ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Introduction to Machine Learning

Generative modeling with neural networks

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

Expectation-Maximization (Soft-EM)

- While not converged
  - E-step: calculate cluster membership weights ("Expected sufficient statistics") for each point (aka "responsibilities"):

    Calculate $\gamma_j^{(t)}(\mathbf{x}_i)$ for each *i* and *j* given estimates of $\mu^{(t-1)}, \Sigma^{(t-1)}, \mathbf{w}^{(t-1)}$ from previous iteration
  - M-step: Fit clusters to weighted data points (closed form Maximum likelihood solution!)

$$w_j^{(t)} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i) \quad \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)}$$

$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)(\mathbf{x}_i-\mu_j^{(t)})(\mathbf{x}_i-\mu_j^{(t)})^T}{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)}$$

Theory behind the EM Algorithm

- Can show that the EM algorithm is equivalent to the following procedure:

- E-Step: Calculate the expected complete data log-likelihood (= function of θ)

$$Q(\theta; \theta^{(t-1)}) = \mathbb{E}_{\mathbf{z}_{1:n}} \left[\log P(\mathbf{x}_{1:n}, \mathbf{z}_{1:n} \mid \theta) \mid \mathbf{x}_{1:n}, \theta^{(t-1)} \right]$$

$$= \sum_{z_{1:n}} P(z_{1:n} \mid x_{1:n}, \theta^{(t-1)}) \log P(x_{1:n}, z_{1:n} \mid \theta)$$

Complete data log-likelihood

- M-Step: Maximize $\theta^{(t)} = \arg\max_{\theta} Q(\theta; \theta^{(t-1)})$

Convergence of EM Algorithm

- Can prove that the EM Algorithm monotonically increases the likelihood

$$\log P(\mathbf{x}_{1:n} \mid \theta^{(t)}) \geq \log P(\mathbf{x}_{1:n} \mid \theta^{(t-1)})$$

- For Gaussian mixtures, EM is guaranteed to converge to a local maximum*
- Quality of solution highly depends on initialization! (as in k-Means)
- **Common strategy**: Rerun algorithm multiple times, and use the solution with largest likelihood

Convergence proof

- **Goal**: Maximize $\theta^* = \arg\max_\theta P(\mathbf{x}_{1:n} \mid \theta)$
- Can rewrite: $P(\mathbf{x}_{1:n} \mid \theta) = \dfrac{P(\mathbf{x}_{1:n}, \mathbf{z}_{1:n} \mid \theta)}{P(\mathbf{z}_{1:n} \mid \mathbf{x}_{1:n}, \theta)}$
- Take logs, and expectation w.r.t. $P(\mathbf{z}_{1:n} \mid \mathbf{x}_{1:n}, \theta^{(t-1)})$ ($\tilde{\theta}$)

$$\mathbb{E}_z\left[\log P(x|\theta) \mid x, \tilde{\theta}\right] = \mathbb{E}_z\left[\log \frac{P(x,z|\theta)}{P(z|x,\theta)} \mid x, \tilde{\theta}\right]$$

($\log P(x|\theta)$)

$$= \mathbb{E}_z\left[\log P(x,z|\theta) \mid x, \tilde{\theta}\right] - \mathbb{E}_z\left[\log P(z|x,\theta) \mid x, \tilde{\theta}\right]$$

($Q(\theta, \tilde{\theta})$) (?)

Convergence proof

- So far, we have

$$\log P(\mathbf{x} \mid \theta) = Q(\theta, \theta^{(t-1)}) - \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta) \mid \mathbf{x}, \theta^{(t-1)}]$$

$\ell(\theta)$

- Want to show $\log P(\mathbf{x} \mid \theta^{(t)}) \geq \log P(\mathbf{x} \mid \theta^{(t-1)})$

Suffices to show that

(1) $Q(\theta^{(t)}, \theta^{(t-1)}) \geq Q(\theta^{(t-1)}, \theta^{(t-1)})$ ✓ Since in M-step $\theta^{(t)} = \arg\max_{\theta} Q(\theta, \theta^{(t-1)})$

(2) $\ell(\theta^{(t)}) \leq \ell(\theta^{(t-1)})$

Convergence proof

- So far, we have

$$\log P(\mathbf{x} \mid \theta) = Q(\theta, \theta^{(t-1)}) - \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta) \mid \mathbf{x}, \theta^{(t-1)}]$$

- Want to show $\log P(\mathbf{x} \mid \theta^{(t)}) \geq \log P(\mathbf{x} \mid \theta^{(t-1)})$

- Compare term by term:
  - EM-Algorithm guarantees (strict inequality unless converged):

$$Q(\theta^{(t)}, \theta^{(t-1)}) \geq Q(\theta^{(t-1)}, \theta^{(t-1)})$$

  - Remains to show:

$$\mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t-1)}) \mid \mathbf{x}, \theta^{(t-1)}] \geq \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t)}) \mid \mathbf{x}, \theta^{(t-1)}]$$

Want to show

$$\mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t-1)}) \mid \mathbf{x}, \theta^{(t-1)}] \geq \mathbb{E}_{\mathbf{z}}[\log P(\mathbf{z} \mid \mathbf{x}, \theta^{(t)}) \mid \mathbf{x}, \theta^{(t-1)}]$$

$$\Leftrightarrow \mathbb{E}_{z}\left[\log P(z|x,\theta^{(t-1)}) - \log P(z|x,\theta^{(t)}) \,\middle|\, x, \theta^{(t-1)}\right] \geq 0$$

$$= \mathbb{E}_{z}\left[\log \frac{P(z|x,\theta^{(t-1)})}{P(z|x,\theta^{(t)})} \,\middle|\, x, \theta^{(t-1)}\right] = \sum_{z_{1:n}} P(z_{1:n}|x_{1:n},\theta^{(t-1)}) \log \frac{P(z_{1:n}|x_{1:n},\theta^{(t-1)})}{P(z_{1:n}|x_{1:n},\theta^{(t)})}$$

$$= KL\left(P(z|x,\theta^{(t-1)}) \,\|\, P(z|x,\theta^{(t)})\right) \geq 0$$

(p) (q)

□

For distributions $p(x), q(x)$ ← $p(x)=0 \Rightarrow q(x)=0$

$$KL(p\|q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] = \begin{cases} \sum_x p(x) \log \frac{p(x)}{q(x)} & \text{for discrete} \\ \int p(x) \log \frac{p(x)}{q(x)} & \text{for continuous} \end{cases} \geq 0$$

In general, $KL(p\|q) \neq KL(q\|p)$; $p=q \Rightarrow KL(p\|q)$

Converse also holds (under some conditions)

How about Hard EM?

- Can show that Hard EM performs alternating optimization on the complete data likelihood

- E step: $z_{1:n}^{(t)} = \arg\max_{z_{1:n}} P(\mathbf{x}_{1:n}, z_{1:n} \mid \theta^{(t-1)})$

$$= \arg\max_{z} P(x_{1:n} \mid \theta^{(t-1)}) \cdot P(z_{1:n} \mid \theta^{(t-1)}, x_{1:n})$$

indep of z

$$= \arg\max_{z_{1:n}} \prod_{i=1}^{n} P(z_i \mid x_{1:i}, \theta^{(t-1)}) \quad \hookrightarrow \quad z_i^{(t)} = \arg\max_{z} P(z \mid x_i, \theta^{(t-1)})$$

- M step: $\theta^{(t)} = \arg\max_{\theta} P(\mathbf{x}_{1:n}, z_{1:n}^{(t)} \mid \theta)$

- The algorithm converges to a local optimum of

$$\max_{z_{1:n}, \theta} P(\mathbf{x}_{1:n}, z_{1:n} \mid \theta)$$

← Note: Different from $\max_{\theta} P(x_i \mid \theta)$

EM Algorithm more generally

- The EM algorithm is much more widely applicable
  - Can be used whenever the E and M steps are tractable.
  - That means, we must be able to *compute* and *maximize* the complete data likelihood
- Can use this, e.g., for
  - (some) missing features
  - Likelihoods beyond Gaussian (e.g., categorical)

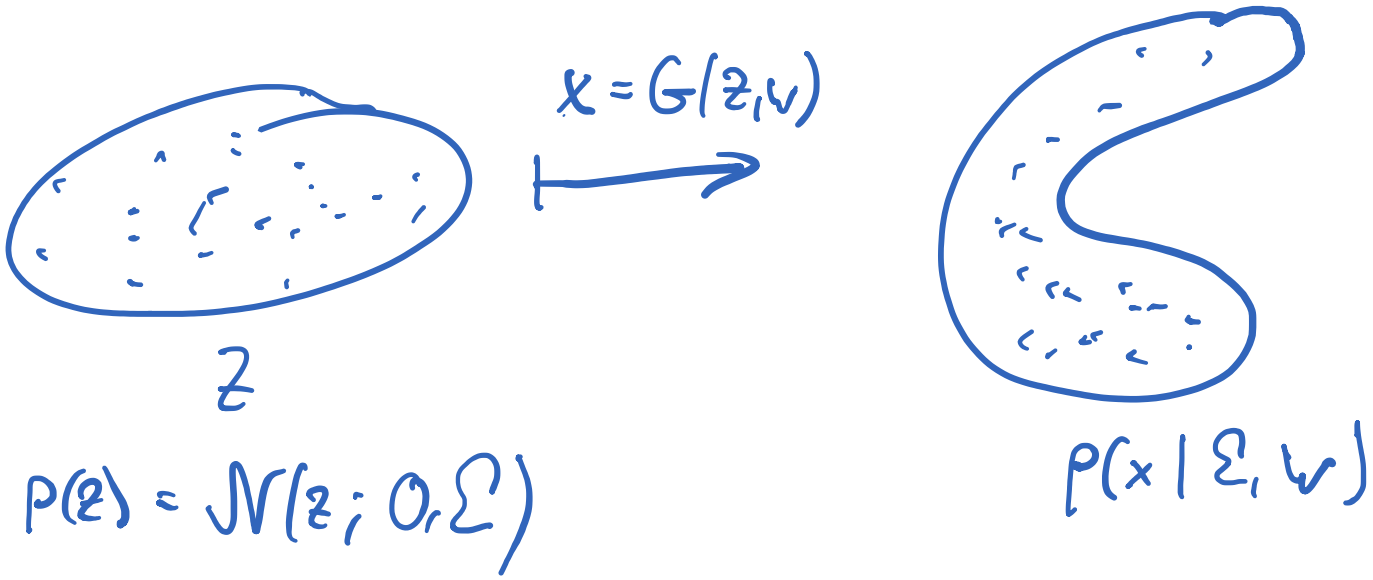
Neural nets for generative models?

- So far, have considered very simple probabilistic models (hand-selected priors and likelihood functions)
- These fail to capture complex, high-dimensional data types (images, audio, …)

- Can we use flexible models like neural networks for generative modeling?

Implicit generative models

- Given sample of (unlabeled) points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- **Goal**: Learn model $\mathbf{X} = G(\mathbf{Z}; \mathbf{w})$
  where *Z* is a "simple" distribution (e.g., low-dimensional Gaussian), and *G* some flexible nonlinear function (neural net)

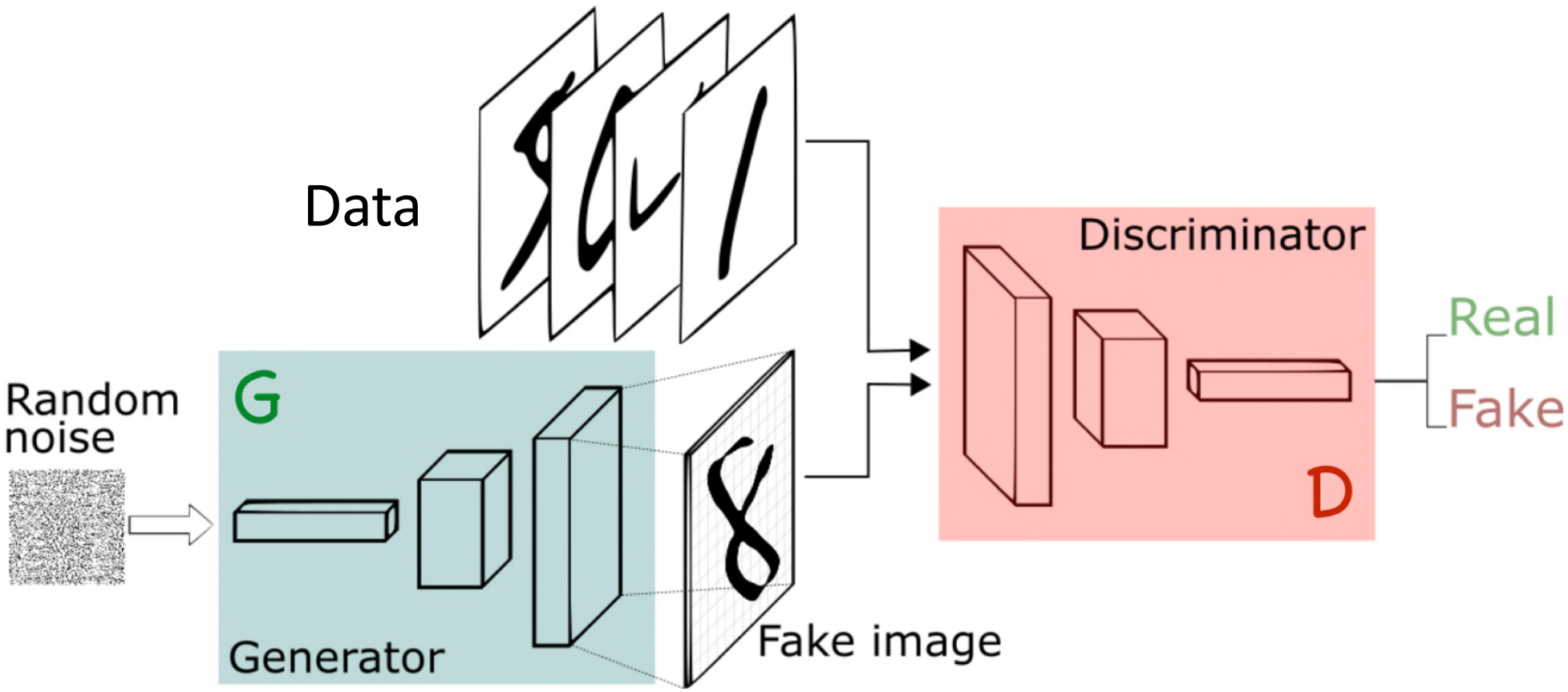


Implicit generative models

- Given sample of (unlabeled) points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- **Goal**: Learn model $\mathbf{X} = G(\mathbf{Z}; \mathbf{w})$
  where *Z* is a "simple" distribution (e.g., low-dimensional Gaussian), and *G* some flexible nonlinear function (neural net)

- **Key challenge**: Hard to compute likelihood of the data!
- Thus, need alternative/surrogate objective functions for training!
- Variants
  - Variational autoencoders (VAEs)
  - Generative adversarial networks (GANs)

GANs [Goodfellow et al 2014]



Key idea: Optimize parameters **w** to make samples from model hard to distinguish from data sample

Use discriminative learning to train generative model!

GANs [Goodfellow et al'14, Salimans et al'16]

Progress

[Brundage et al, 2018]

Newer examples [Kerras et al ’18]

Generative Adversarial Networks

- Simultaneously train two neural networks
  - Generator G tries to produce realistic examples
  - Discriminator D tries to detect "fake" examples
- Can view as a game:

$D: \mathbb{R}^d \to [0,1]$ wants: $D(x) = \begin{cases} \approx 1 & \text{if } x \text{ is "real"} \\ \approx 0 & \text{if } x \text{ is "fake"} \end{cases}$

$G: \mathbb{R}^m \to \mathbb{R}^d$ want: $D(G(z)) \approx 1$ for samples z

$$\min_G \max_D \underbrace{\mathbb{E}_{x \sim \text{Data}} \log D(x) + \mathbb{E}_{z \sim \text{Noise}} \log\left(1 - D(G(z))\right)}_{M(G,D)}$$

Generative Adversarial Networks

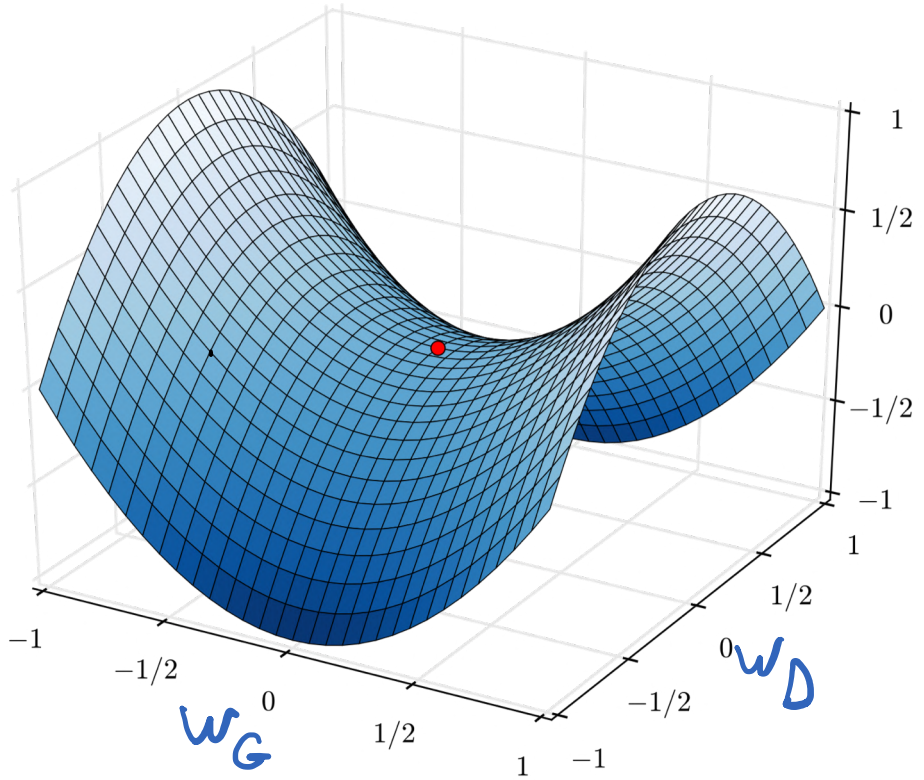
- Simultaneously train two neural networks
  - Generator G tries to produce realistic examples
  - Discriminator D tries to detect ”fake” examples
- Can view as a game:

$$\min_{\mathbf{w}_G} \max_{\mathbf{w}_D} \underbrace{\mathbb{E}_{\mathbf{x}\sim\text{Data}} \log D(\mathbf{x};\mathbf{w}_D) + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}} \log\Big(1 - D\big(G(\mathbf{z};\mathbf{w}_G);\mathbf{w}_D\big)\Big)}_{M(\mathbf{w}_G,\mathbf{w}_D)}$$

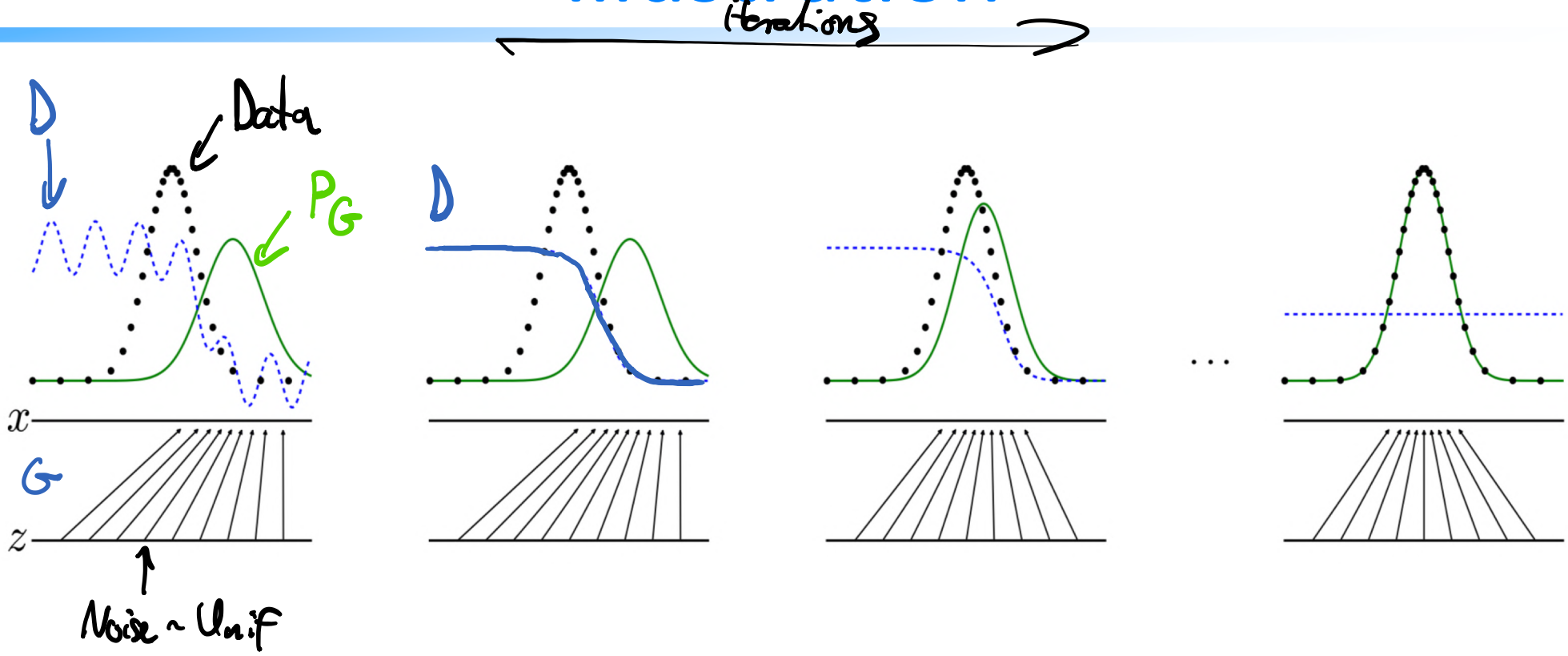
Training GANs

- Training a GAN requires finding a saddle point rather than a (local) minimum

$$\min_{\mathbf{w}_G} \max_{\mathbf{w}_D} M(\mathbf{w}_G, \mathbf{w}_D)$$



Illustration



[Goodfellow et al 2014]

Convergence guarantee [Goodfellow et al 2014]

- If G and D have "enough capacity", then the data generating distribution is indeed a saddle point of

$$\min_{\mathbf{w}_G} \max_{\mathbf{w}_D} M(\mathbf{w}_G, \mathbf{w}_D)$$

Key idea: $l^*(w_G) = \max_{w_D} M(w_G; w_D)$ is (up to constants) the Jensen-Shannon divergence $JS(p_{Data} \| p_G)$

$JS(p \| q) = \frac{1}{2} KL(p \| \frac{p+q}{2}) + \frac{1}{2} KL(q \| \frac{p+q}{2})$; $JS(p\|q)=0 \Leftrightarrow p=q$

- In practice, train on finite sample
  - ➔ danger of "memorization"
  - ➔ Discriminator should not be "too powerful"

Simultaneous gradient descent

- Common training approach: Simultaneously apply (stochastic) gradient descent to the empirical GAN objective

$$\mathbf{w}_G^{(t+1)} = \mathbf{w}_G^{(t)} - \eta_t \nabla_{\mathbf{w}_G} M(\mathbf{w}_G, \mathbf{w}_D^{(t)})$$

$$\mathbf{w}_D^{(t+1)} = \mathbf{w}_D^{(t)} + \eta_t \nabla_{\mathbf{w}_D} M(\mathbf{w}_G^{(t)}, \mathbf{w}_D)$$

min!

max!

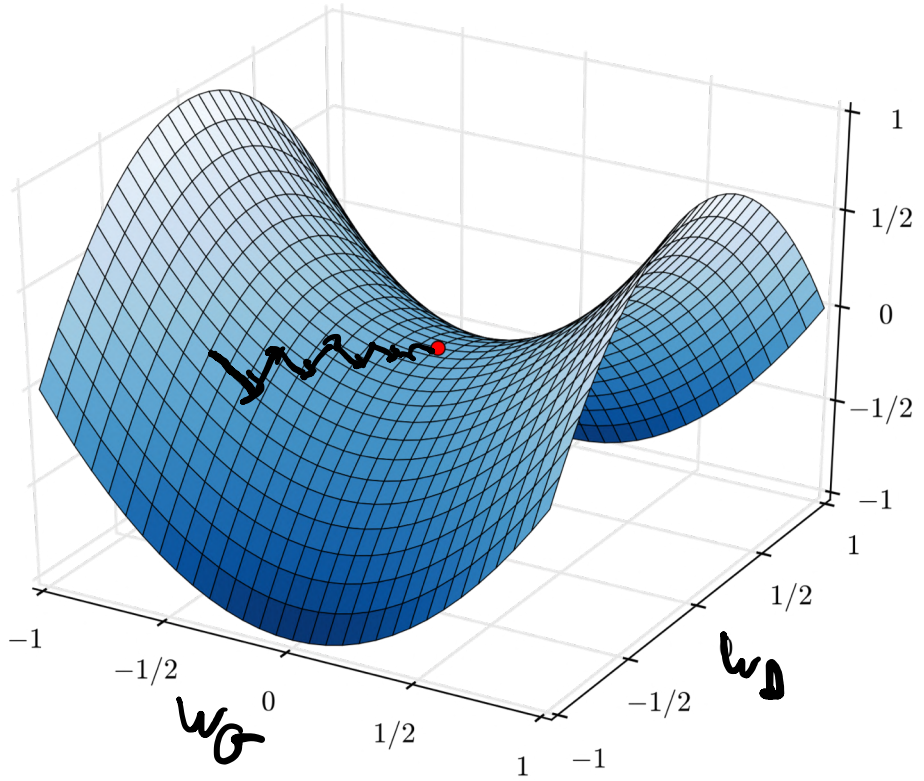
may also use t+1

- Gradients are approximated by samples of (minibatches of) data points

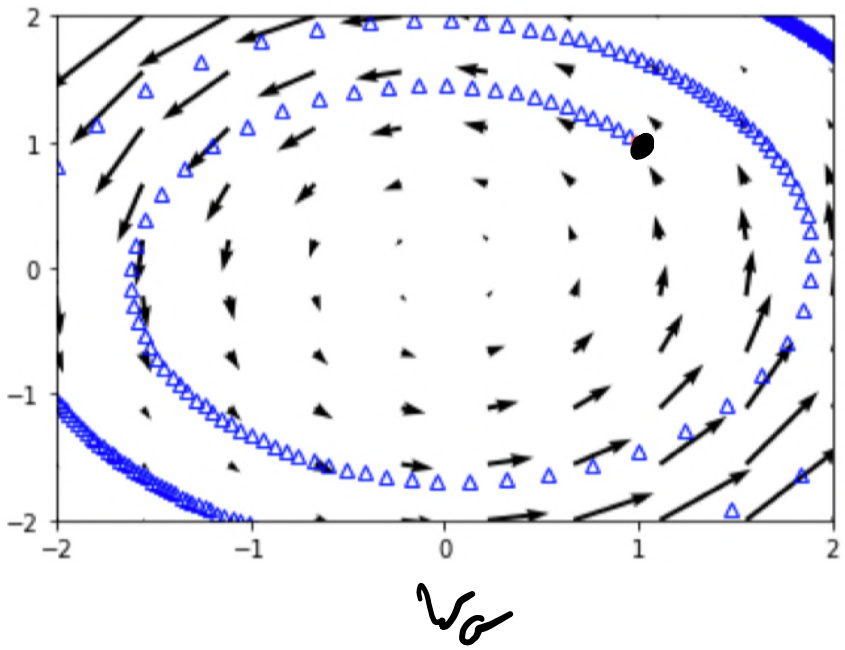
Training GANs

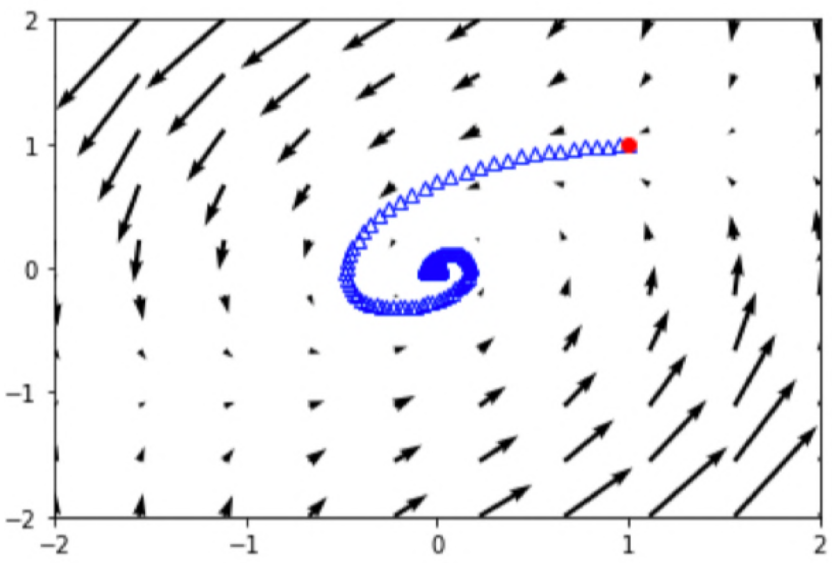
- Training a GAN requires finding a saddle point rather than a (local) minimum

$$\min_{\mathbf{w}_G} \max_{\mathbf{w}_D} M(\mathbf{w}_G, \mathbf{w}_D)$$



Challenge: oscillations / divergence

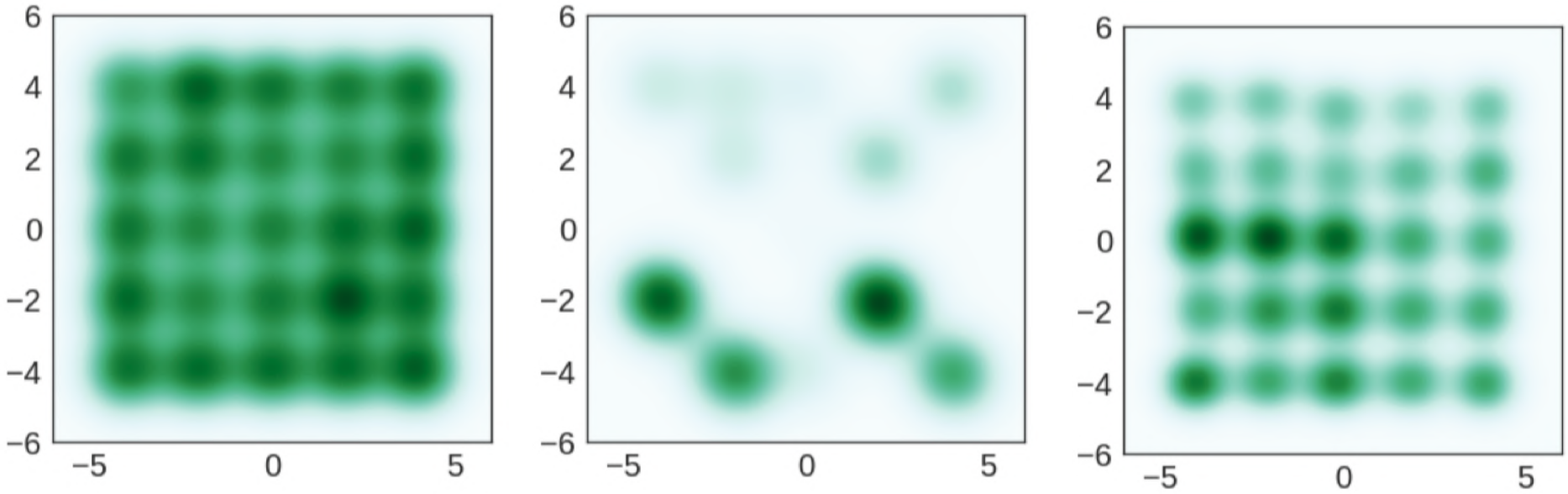




With regularizer
(gradient penalties)

[Mescheder et al 2018]

Challenge: Mode Collapse



Data GAN VEEGAN

[Srivastava et al 2017]

Challenges in training GANs

- Data memorization may be a degenerate solution
- Simultaneous training can lead to oscillations
- May suffer from "mode collapse"

- Much ongoing research on how to train GANs at scale, in a robust manner

Evaluating GANs

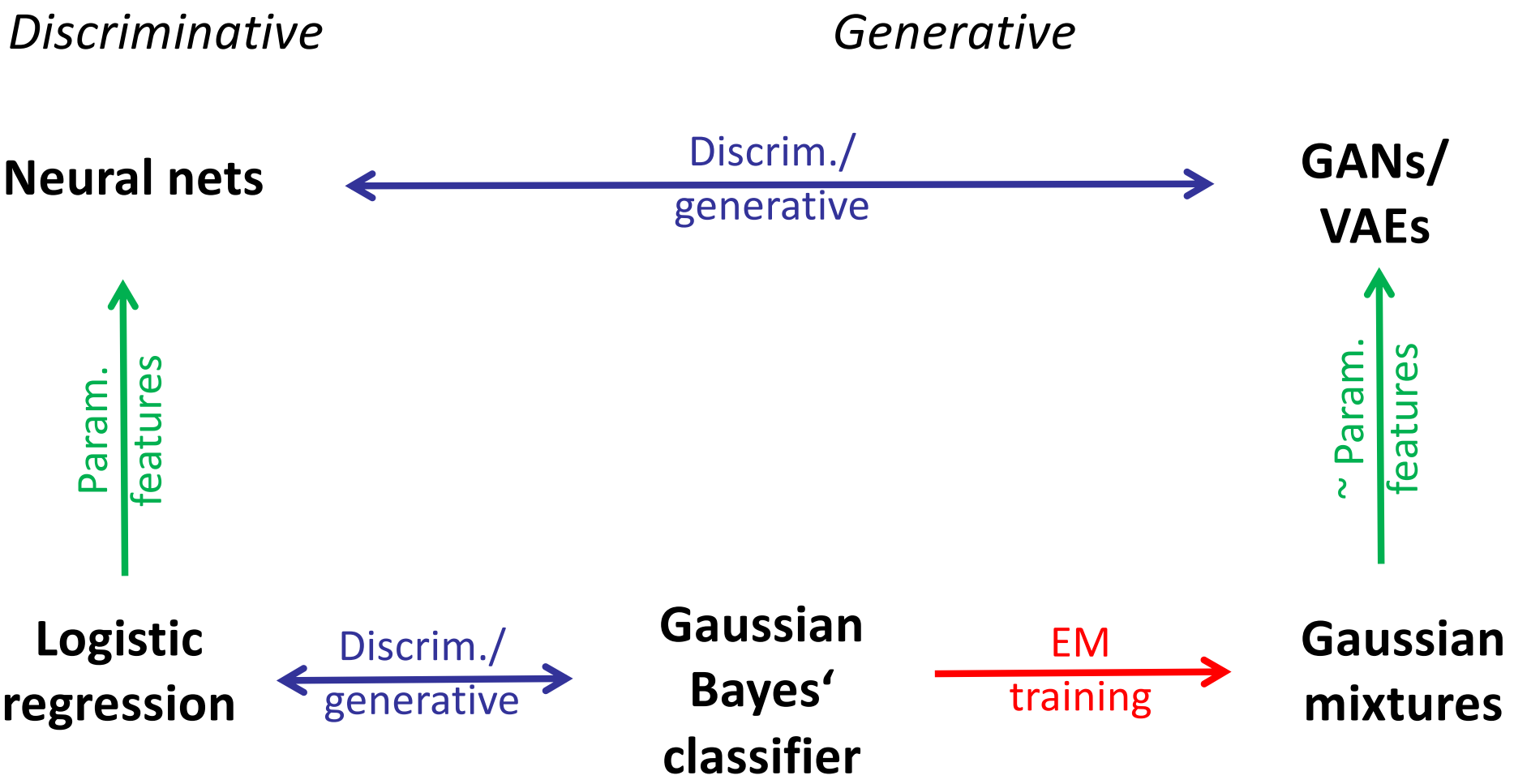
- Cannot compute likelihood on holdout set!
- Generally, difficult problem, with no well-accepted, domain-independent solution
- Various heuristics like Inception score / FID (specifically for images)

- Open area of research

What you need to be able to do

- Understand Gaussian mixture models, and apply the EM algorithm (both variants) for unsupervised and semi-supervised learning of GMMs
  - Initialization, selecting *k* via cross-validation
  - Regularization
- Use GMMs as part of Gaussian-Mixture Bayes Classifier
- GANs use discriminative learning to train generative models!

Generative vs. discriminative modeling

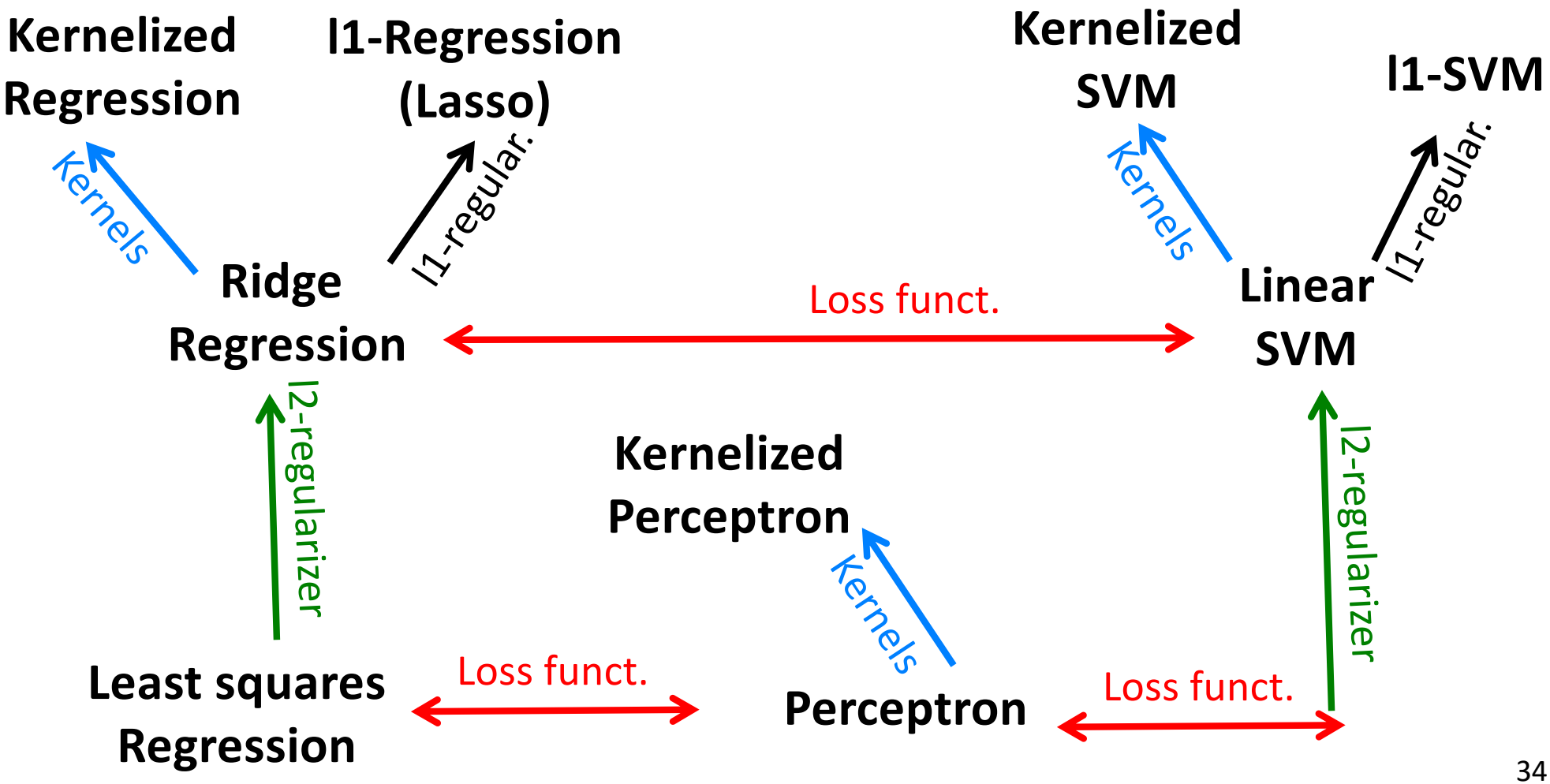


Representation/ features	Linear hypotheses; nonlinear hypotheses with nonlinear feature transforms, kernels, learn nonlinear features via neural nets		
Paradigm:	Discriminative vs. generative		
Probabilistic / Optimization	Likelihood	*	Prior
	Loss-function	+	Regularization
Model:	Squared loss = Gaussian lik., 0/1, Perceptron, Hinge, cost sensitive, multi-class hinge, reconstruction error, logistic loss=Bernoulli lik., cross-entropy loss=Categorical lik. Adversarial losses		L^2 norm (=Gaussian prior), L^1 norm (=Laplace prior), early stopping, dropout Categorical;. Beta/Dirichlet priors
Method:	Exact solution, Gradient Descent, (mini-batch) SGD, Reductions, EM, simultaneous SGD, Bayesian model avg.		
Evaluation metric:	Mean squared error, Accuracy, F1 score, AUC, Confusion matrices, compression performance, log-likelihood on validation set, Inception score, …		
Model selection:	K-fold Cross-Validation, Monte Carlo CV, Bayesian model selection		

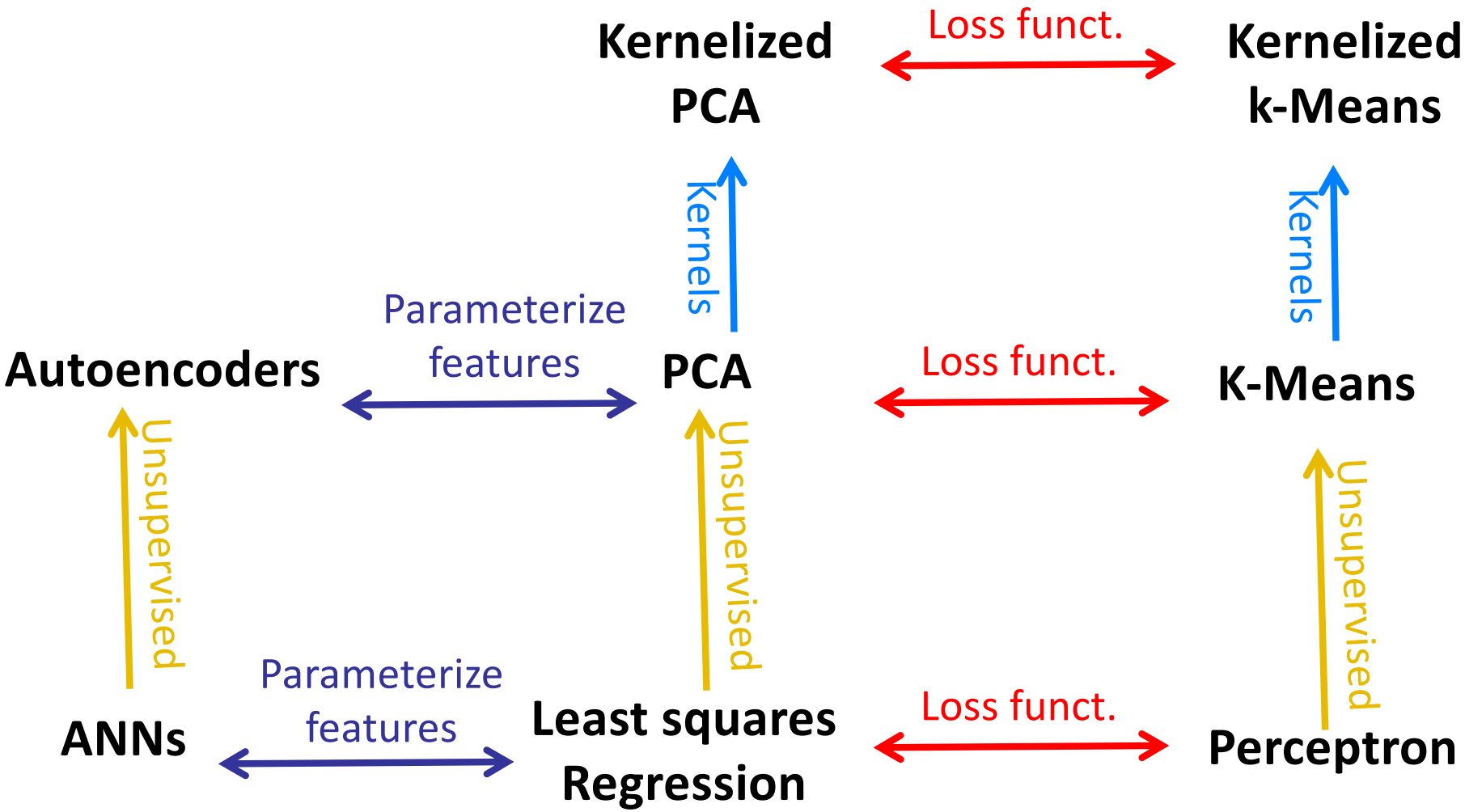
You learned a lot

- Supervised learning:
  - Linear regression, ridge regression, Perceptron, Support Vector Machines, kernelized SVM, kernel ridge regression, k-Nearest Neighbor, l1-SVM, Lasso, logistic regression, neural nets, Gaussian and categorical (Naive) Bayes Classifiers
- Unsupervised learning:
  - k-Means, Gaussian mixtures, semi-supervised GMMs, Principal Component Analysis, Kernel-PCA, neural net autoencoders, Generative Adversarial Networks (GANs)
- Optimization algorithms:
  - (Stochastic) Gradient Descent, EM Algorithm

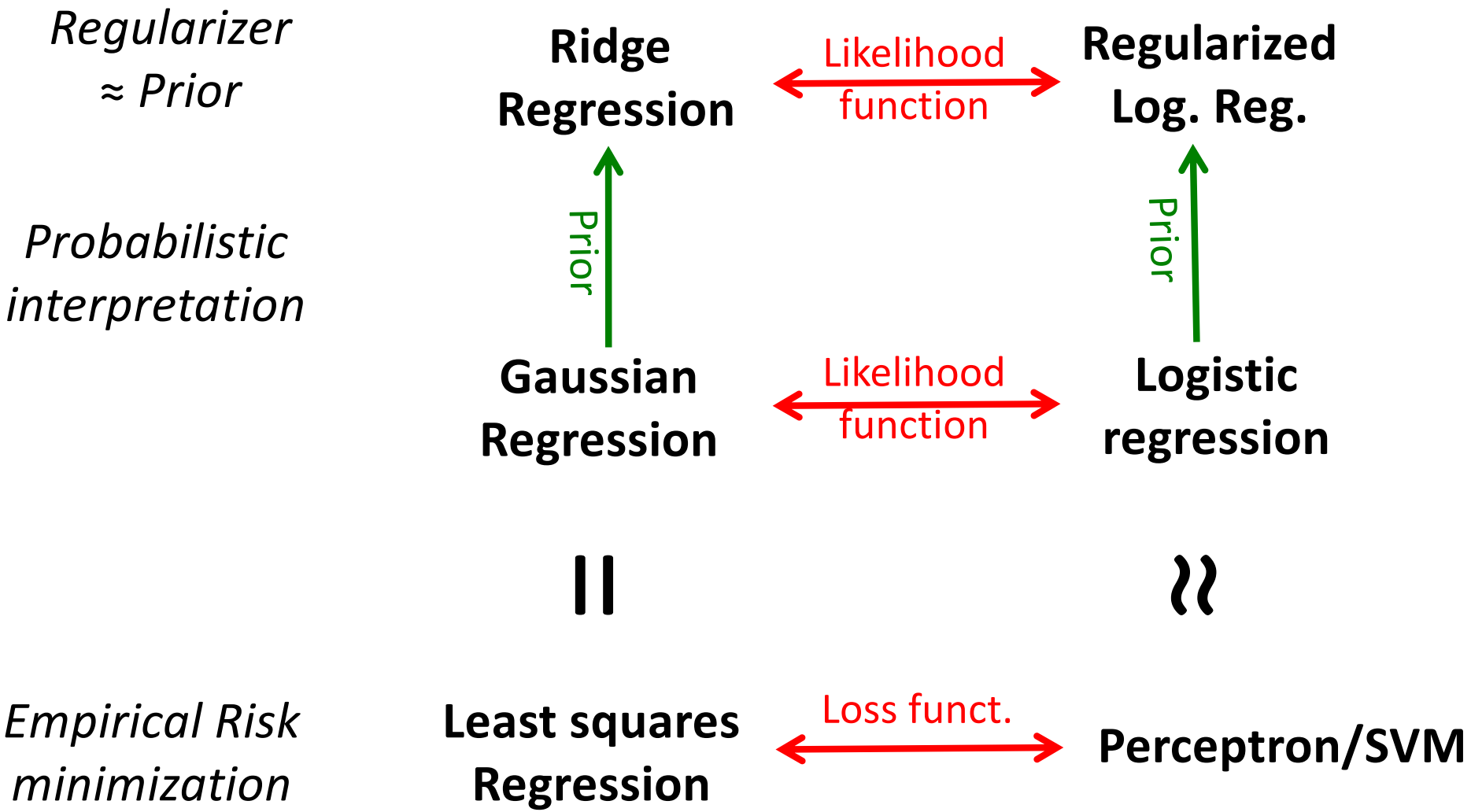
Supervised learning via risk minimization



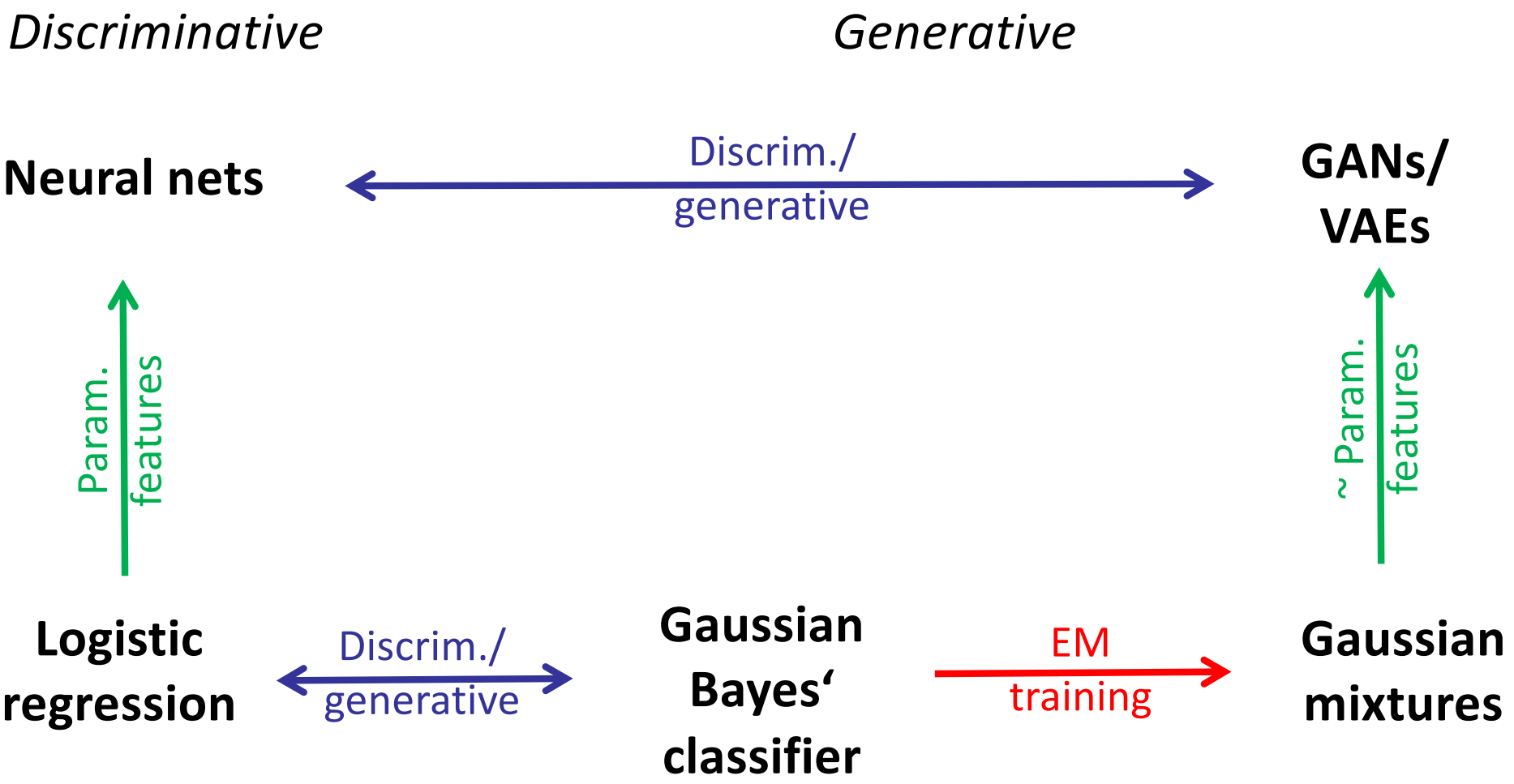
Supervised vs. unsupervised learning



Discriminative probabilistic modeling



Generative vs. discriminative modeling



Key concepts

- Trade goodness of fit and model complexity
- Separate model from algorithm
- Regularizers control overfitting
- Parameter/model selection via cross-validation
- The kernel trick
- Loss = likelihood, regularizer = prior
- Discriminative vs. generative models
- Unsupervised learning = latent variable modeling
  (clustering = classification, dim. Reduction = regression)
- Can train generative models using discriminative learning

Where to learn more

- Other courses
  - Advanced Machine Learning (Fall)
  - Probabilistic Artificial Intelligence (Fall)
  - Deep Learning (Fall)
  - Computational Intelligence Lab (Spring)
  - Statistical Learning Theory (Spring)
- Conference proceedings
  - Machine learning: ICML, NeurIPS, ICLR, AISTATS, …
  - Data Mining: KDD, WWW, WSDM, …
  - AI: AAAI, IJCAI, …
- Bachelor's / Master's thesis ☺