Variational Auto Encoders

Machine Learning WS2021/22

Prof. Gerhard Neumann

KIT, Institut für Anthrophomatik und Robotik

Learning Outcome for today...

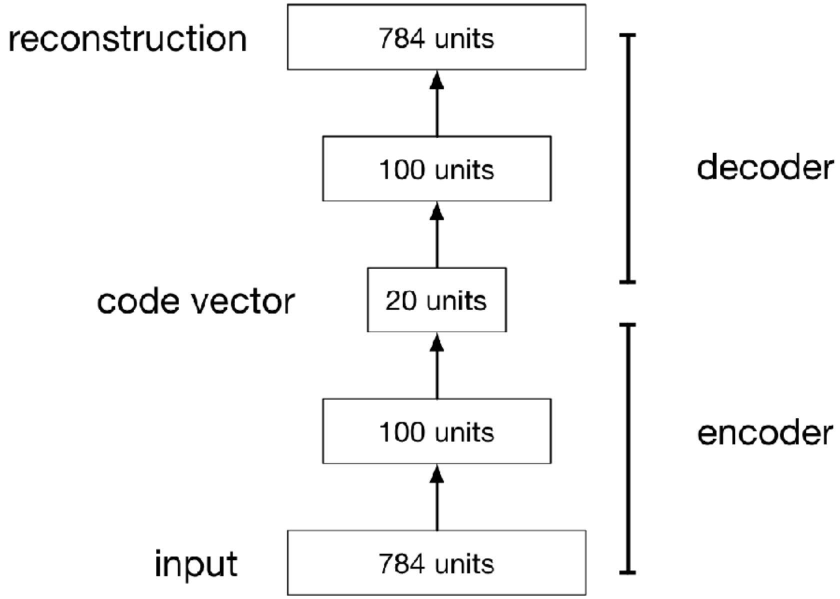
In this lecture, we will..

- Understand the auto-encoders and what you can do with it
- How to use them as generative model and its connection to latent variable models
- What variational auto-encoders are...
- ... and how to train them using Variational Bayes

Auto-encoders

Auto-encoders

- An autoencoder is a feed-forward neural net whose job it is to take an input x and predict x
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input



Minimize reconstruction loss:

$$L(\boldsymbol{\theta}) = \sum_i ||\mathrm{dec}_{\boldsymbol{\theta}}\left(\mathrm{enc}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) - \boldsymbol{x}_i||^2$$

- **Note:** the simplest auto-encoder only has 1 linear layer for the encoder and decoder -> PCA

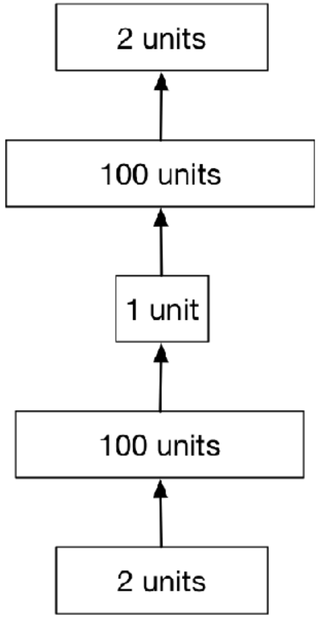
Auto-encoders

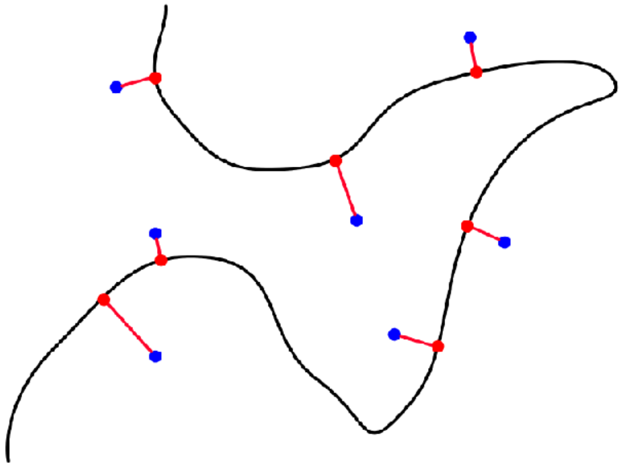
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
  - Note: this requires a VAE, not just an ordinary autoencoder.
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data
- Learn a semantically meaningful representation where you can, e.g., interpolate between different images.

Deep auto-encoders

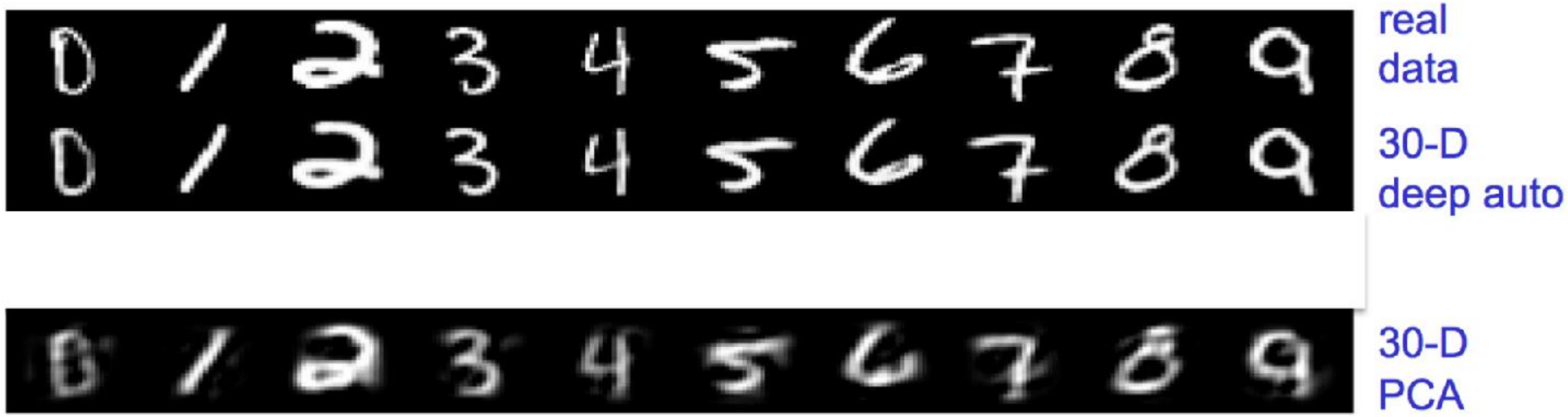
- Deep nonlinear autoencoders learn to project the data, not onto a linear subspace, but onto a nonlinear manifold
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.





Deep auto-encoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



Some limitations of autoencoders

- They’re not generative models, so they don’t define a distribution
- How to choose the latent dimension?

Generative Model

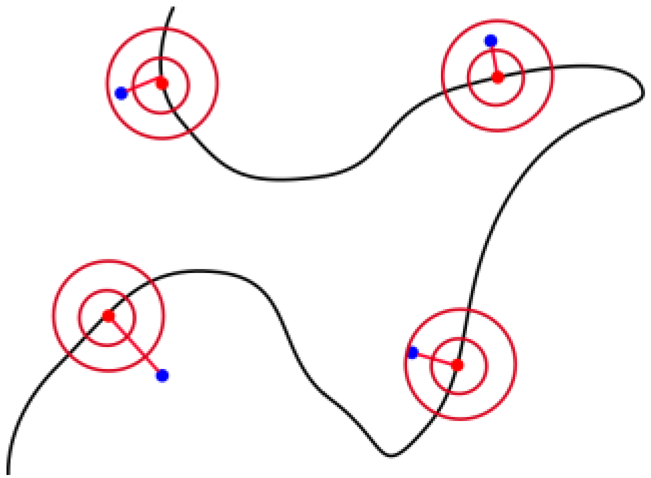
- Consider training a generator network with maximum likelihood.

$$p(\boldsymbol{x}) = \int p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{z}$$

- One problem: if z is low-dimensional and the decoder is deterministic, then p(x) = 0 almost everywhere!
  - The model only generates samples over a low-dimensional sub-manifold of X .
- Solution: define a noisy observation model, e.g.

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \sigma^2 \boldsymbol{I})$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})$ is the function computed by the decoder with parameters θ.



Latent Variable Models and Variational Bayes

Latent Variable Models

Recap: latent variable models

- Examples: mixture models, missing data, latent factors, variational auto-encoders
- Observed variables: **x**, Latent variables: **z**
- Parametric model: $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$
- Marginal distribution: $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$

At least the integral $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$ **is well-defined, but how can we compute it?**

- The decoder function is very complicated, so there's no hope of finding a closed form.

Marginal Likelihood for Latent Variable Models

(Marginal) Log-Likelihood:

$$\mathrm{LogLik}(\mathcal{D}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left(\int p(\boldsymbol{x}_i|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} \right) \approx \sum_{i=1}^{N} \log \left(\frac{1}{M} \sum_{\boldsymbol{z}_j \sim p(\boldsymbol{z})}^{M} p(\boldsymbol{x}_i|\boldsymbol{z}_j) \right)$$

… which is computationally infeasible in most cases

- Requires a lot of samples $\boldsymbol{z}_j$ for each $\boldsymbol{x}_i$ due to uninformed sampling of $\boldsymbol{z}_j$ (high variance in $p(\boldsymbol{x}|\boldsymbol{z})$)

Variational Bayes

Variational Bayes (VB) uses a lower bound of the marginal log-likelihood for the optimization

- For simplicity, lets consider only a single data-point first

$$\underbrace{\log p(\boldsymbol{x})}_{\text{marginal log-like}} = \underbrace{\int q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}}_{\text{Lower Bound } \mathcal{L}(q)} + \underbrace{\int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z}}_{\text{KL Divergence: } \mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x}))}$$

- Where $q(\boldsymbol{z})$ is called the variational / auxiliary distribution
  - This decomposition holds for any $q(\boldsymbol{z})$
  - By introducing q(z), the optimization will become much simpler
- Why is that the same?
  - We can use Bayes rule for $p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})}$ and all terms except $p(\boldsymbol{x})$ cancel

Note

Expectation maximization is a special case of this decomposition

Simplification in EM:

- Posterior $p(\boldsymbol{z}|\boldsymbol{x})$ can be computed in closed form
- **Examples:** Gaussian Mixture Models, Probabilistic PCA

Optimization in Variational Bayes

VB optimizes the lower bound instead of the log-likelihood

$$\mathcal{L}(q,p) = \int q(\boldsymbol{z}) \log \left(p(\boldsymbol{x}|\boldsymbol{z}) \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \right) d\boldsymbol{z} = \int q(\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} - \mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))$$

- Why is it a lower bound? Since $\mathrm{KL}(q||p) \geq 0$ it follows that $\mathcal{L}(q,p) \leq \log p(\boldsymbol{x})$
- Its also called **Evidence lower bound (ELBO)**, as the marginal likelihood is often called evidence

Joint optimization of the lower bound $\mathcal{L}(q,p)$ w.r.t *p* and *q* using stochastic gradient descent

$$q^*, p^* = \arg\max_{q,p} \mathcal{L}(q,p)$$

- In practice, $q_\phi(\boldsymbol{z}), p_\varphi(\boldsymbol{x}|\boldsymbol{z})$ and $p_\varphi(\boldsymbol{z})$ will be parametrized distributions and we optimize over ϕ and φ
- We always improve the lower-bound, but there is no guarantee to improve the marginal likelihood
- Standard for most continuous latent variable models (e.g. Variational Auto-Encoder)
- Lower bound is only tight if $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x}))$ can be set to 0. Thats only true for EM.

Objective for the variational distribution

What does q learn?

$$\mathrm{KL}\big(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})\big) = \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} = \int q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x})q(\boldsymbol{z})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})} d\boldsymbol{z} = \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})} d\boldsymbol{z} + \mathrm{const}$$

$$= -\mathcal{L}(q,p) + \mathrm{const}$$

$$\Rightarrow \mathrm{argmin}_q \, \mathrm{KL}\big(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})\big) = \mathrm{argmax}_q \, \mathcal{L}(q,p)$$

- By maximizing the variational lower bound w.r.t $q(\boldsymbol{z})$, the variational distribution will approximate the posterior, i.e.,

$$q(\boldsymbol{z}) \approx p(\boldsymbol{z}|\boldsymbol{x})$$

Full-Dataset Lower Bound

$$\mathcal{L}(\{q_i\}, p) = \frac{1}{N} \sum_i \int q_i(\boldsymbol{z}) \log p(\boldsymbol{x}_i|\boldsymbol{z}) d\boldsymbol{z} - \mathrm{KL}(q_i(\boldsymbol{z})||p(\boldsymbol{z}))$$

- Introduced individual variational distribution $q_i(\boldsymbol{z})$ for each data point

✓ **More directed sampling:**

- Instead of sampling from the uninformed prior $p(\boldsymbol{z})$ ...
- ... we can now sample from the variational distributions $q_i(\boldsymbol{z}) \approx p(\boldsymbol{z}|\boldsymbol{x}_i)$
- Each $q_i(\boldsymbol{z})$ will produce samples with high $p(\boldsymbol{x}|\boldsymbol{z})$ once optimized!

✓ **Integral is outside the log:**

- only one sample from $q_i(\boldsymbol{z})$ needed to obtain unbiased estimate of the lower bound
- i.e. suitable for stochastic gradient descent while the marginal loglikelihood is not

Special Case 1: Expectation Maximization

$$\underbrace{\log p(\boldsymbol{x})}_{\text{marginal log-like}} = \underbrace{\int q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z})} dz}_{\text{Lower Bound } \mathcal{L}(q)} + \underbrace{\int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} dz}_{\text{KL Divergence: } \mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x}))}$$

Expectation-Maximization uses the same decomposition, but two separate optimization steps

- **Maximization Step:**
  - Keep $q(\boldsymbol{z})$ fixed, maximize Lower bound $\mathcal{L}(q,p)$ w.r.t. model distribution $p_{\boldsymbol{\varphi}}(\boldsymbol{x}|\boldsymbol{z})$ and $p_{\boldsymbol{\varphi}}(\boldsymbol{z})$

$$\boldsymbol{\varphi}^* = \arg\max_{\boldsymbol{\varphi}} \mathcal{L}(q, p_{\boldsymbol{\varphi}})$$

- **Expectation Step:**
  - Keep model $p_{\boldsymbol{\varphi}}(\boldsymbol{x}|\boldsymbol{z})$ and $p_{\boldsymbol{\varphi}}(\boldsymbol{z})$ fixed, minimize KL w.r.t q

$$q^* = \arg\min_q \mathrm{KL}\big(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})\big)$$

$$\rightarrow q^*(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$$

Sidenote

- In EM the KL can be set to zero (i.e. we can compute posterior $p_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$)
- Only works in special cases
  - e.g. discrete z, GMMs
- In this case the lower bound is tight
- increasing lower bound always increases marginal log-like

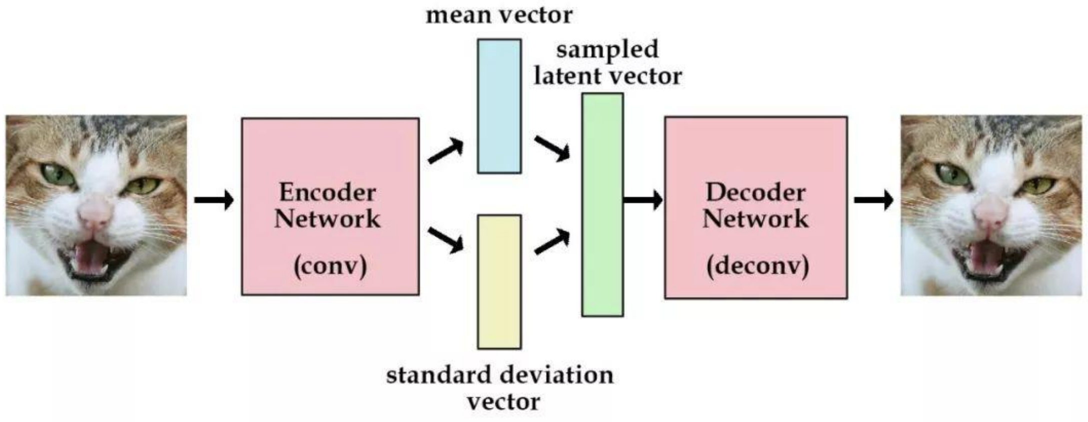
Special Case 2: Amortized Variational Inference

Instead of using an individual auxiliary distribution $q_i(\boldsymbol{z})$ **per data-point** $\boldsymbol{x}_i$, we can use an "amortized" distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)$ that is given by a DNN

$$\mathcal{L}(q,p) = \frac{1}{N}\sum_i \int q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)\log p_{\boldsymbol{\varphi}}(\boldsymbol{x}_i|\boldsymbol{z})d\boldsymbol{z} - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\varphi}}(\boldsymbol{z}))$$

This is the standard objective used for variational auto-encoders (VAE)

- Encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
- Decoder $p_{\boldsymbol{\varphi}}(\boldsymbol{x}|\boldsymbol{z})$
- Latent Prior $p_{\boldsymbol{\varphi}}(\boldsymbol{z})$



Optimization over the variational distribution

$$\mathcal{L}(q,p) = \frac{1}{N}\sum_i \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i) \log p_{\boldsymbol{\varphi}}(\boldsymbol{x}_i|\boldsymbol{z}) d\boldsymbol{z} - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\varphi}}(\boldsymbol{z}))$$

But: How can we optimize over the sampling distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$?

- Different to standard max-likelihood: Here, samples are not fixed but generated!
- Standard gradients can be used (related to policy gradients), but very inefficient as it does not use gradient information of $\partial \log p_{\boldsymbol{\varphi}}(\boldsymbol{x}|\boldsymbol{z})/\partial \boldsymbol{z}$

- We need something more efficient: **Reparemetrization trick**!

Basics: Reparameterization Trick

We want to optimize distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ using the following expected objective

$$\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}} \; \mathbb{E}_{p_{\boldsymbol{\theta}}}\left[f(\boldsymbol{x})\right] = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x}$$

... and we are given $\frac{df}{d\boldsymbol{x}}(\boldsymbol{x})$. How can we exploit this information?

We can reparametrize the expectation:

- Introduce random variable $\boldsymbol{\xi} \sim q(\boldsymbol{\xi})$ where q is a simple, parameter-free distribution (e.g., $q(\boldsymbol{\xi}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$)
- If we can find a mapping $\boldsymbol{x} = \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})$ such that ***x*** is distributed as $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ then:

$$\int p_{\boldsymbol{\theta}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = \int q(\boldsymbol{\xi}) f(\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})) d\boldsymbol{\xi}$$

We moved the parameters from distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ into a function $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})$

Basics: Reparameterization Trick

Example: Lets assume $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $q(\boldsymbol{\xi}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. If we set

$$\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi}) = \boldsymbol{\mu} + \boldsymbol{A}^T\boldsymbol{\xi}, \quad \text{with } \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \text{ and } \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{\Sigma}$$

Then $\boldsymbol{x}' = \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})$ is distributed with $p(\boldsymbol{x}') = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{A}^T\boldsymbol{I}\boldsymbol{A}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Reparametrization Trick:

$$\int p_{\boldsymbol{\theta}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = \int q(\boldsymbol{\xi}) f(\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})) d\boldsymbol{\xi}$$

Reparametrized Gradient:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}}}[f(\boldsymbol{x})] = \nabla_{\boldsymbol{\theta}} \int q(\boldsymbol{\xi}) f(\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})) d\boldsymbol{\xi} = \int q(\boldsymbol{\xi}) \frac{\partial \boldsymbol{h}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}(\boldsymbol{\xi}) \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{\xi})) d\boldsymbol{\xi}$$

- We can now use the gradient $\frac{\partial f}{\partial \boldsymbol{x}}$ to compute $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}}}[f(\boldsymbol{x})]$!

Back to optimization over the variational distribution

- Lower bound: $$\mathcal{L}(q,p) = \frac{1}{N}\sum_i \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)\log p_{\boldsymbol{\varphi}}(\boldsymbol{x}_i|\boldsymbol{z})d\boldsymbol{z} - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\varphi}}(\boldsymbol{z}))$$
- Distribution: $$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \sigma^2_{\boldsymbol{\phi}}(\boldsymbol{x})\boldsymbol{I}\right)$$
- Reparametrization function: $$\boldsymbol{h}_{\boldsymbol{\phi}}(\boldsymbol{\xi},\boldsymbol{x}) = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \circ \boldsymbol{\xi}$$
- Reparametrized lower bound:

$$\mathcal{L}(q,p) = \frac{1}{N}\sum_i \int p(\boldsymbol{\xi})\Big(\underbrace{\log p_{\boldsymbol{\varphi}}\big(\boldsymbol{x}_i|\boldsymbol{h}(\boldsymbol{\xi},\boldsymbol{x})\big)}_{\text{reconstruction}} + \underbrace{\log p_{\boldsymbol{\varphi}}\big(\boldsymbol{h}(\boldsymbol{\xi},\boldsymbol{x})\big) - \log q_{\boldsymbol{\phi}}\big(\boldsymbol{h}(\boldsymbol{\xi},\boldsymbol{x})|\boldsymbol{x}\big)}_{\text{KL-term}}\Big)$$

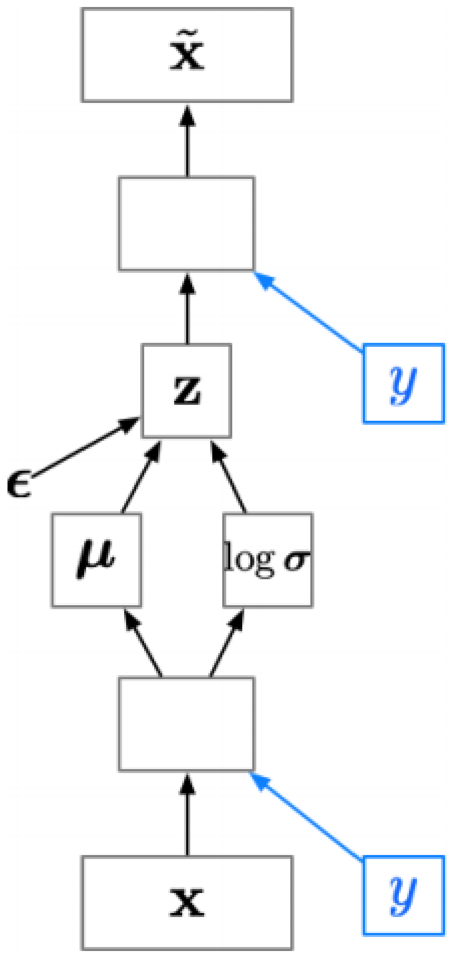
Variational Auto-Encoders

Faces produced by variational auto-encoders

- In comparison to other generative models (such as GANs), VAEs produce rather blurry images
  - More specific methods (e.g. hierarchical VAEs) can achieve similar performance to GANs
  - Most likely cause: Maximum Likelihood objective of VAEs
- In short, a VAE is like an autoencoder, except that it's also a generative model
  - defines a distribution p(x)

Class conditional VAEs

- So far, we haven't used the labels y. A class-conditional VAE provides the labels to both the encoder and the decoder.
- Since the latent code **z** no longer has to model the image category, it can focus on modeling the stylistic features.
- If we're lucky, this lets us disentangle style and content. (Note: disentanglement is still a dark art.)
  - See Kingma et al., "Semi-supervised learning with deep generative models."

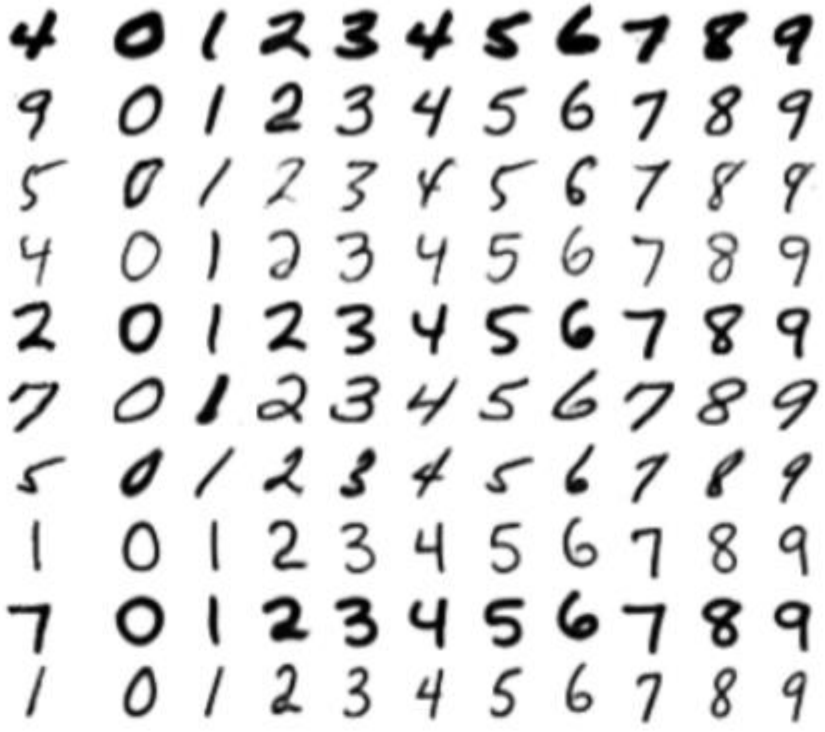


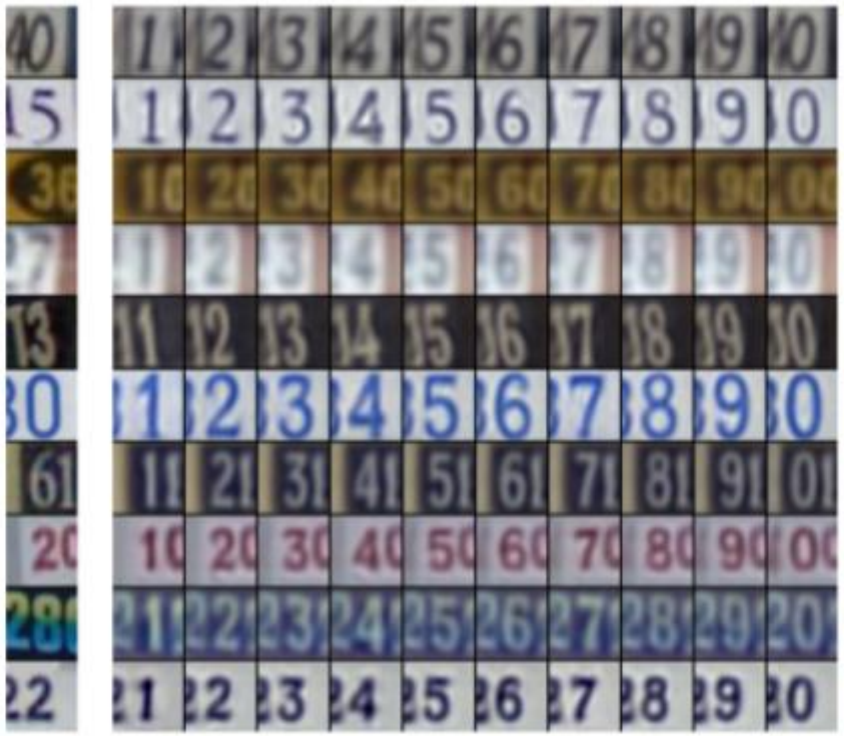
Class-conditional VAE

- By varying two latent dimensions (i.e. dimensions of z) while holding y fixed, we can visualize the latent space.

Class-conditional VAE

- By varying the label y while holding **z** fixed, we can solve image analogies





Latent Space Interpolations

- You can often get interesting results by interpolating between two vectors in the latent space:

Ha and Eck, "A neural representation of sketch drawings"

Wrap-Up Variational Bayes

VB is a method for optimizing the data likelihood of latent variable models

- It introduces a variational distribution $q_i(\boldsymbol{z})$ over the latent variable
  - Variational distribution should approximate posterior $q_i(\boldsymbol{z}) \approx p(\boldsymbol{z}|\boldsymbol{x}_i)$
- ... and decomposes the marginal likelihood in a lower bound and a KL-term
- **The lower bound is in general easier to optimize than the marginal log-likelihood:**
  - ✓ The integral has moved outside the log
  - ✓ More direct sampling in latent space by sampling from approximate posterior $q_i(\boldsymbol{z})$ instead of prior $p(\boldsymbol{z})$
  - × No guarantee that we also improve marginal loglikelihood (except in the special case of EM)
- Expectation Maximization is a special case where posterior can be computed analytically
- Most prominent application of VB is the Variational Auto-Encoder

Intermediate Lecture Wrap-Up – Algorithms

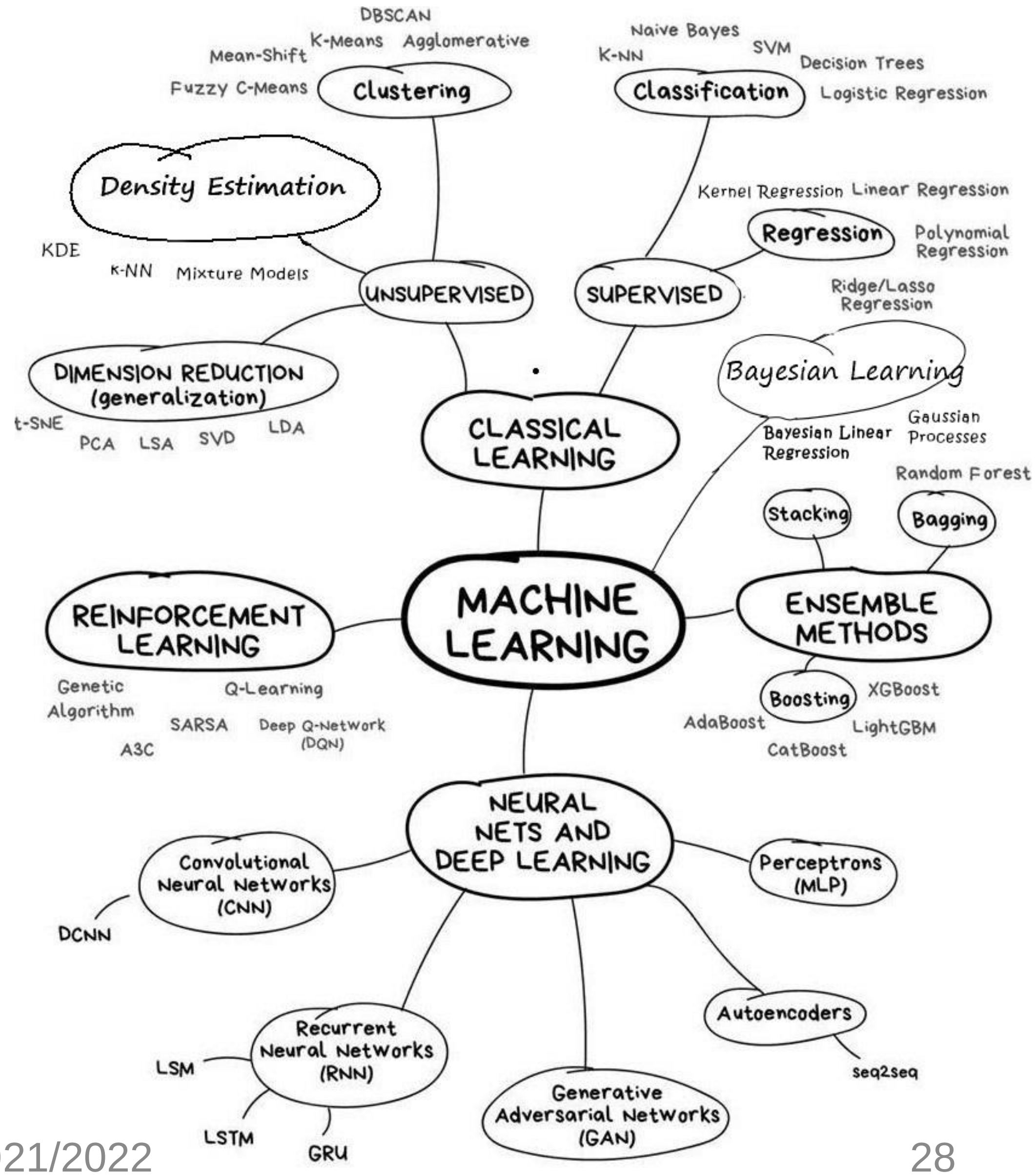
Chapter 1: Classical Supervised Learning
- ✓ Linear Regression,
- ✓ Ridge Regression,
- ✓ k-NN,
- ✓ Trees and Forests

Chapter 2: Kernel Methods
- ✓ Kernel-Regression
- ✓ Support Vector Machines

Chapter 3: Bayesian Learning
- ✓ Bayesian Linear Regression
- ✓ Gaussian Processes

Chapter 4: Neural Networks
- ✓ Backpropagation
- ✓ MLPs, CNNs, LSTMs

Chapter 5: Unsupervised Learning
- ✓ PCA
- ✓ K-means
- ✓ Expectation-Maximization
- ✓ Variational Auto Encoders



Intermediate Lecture Wrap-Up – Basics

Chapter 1: Classical Supervised Learning

- ✓ Matrix/Vector Calculus
- ✓ Probability Theory, Maximum Likelihood
- ✓ Gradient Descent

Chapter 2: Kernel Methods

- ✓ Sub-gradients
- ✓ Constraint Optimization

Chapter 3: Bayesian Learning

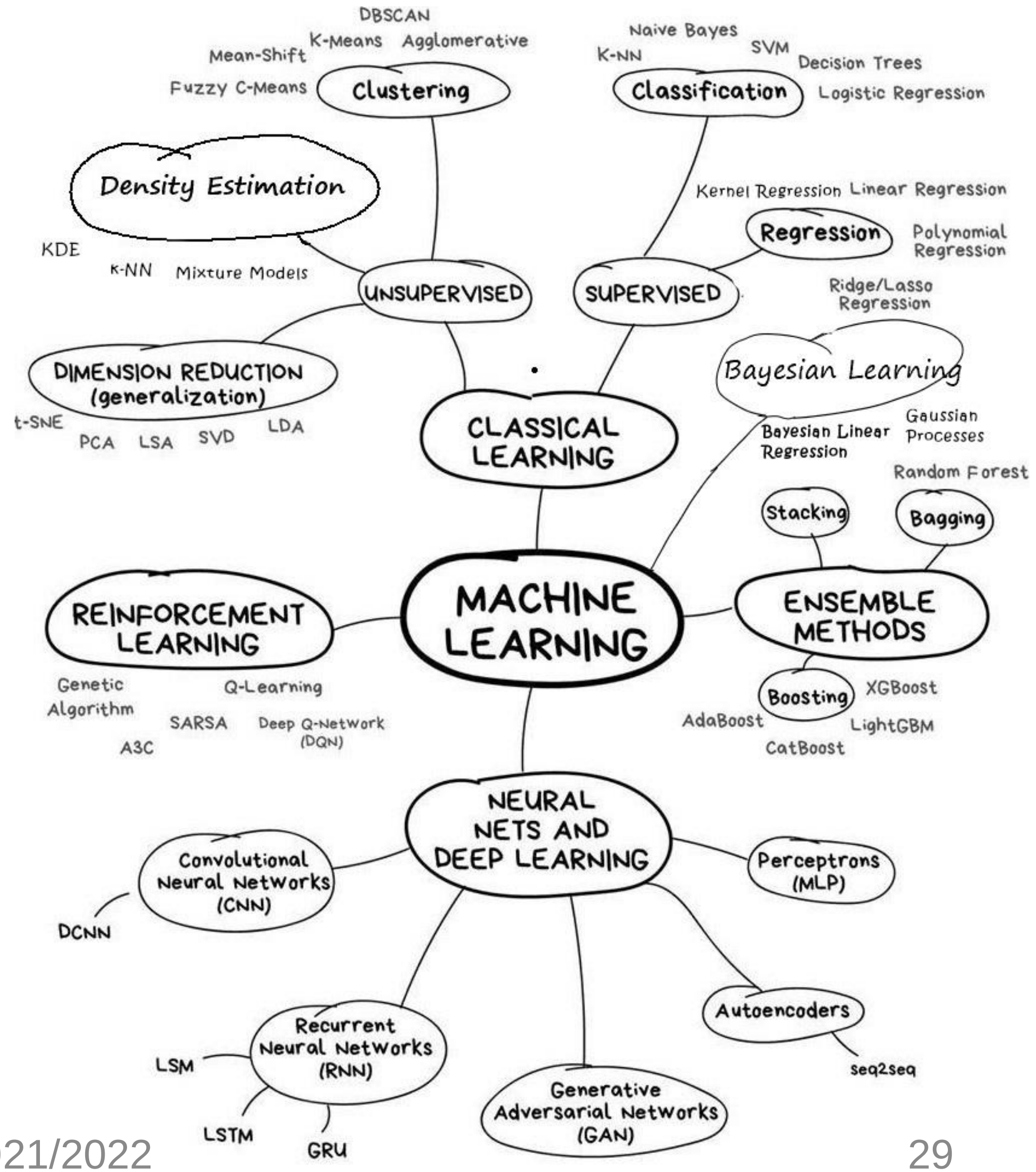
- ✓ “Completing the Square”
- ✓ Gaussian Conditioning

Chapter 4: Neural Networks

- ✓ Multivariate chain rule

Chapter 5: Unsupervised Learning

- ✓ KL-divergences
- ✓ Reparametrization trick



The ML algorithm “coordinate system”

Most ML algorithms can be grouped along 3 axis:

- **Representation:** What is the underlying representation of our model?
- **Loss function:** How do we define what is a good and what is a poor model?
- **Optimization:** How do we optimize?

... of course more axis exists, e.g. Regularization

Intermediate Lecture Wrap-Up – Representations

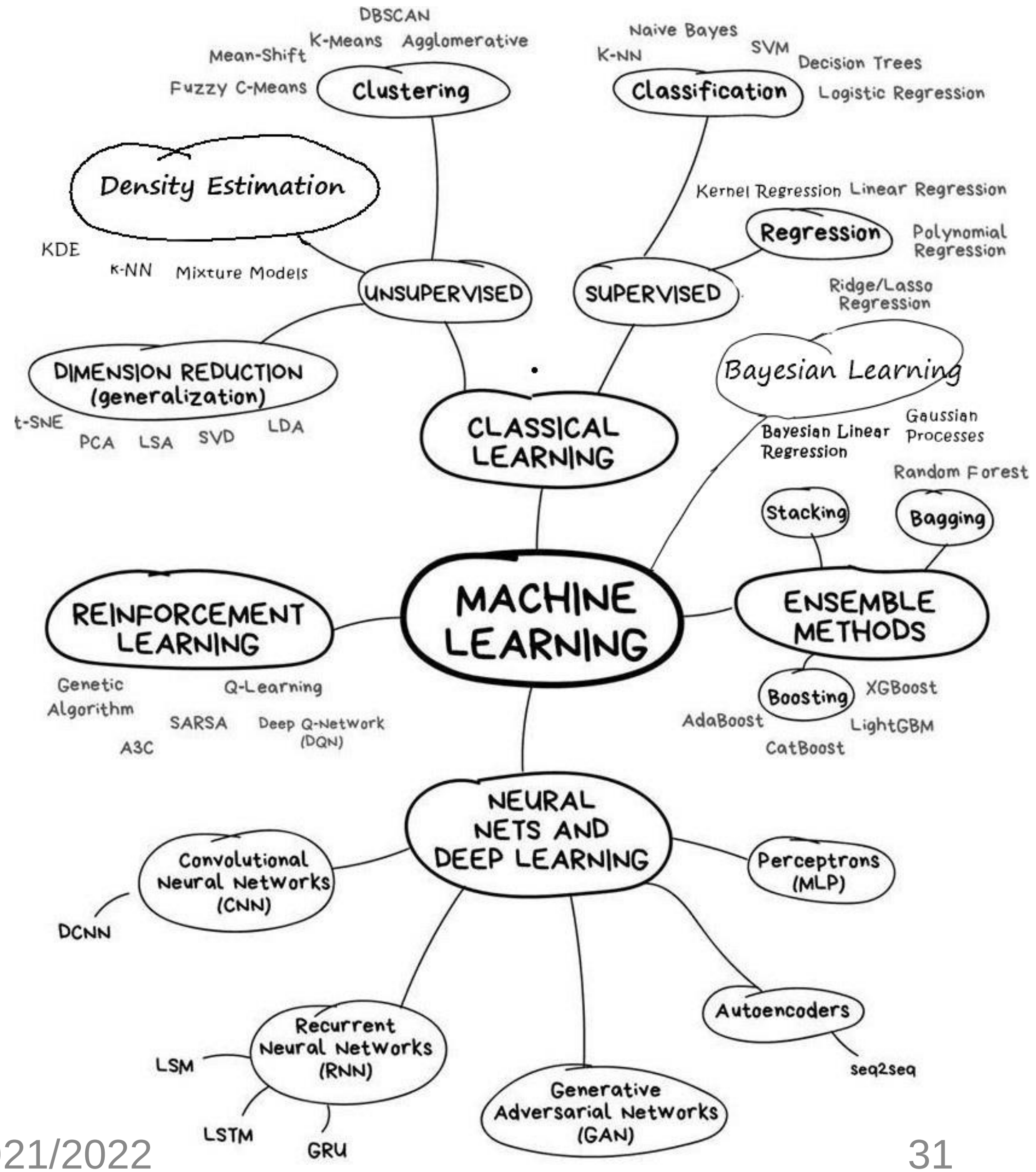
Chapter 1: Classical Supervised Learning
- ✓ Features / Basis Functions: Linear (Ridge) Regression, Logistic Regression
- ✓ Instances: k-NN
- ✓ Trees: CART
- ✓ Ensembles: Forests

Chapter 2: Kernel Methods
- ✓ Kernels: SVM and Kernel Regression

Chapter 3: Bayesian Learning
- ✓ Features: Bayesian Linear Regression
- ✓ Kernels: Gaussian Processes

Chapter 4: Neural Networks
- ✓ Feed-forward Neural Networks, CNNs, Recurrent Neural Networks, LSTMs, GRU: Backprob

Chapter 5: Unsupervised Learning
- ✓ Cluster centroids: k-means
- ✓ Linear subspaces: PCA
- ✓ Mixture Models: Expectation Maximization
- ✓ (Variational) Auto-encoders: Variational Bayes



Intermediate Lecture Wrap-Up – Loss Functions

Chapter 1: Classical Supervised Learning

- ✓ Mean/Summed Squared error (SSE): Linear Regression
- ✓ Gaussian Log-Likelihood: Probabilistic linear Regression
- ✓ Binary Cross Entropy Likelihood: Logistic Regression
- ✓ Soft-Max Likelihood: Multi-class classification

Chapter 2: Kernel Methods

- ✓ SSE: Kernel Regression
- ✓ Maximum Margin or Hinge Loss: SVM

Chapter 3: Bayesian Learning

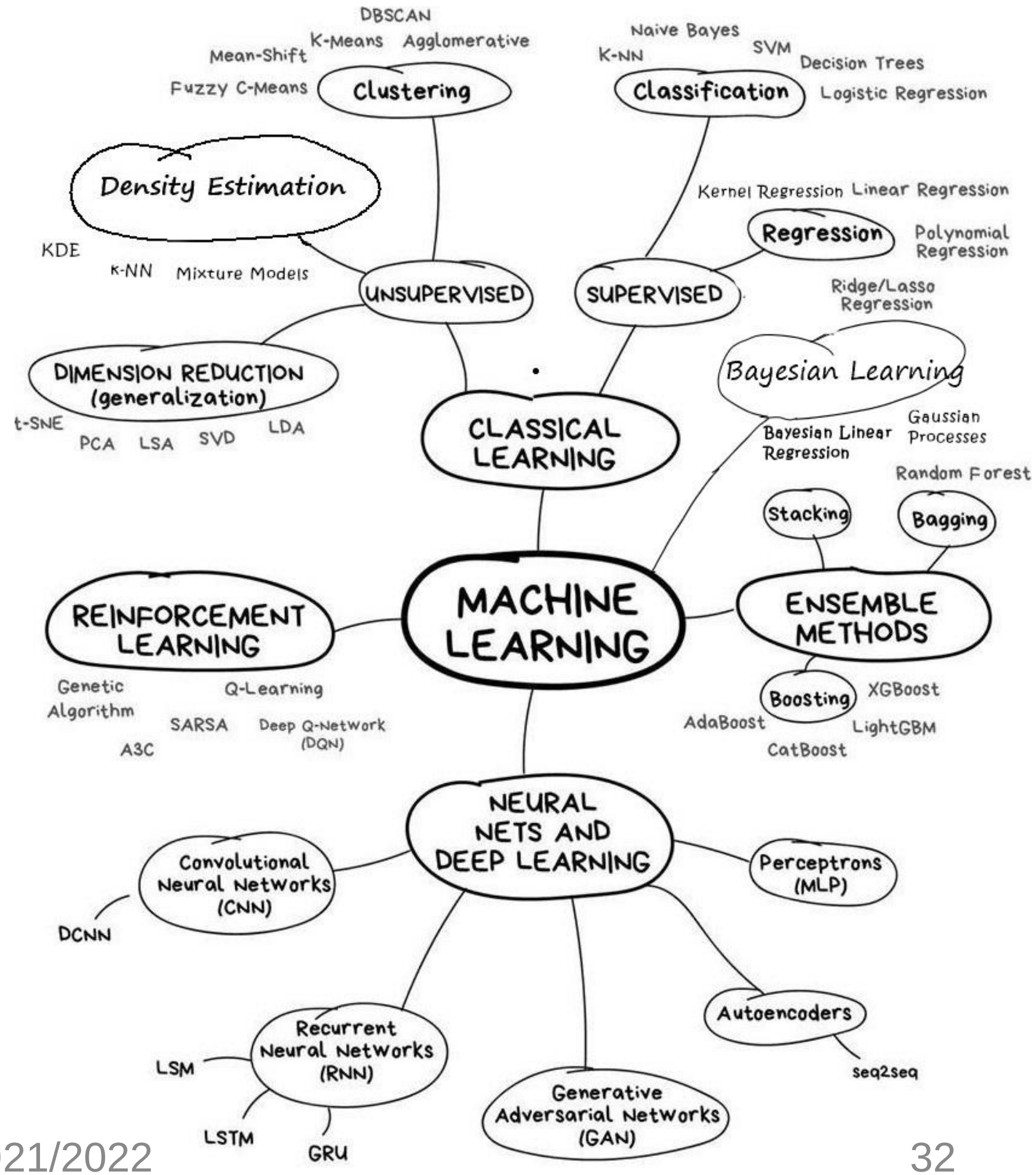
- ✓ Maximum a-posteriori solution: Probabilistic ridge regression

Chapter 4: Neural Networks

- Most of that above...

Chapter 5: Unsupervised Learning

- ✓ Reconstruction Loss: PCA, k-means
- ✓ Marginal Log-likelihoods: EM
- ✓ Evidence Lower Bound (ELBO): Variational Bayes



Intermediate Lecture Wrap-Up – Optimization Methods

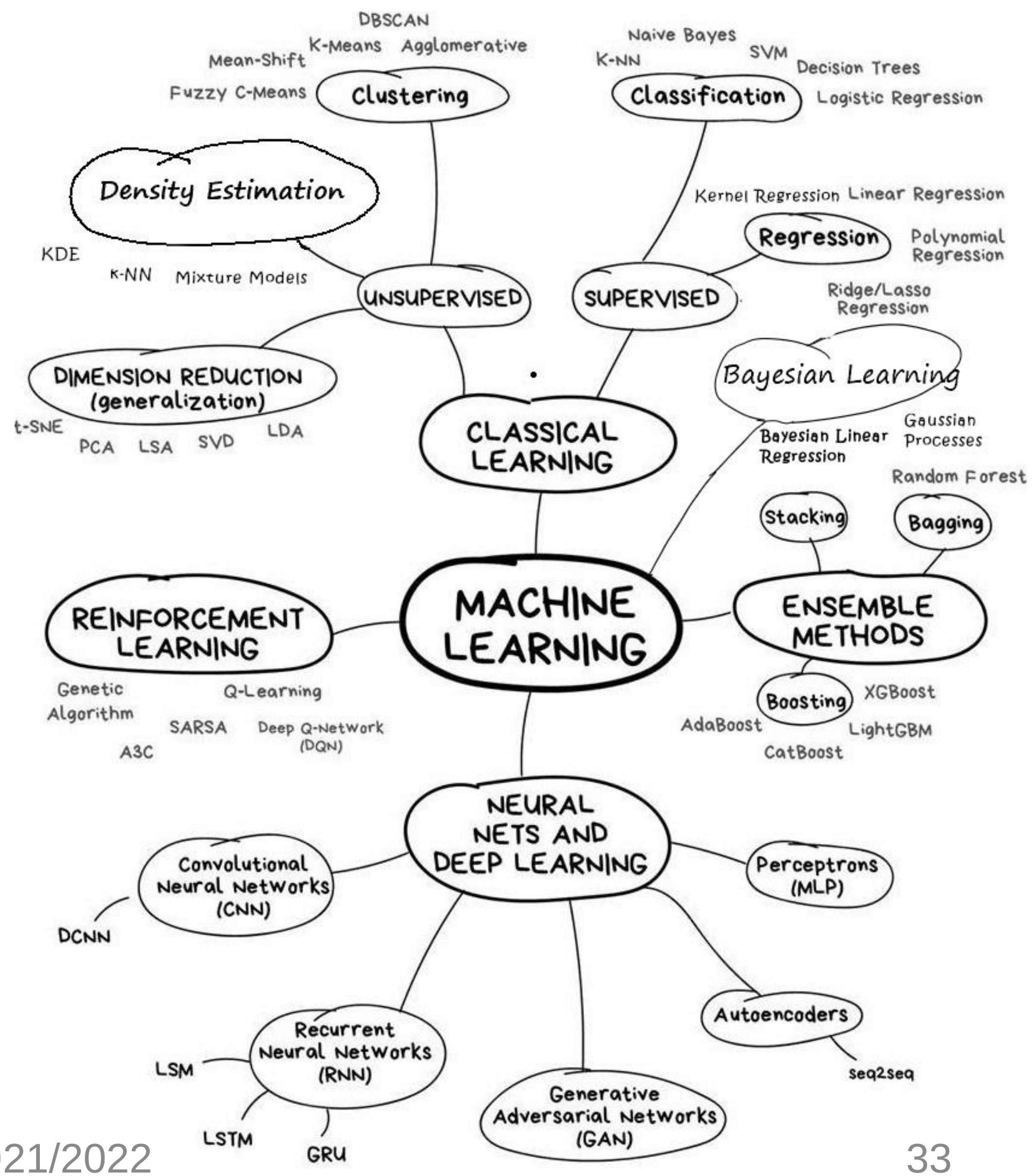
Chapter 1: Classical Supervised Learning
- ✓ Least Squares Solution: Linear Regression
- ✓ Gradient Descent: Logistic Regression

Chapter 2: Kernel Methods
- ✓ Least Squares Solution: Kernel Regression
- ✓ Sub-Gradients: SVM
- ✓ Lagrangian Optimization: SVMs

Chapter 3: Bayesian Learning
- ✓ Posterior approximation

Chapter 4: Neural Networks
- ✓ More specialized gradient descent methods
- ✓ Adam, 2nd order methods

Chapter 5: Unsupervised Learning
- ✓ Expectation-Maximization
- ✓ Variational Bayes



Where to go from now?

Other ML lectures:

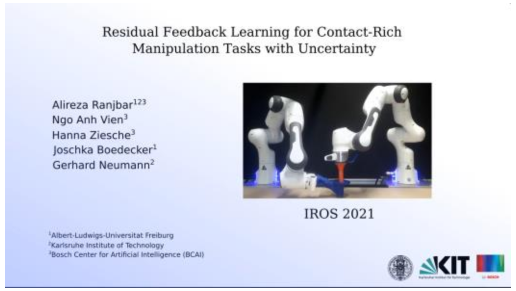
- WS: Reinforcement Learning, **Me**
- SS: Deep Learning and Neural Networks: Prof. Waibel
- SS: Deep Learning for Computer Vision: Prof. Stiefelhagen
- WS: Optimization Methods for Machine Learning and Engineering
- SS: Cognitive Systems, Prof. Waibel and **Me**
- SS: Pattern Recognition, Prof. Beyerer
- SS: Maschinelles Lernen in den Materialwissenschaften, Prof. Friedrich
- SS: Maschinelles Lernen in der Computersicherheit, Prof. Wressnegger

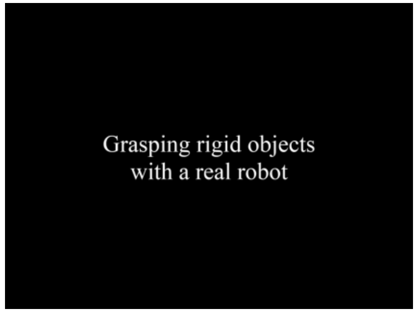
A bit of self-advertisment

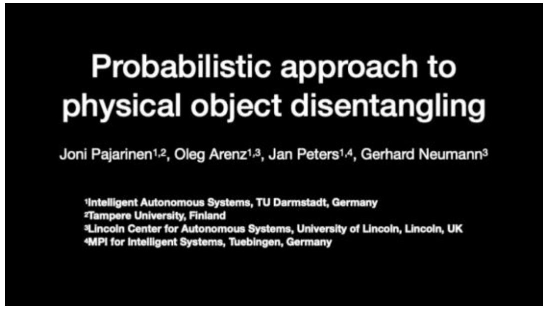
Autonomous Learning Robots Lab

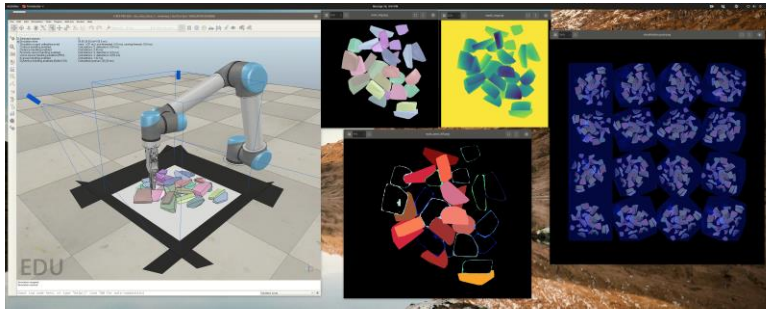
What else do we offer?

- **Hot research topics:** Robot Reinforcement Learning, Deep Learning, Imitation Learning, Robotics, Human-Robot Collaboration, Variational Inference
- **Projektpraktikum and Seminar**
  - Work on your own research topic together with your supervisor
  - Get to know latest state of the art algorithms
  - Get experience in doing top-nodge research
- **Praxis der Forschung**
  - 2 semester, 24 ECTS intensive research project
- Interested in a **Master-Thesis or Bachelor Thesis**?
  - Have a look at https://alr.anthropomatik.kit.edu/
  - Use real robots (Franka Panda arms)
  - High success-rate of turning your thesis into a paper!
- **Hiwi Positions:**
  - Use robots, cameras, physics simulation, benchmark algorithms etc...









The end

Announcement: Fragestunde, 18.02 16:00