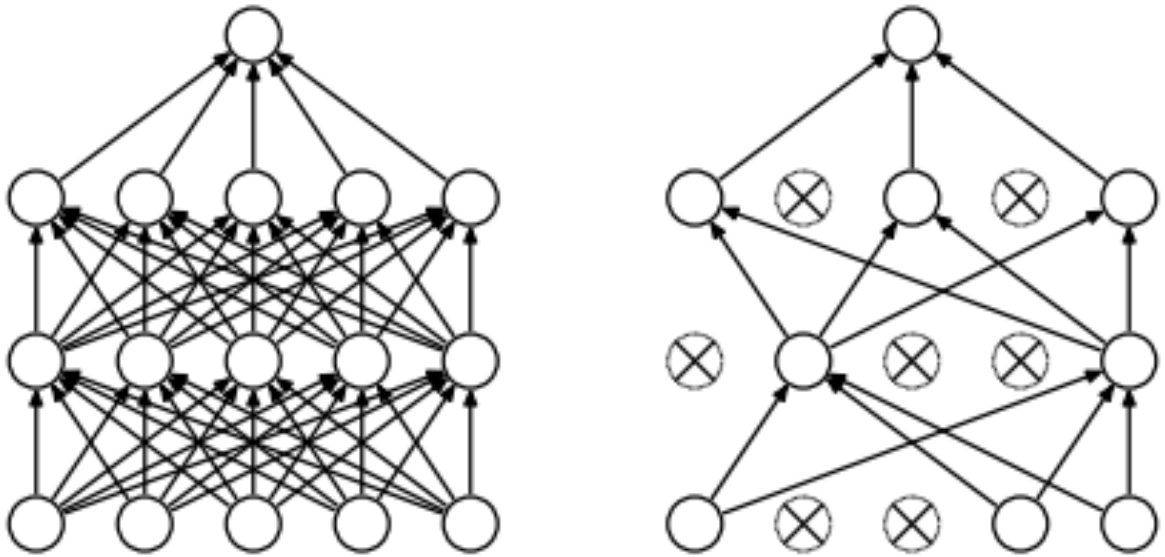
- Input dimensions have "topology": (1D, speech, 2D image, 3D MRI, 2+1D video, 4D fMRI)

Forward: Filter, subsample, filter, nonlinearity, subsample, …., classify



Backward: backpropagation (propagate error signal backward)

Dropout



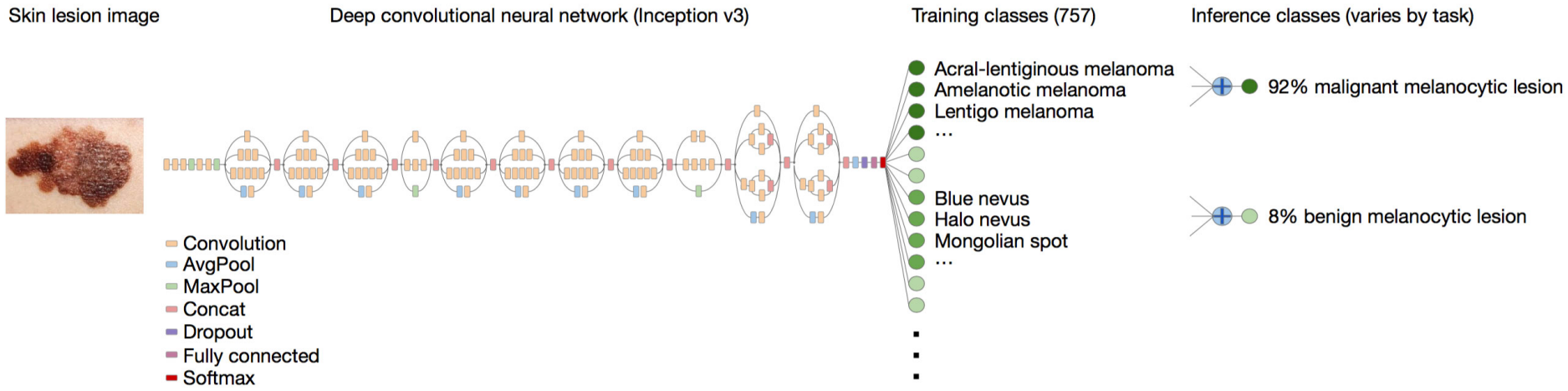
(a) Standard Neural Net (b) After applying dropout.

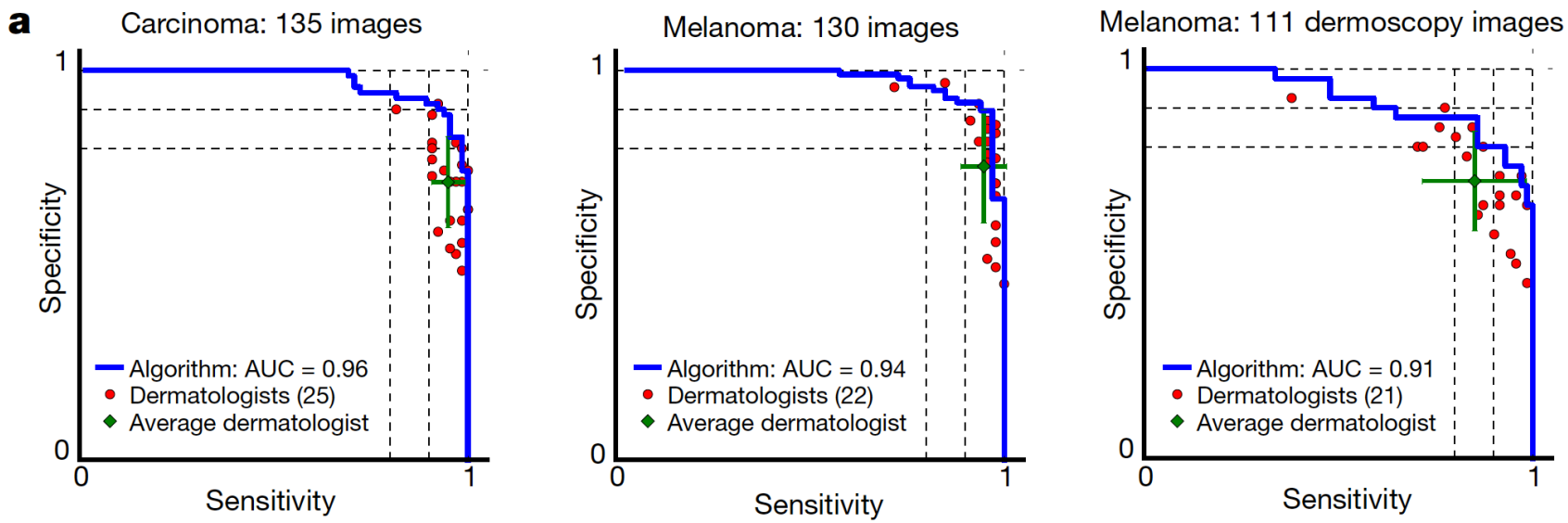
Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

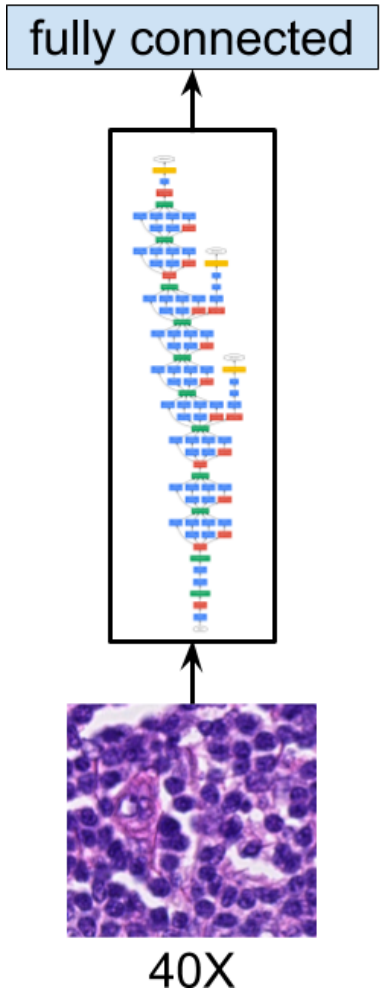
Example: Dermatology

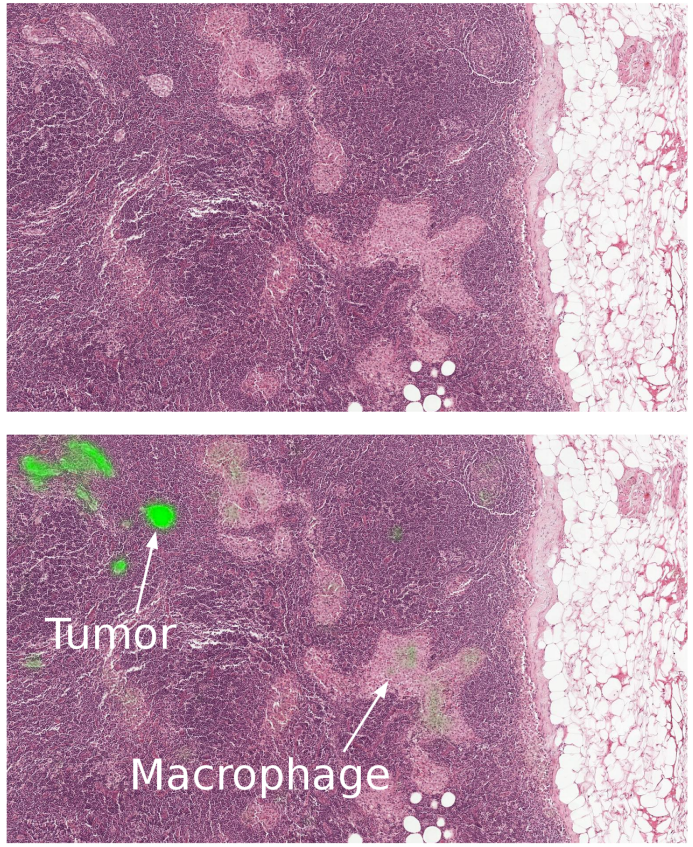
Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva[1]*, Brett Kuprel[1]*, Roberto A. Novoa[2,3], Justin Ko[2], Susan M. Swetter[2,4], Helen M. Blau[5] & Sebastian Thrun[6]




a
Carcinoma: 135 images
Melanoma: 130 images
Melanoma: 111 dermoscopy images
Specificity
Sensitivity
1
0
0
1
Algorithm: AUC = 0.96
Dermatologists (25)
Average dermatologist
Algorithm: AUC = 0.94
Dermatologists (22)
Average dermatologist
Algorithm: AUC = 0.91
Dermatologists (21)
Average dermatologist





Beter dan de patholoog

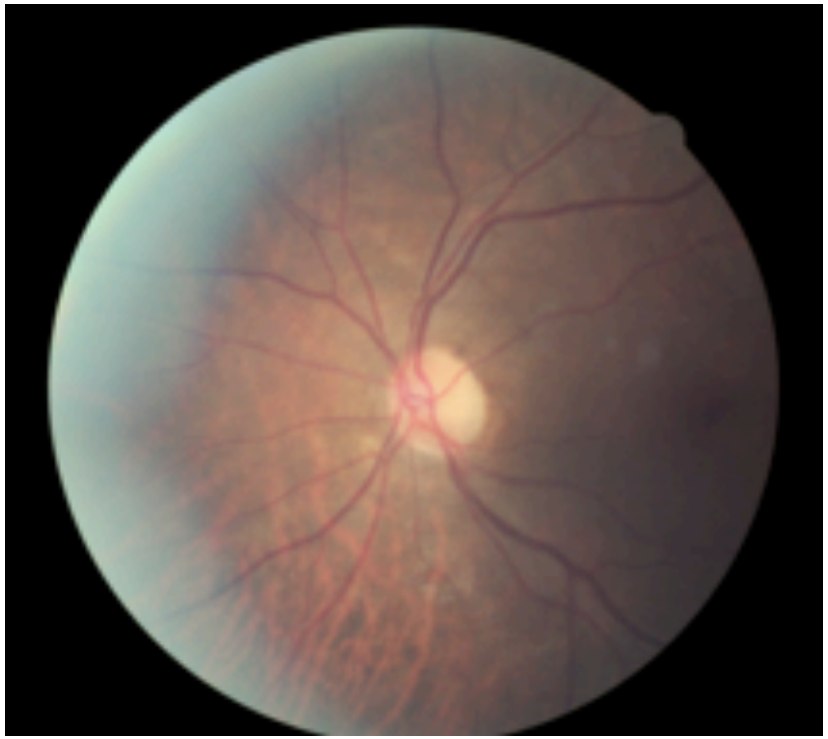
Datzelfde principe heeft Google nu toegepast op de data van het Radboud. Het algoritme werd geprogrammeerd om kankercellen te vinden op de foto's en vervolgens aan het werk gezet. Volgens de onderzoekers haalde het algoritme een score van 89 procent, terwijl een patholoog gemiddeld 73 procent haalt op dezelfde foto's.

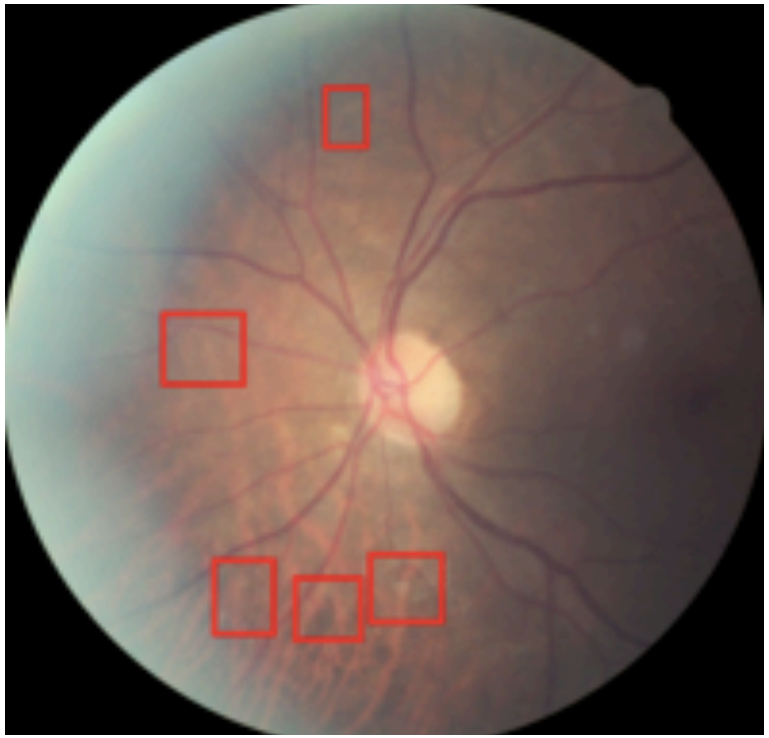
Example: Retinopathy

JAMA | Original Investigation | INNOVATIONS IN HEALTH CARE DELIVERY

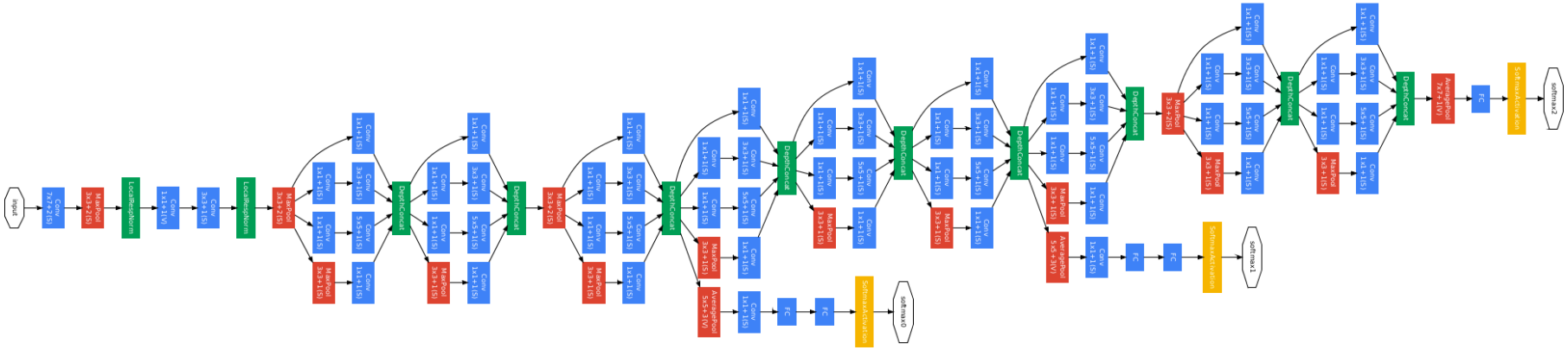
Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD; Lily Peng, MD, PhD; Marc Coram, PhD; Martin C. Stumpe, PhD; Derek Wu, BS; Arunachalam Narayanaswamy, PhD; Subhashini Venugopalan, MS; Kasumi Widner, MS; Tom Madams, MEng; Jorge Cuadros, OD, PhD; Ramasamy Kim, OD, DNB; Rajiv Raman, MS, DNB; Philip C. Nelson, BS; Jessica L. Mega, MD, MPH; Dale R. Webster, PhD





What do these Problems have in common?



It’s the same CNN in all cases: Inception-v3

So..., CNNs work really well.

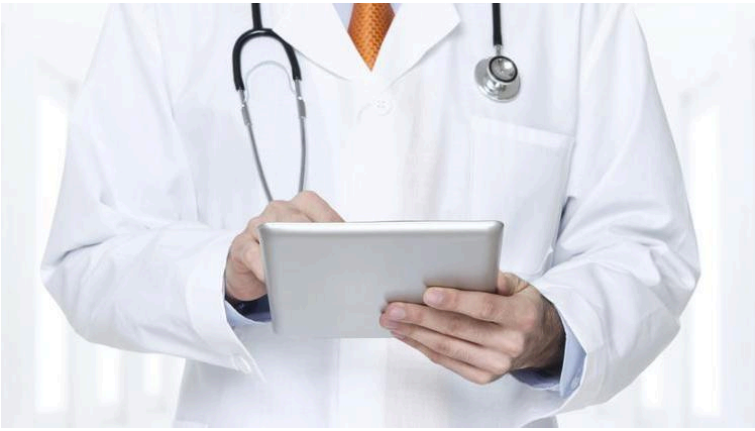
However:

- They are way too big
- They consume too much energy
- They use too much memory

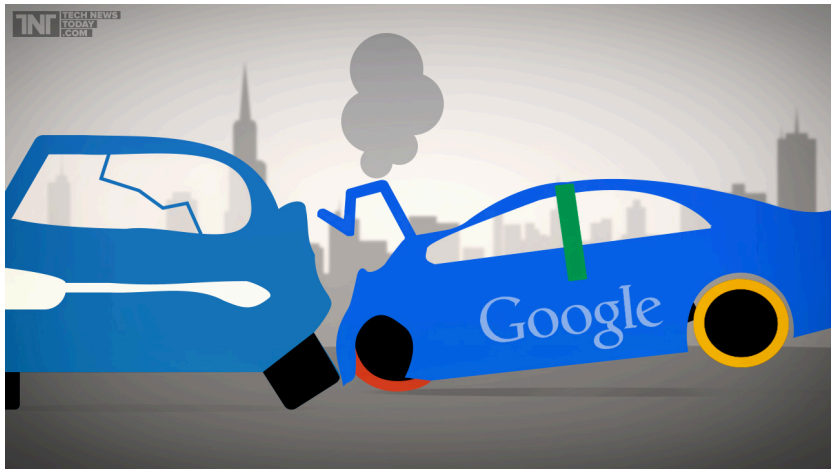
- → we need to make them more efficient!

Reasons for Bayesian Deep Learning

- Automatic model selection / pruning
- Automatic regularization
- Realistic prediction uncertainty (important for decision making)



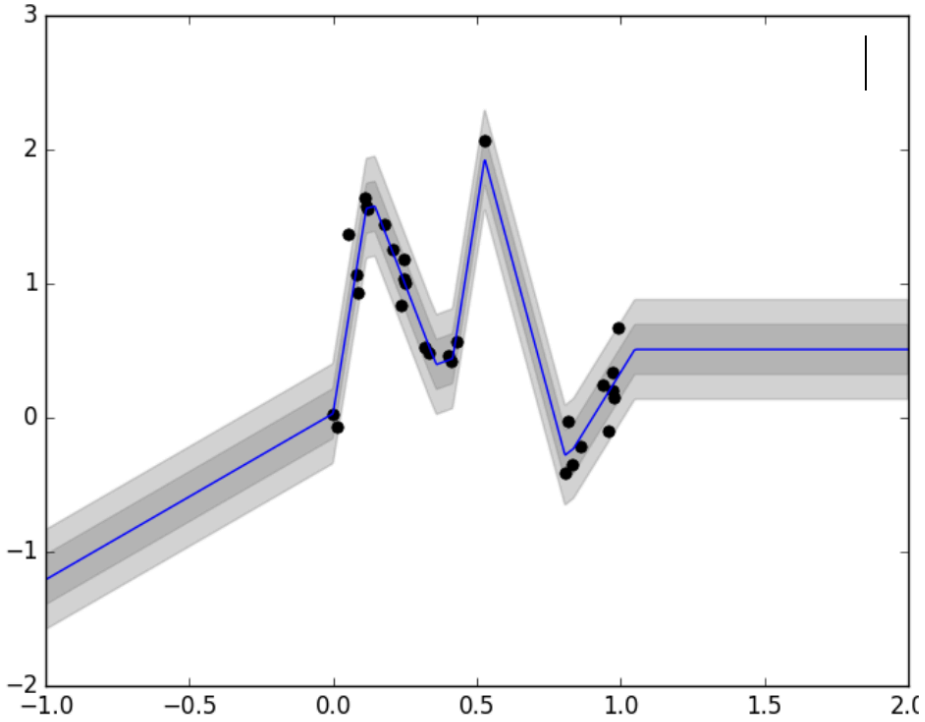
Computer Aided Diagnosis



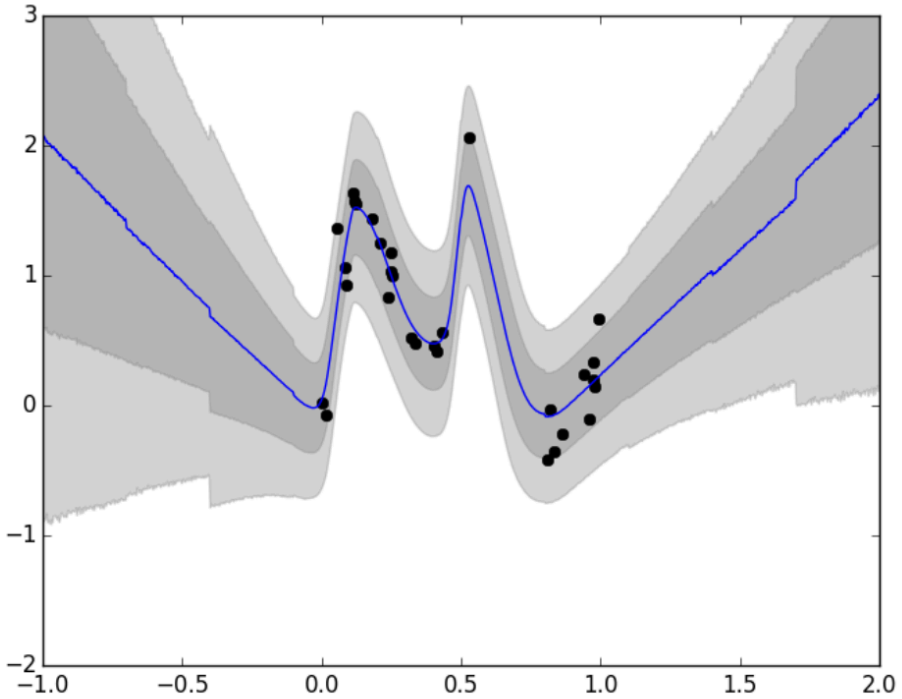
Autonomous Driving

Example

Increased uncertainty away from data



MAP estimate of parameters



Bayesian estimate of parameters

Bayesian Learning

$$P(X|M) = \int d\Theta \; P(X|\Theta, M)P(\Theta|M)$$ (model evidence)

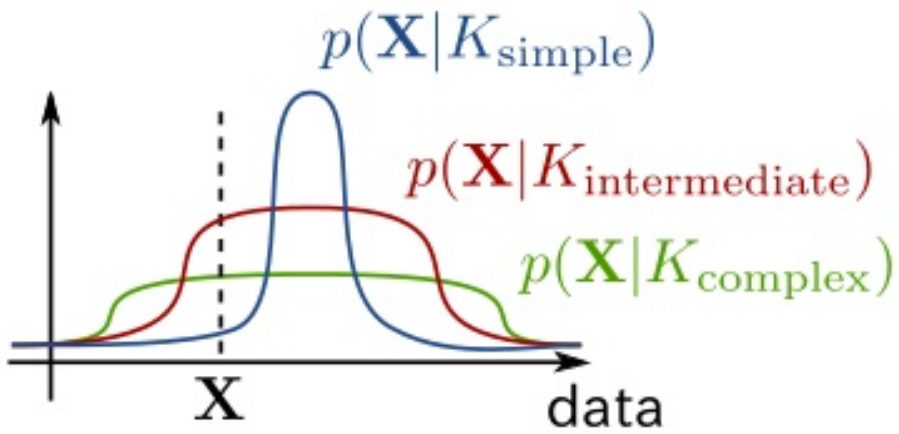
$$P(\Theta|X, M) = \frac{P(X|\Theta, M)P(\Theta|M)}{P(X|M)}$$ (posterior)

$$P(x|X, M) = \int d\Theta \; P(x|\Theta, M)P(\Theta|X, M)$$ (prediction)

$$P(X) = \sum_M P(X|M)P(M)$$ (evidence)

$$P(M|X) = \frac{P(X|M)P(M)}{P(X)}$$ (model selection)

Complex models can have lower marginal likelihood:



Picture credit: Kohei Hayashi[1,2] Shin-ichi Maeda[3] Ryohei Fujimaki[4]

Variational Bayes

$$\log P(X) \geq \int_{\Theta} d\Theta \quad Q(\Theta)\left[\log P(X|\Theta) + \log P(\Theta) - \log Q(\Theta)\right] \equiv B(Q(\Theta)|X)$$

$$= E_{Q(\Theta)}[\log P(X|\Theta)] - KL[Q(\Theta)||P(\Theta)])$$

Sparsifying & Compressing CNNs

- DNNs are vastly overparameterized (e.g. distillation, Bucilua et al 2006).
- Interpret variational bound as coding cost for data transmission (minimum description length)
- Idea: learn a soft weight sharing prior, a.k.a. quantize the weights (Nowlan & Hinton 1991, Ullrich et al 2016)

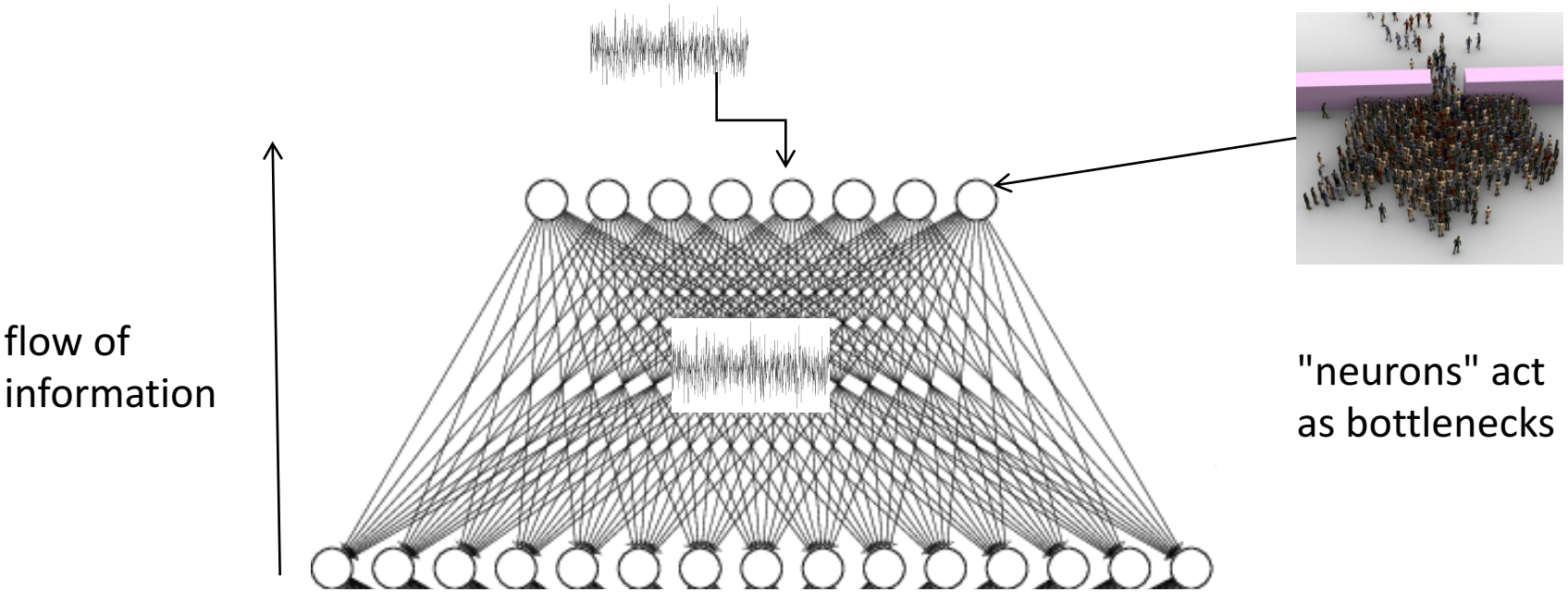
$$E_{Q(\Theta)}[\log P(X|\Theta)] - KL[Q(\Theta)||P(\Theta)])$$

error loss ~N

complexity loss ~const.

Full Bayesian Deep Learning

The signal in NNs are very robust to noise addition (e.g dropout)



THE PLAN:

- Marginalize out weights for the price of introducing stochastic hidden units.
- Reinterpret stochasticity on hidden units as dropout noise.
- Use sparsity inducing priors to prune weights / hidden units.

Stochastic Variational Bayes

$$B(Q(\Theta)|X) = \int_{\Theta} d\Theta \;\; Q(\Theta)\,[\log P(X|\Theta) + \log P(\Theta) - \log Q(\Theta)]$$

$$\nabla_{\Phi} B = \int_{\Theta} d\Theta \;\; \underbrace{Q_{\Phi}(\Theta)}_{\text{sample}}\; \nabla_{\Phi} \log Q_{\Phi}(\Theta)\,[\underbrace{\log P(X|\Theta)}_{\text{subsample mini-batch X}} + \log P(\Theta) - \log Q_{\Phi}(\Theta)]$$

$$\nabla_{\Phi} B = \frac{1}{S}\sum_{s=1}^{S} \underbrace{\nabla_{\Phi} \log Q_{\Phi}(\Theta_s)\left[\frac{N}{n}\sum_{i=1}^{n} \log P(x_i|\Theta_s) + \log P(\Theta_s) - \log Q_{\Phi}(\Theta_s)\right]}_{\text{very high variance}}$$

- Reparametrization? Yes but not enough: same sample Θ_s for all data cases X_i in minibatch induces high correlations between data-cases and thus high variance in gradient.

Local Reparametrization

Kingma, Salimans & Welling 2015

$$\int dW Q(W) \log P(Y|X,W) - \mathrm{KL}(Q(W)||P(W))$$

$$\left(P(X|\Theta) \rightarrow P(Y|W,X) \right)$$

Reparameterize:

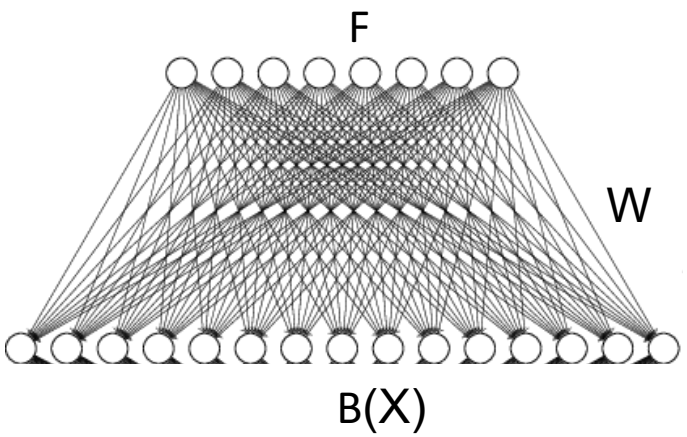
compute exactly

$$\int dW dF \; Q(W) \; L(Y, \phi(F)) \; I(F - W\phi(B(X))) =$$

$$\int dF \; Q(F|B) \; L(Y, \phi(F))$$

- Hidden units now become stochastic and correlated.
- We draw different samples F_{is} for different data-cases in the minibatch (and it's much less expensive than resampling all the weights independently per data case)

Conclusion: using this trick we can further reduce variance in the gradients

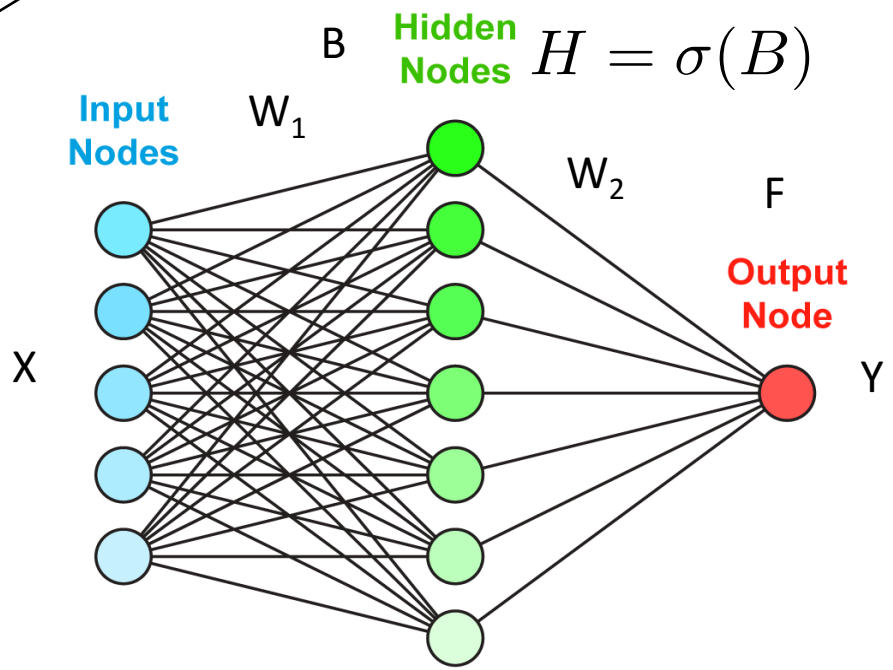


Two Layers

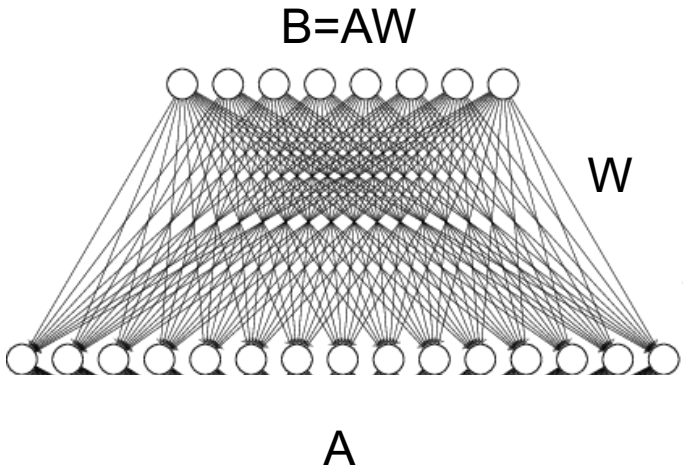
$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})}\left[\log p(\mathbf{Y}|\mathbf{F})\right] - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i))\right]$$

Now use the “normal” reparameterization trick

$$H = \sigma(B)$$



Variational Dropout



If $Q(W) = \prod_{ij} N(w_{ij}|\theta_{ij}, \alpha_{ij}\theta_{ij}^2)$ then $B = (A \cdot \xi)\theta; \quad \xi_{ij} \sim N(1, \alpha_{ij})$

multiplicative dropout noise

Conclusion: by using a special form of posterior we simulate dropout noise:
i.e. dropout can be understood as variational Bayesian inference with multiplicative noise.

Y Gal, Z Ghahramani 2016, Dropout as a **Bayesian** approximation: Representing model uncertainty in **deep learning**

S **Wang**, C Manning, Fast **dropout** training

Sparsity Inducing Priors

(Kingma, Salimans, Welling 2015, Mochanov, Ashuka, Vetrov 2017)

$$p(|w_{ij}|) \propto \frac{1}{|w_{ij}|}$$

(improper prior)

$$q(w_{ij} \,|\, \theta_{ij}, \alpha) = \mathcal{N}(w_{ij} \,|\, \theta_{ij}, \alpha\theta_{ij}^2)$$

(variational dropout posterior)

$$-D_{KL}(q(w_{ij} \,|\, \theta_{ij}, \alpha_{ij}) \,\|\, p(w_{ij})) \approx$$

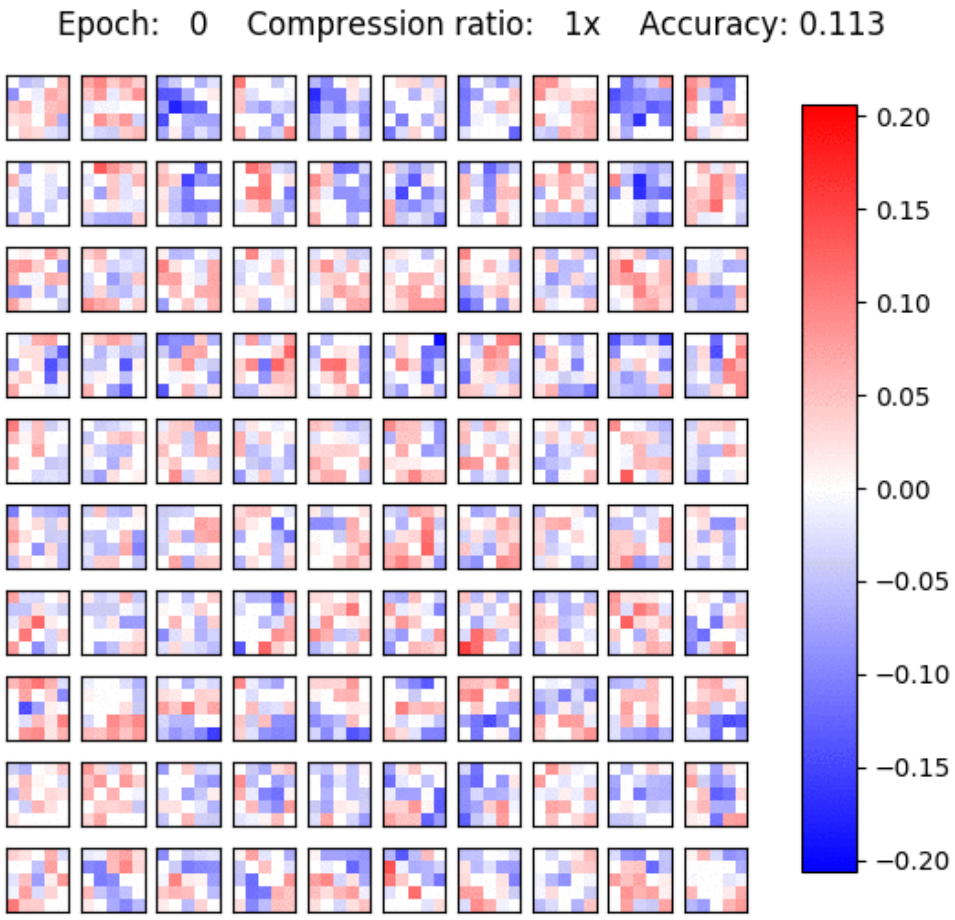
$$\approx k_1\sigma(k_2 + k_3 \log \alpha_{ij})) - 0.5 \log(1 + \alpha_{ij}^{-1}) + \mathrm{C}$$

$$k_1 = 0.63576 \quad k_2 = 1.87320 \quad k_3 = 1.48695$$

Learn dropout rate α_{ij}. When $\alpha_{ij} \to \infty$ weight is pruned

Conclusion: we can learn the dropout rates and prune unnecessary weights.

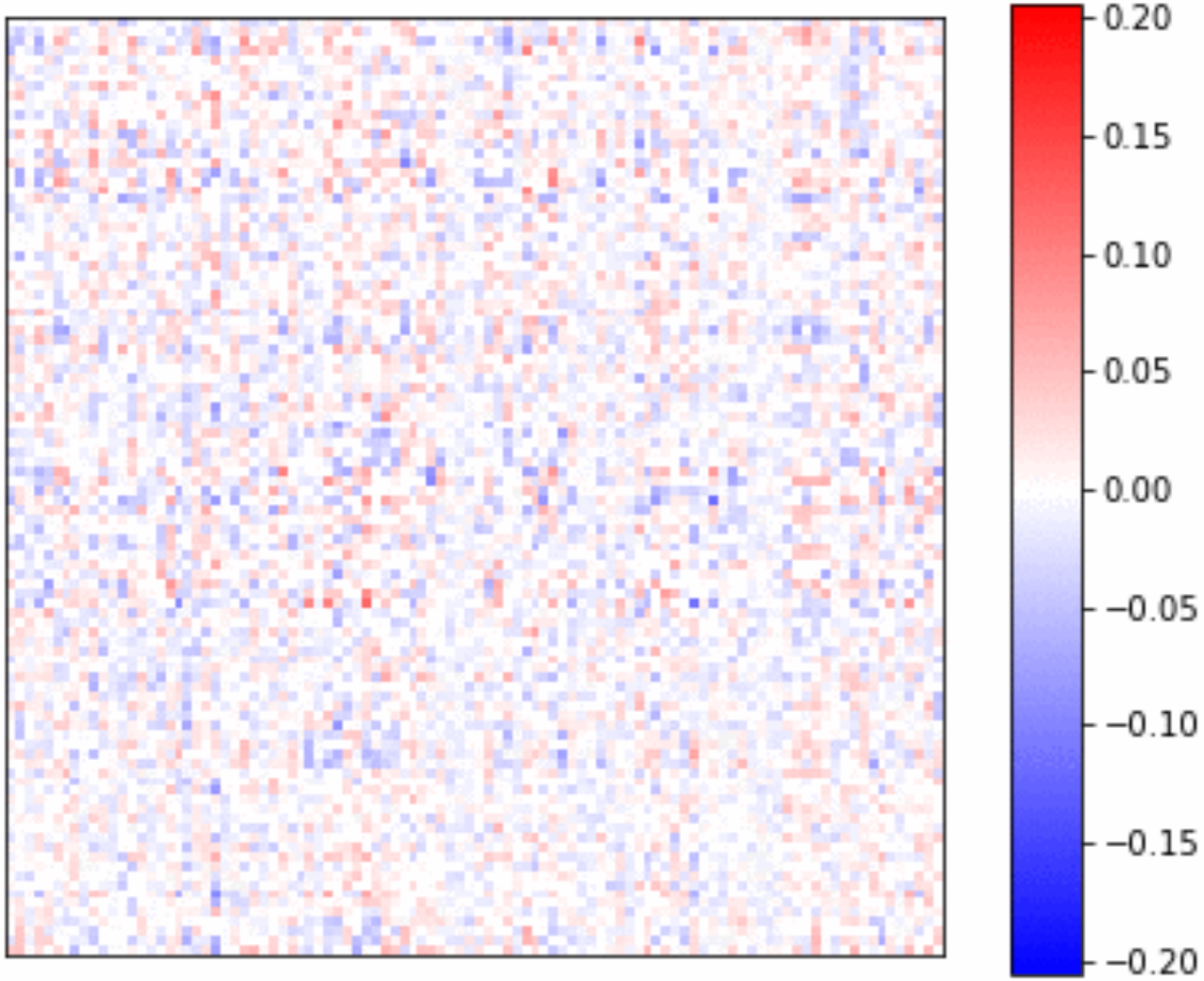
Variational Dropout



Animation: Molchanov, D., Ashukha, A. and Vetrov, D.

Epoch: 0 Compression ratio: 1x Accuracy: 0.113

Fully connected layer



Animation: Molchanov, D., Ashukha, A. and Vetrov, D.

Node (instead of Weight) Sparsification

(Louizos, Ullrich, Welling, 2017)

Use hierarchical prior: $$P(W, z) = \prod_{\text{hidden units i}} p(z_i) \prod_{\text{units j outgoing from node i}} P(w_{ij}|z_i)$$

Prior-posterior pair

$$p(\mathbf{W}, \mathbf{z}) \propto \prod_{i}^{A} \frac{1}{|z_i|} \prod_{ij}^{A,B} \mathcal{N}(w_{ij}|0, z_i^2)$$

$$q_\phi(\mathbf{W}, \mathbf{z}) = \prod_{i=1}^{A} \mathcal{N}(z_i|\mu_{z_i}, \sigma_{z_i}^2) \prod_{i,j}^{A,B} \mathcal{N}(w_{ij}|z_i\mu_{ij}, z_i^2\sigma_{ij}^2)$$

$$\sigma_{z_i}^2 = \mu_{z_i}^2 \alpha_i$$ (dropout multiplicative noise)

Conclusion: by using special, hierarchical priors we can prune hidden units instead of individual weights (which is much better for compression).

Preliminary Results

(Louizos, Ullrich, Welling 2017, submitted)

Network & size	Method	Pruned architecture	Bit-precision
LeNet-300-100	Sparse VD	512-114-72	8-11-14
784-300-100	BC-GNJ	278-98-13	8-9-14
	BC-GHS	311-86-14	13-11-10
LeNet-5-Caffe	Sparse VD	14-19-242-131	13-10-8-12
	GD	7-13-208-16	-
20-50-800-500	GL	3-12-192-500	-
	BC-GNJ	8-13-88-13	18-10-7-9
	BC-GHS	5-10-76-16	10-10–14-13
VGG	BC-GNJ	63-64-128-128-245-155-63- -26-24-20-14-12-11-11-15	10-10-10-10-8-8-8- -5-5-5-5-5-6-7-11
(2× 64)-(2× 128)- -(3×256)-(8× 512)	BC-GHS	51-62-125-128-228-129-38- -13-9-6-5-6-6-6-20	11-12-9-14-10-8-5- -5-6-6-6-8-11-17-10

Additional Bayesian Bonus:
By monitoring posterior fluctuations of weights one can determine their fixed point precision.

Model Original Error %	Method	$\frac{\lvert \mathbf{w} \neq 0 \rvert}{\lvert \mathbf{w} \rvert}$%	Compression Rates (Error %) Pruning	Fast Prediction	Maximum Compression
LeNet-300-100	DC	8.0	6 (1.6)	-	40 (1.6)
	DNS	1.8	28* (2.0)	-	-
1.6	SWS	4.3	12* (1.9)	-	64(1.9)
	Sparse VD	2.2	21(1.8)	84(1.8)	113 (1.8)
	BC-GNJ	10.8	9(1.8)	36(1.8)	58(1.8)
	BC-GHS	10.6	9(1.8)	23(1.9)	59(2.0)
LeNet-5-Caffe	DC	8.0	6*(0.7)	-	39(0.7)
	DNS	0.9	55*(0.9)	-	108(0.9)
0.9	SWS	0.5	100*(1.0)	-	162(1.0)
	Sparse VD	0.7	63(1.0)	228(1.0)	365(1.0)
	BC-GNJ	0.9	108(1.0)	361(1.0)	573(1.0)
	BC-GHS	0.6	156(1.0)	419(1.0)	771(1.0)
VGG	BC-GNJ	6.7	14(8.6)	56(8.8)	95(8.6)
8.4	BC-GHS	5.5	18(9.0)	59(9.0)	116(9.2)

- Compression rate of a factor 700x with no loss in accuracy!
- Compression rates for node sparsity are higher because encoding is cheaper.

Conclusions

- Deep learning is a no silver bullet: it is mainly very good at signal processing (auditory, image data)
- Optimization plays an important role in getting good solutions (e.g. reducing variance gradients)
- But… deep learning is more than optimization, it's also statistics!
- DL can be successfully combined with "classical" graphical models (as a glorified conditional distribution)
- Bayesian DL has a elegant interpretation as principled dropout
- Bayesian DL is ideally suited for compression
- There is a lot we do not understand about DL:
  - Why do they not overfit (easy to get 0 training error on data with random labels)
  - Why does SGD regularize so effectively?
  - Strange behavior in the face of adversarial examples
  - Huge over-parameterization (up to 400x)