Talk 1 - Y. Bengio - Deep learning I

28/08/17

1. Backprop
2. DNNs
3. RNNs

} Broad intro
Colegeden afterwards

- Attention mechanisms → liberate NNs from fixed-size input
  - 
  - "there is a dog"
  - network has attention at certain spots of image while speaking.

- Powerful priors ⇒ Can beat curse of dimensionality
  - ↳ Mimic how brain works with heuristics
  - ↳ Compositionality assumption: Exponential gain in representational power
    - ① ⇒ distr. representations
    - ② ⇒ deep architecture (layers of features) → sequential
    - → Assumption: Compositions are able to grasp underlying phenomenons in the world!

  ① logical partitions vs. distr. partitions
  ② World is compositional

- Attention ⇒ gating units → control info flow!
  - ↳ E.g. Neural translation
  - ↳ System I: Cortex → fast ⇒ intuitive
    System II: Hippocampus → slow ⇒ logical
    } RNN acts as a controller to read and write

- SGD → extremely simple method
  - ↳ But: extremely powerful technique!

- Problems
  - → Current models cheat by picking on some regularities → superficial clarity
  - → Not able to capture long-term dependencies → long-scale dependencies
  - → Heavily relies on smooth differential predictors

- Autoencoders
  - → Mapping of low & high-levels of abstraction
  - → distortion of space → deform complex distr. into space where it is simple to do computations → manifolds disentangle
  - ↳ O-shot learning: learn without data → prob. mapping between representation spaces and embedded points → classification based on similarity in different embedded spaces

- Challenges
  - → Unsupervised learning
  - → Meaningful features ⇏ causal factors of the world
  - → Connection to RL → reason about rare events
    - Co extend framework to causal reasoning
  - → Hardware: specialized chips/GPUs for NNs
  - → Representation learning: "Good" representation
    - ⇏ indep. controllable of the world
    - ↳ discovery by acting in the world

Talk 2 – S. Meyn – RL

28/08/17

- RL & Stoch. Approx.
  - → Stoch. Approx.: $\bar{f}(\theta^*) := \mathbb{E}[f(\theta, W)]\big|_{\theta=\theta^*} = 0$
    - $\mathbb{E}$: exp. ⇒ complex integral
    - $f(\theta, W)$: potentially complex / unknown
  - → Monte Carlo Average Estimation: $\theta(n) = \frac{1}{n}\sum_{i=1}^{n} c(X(i))$ ⇒ ergodicity ⇒ recursion available
  - → MDPs: $P\{X(t+1) \in A \mid X(t), U(t) = u, \text{prior hist.}\}$
    - $U(t)$: decision u
    - to $h^*(x) = \min_u \sum_{t=0}^{\infty} \beta^t \mathbb{E}[c(X(t), U(t)) \mid X(0) = x]$
      - β: discount
      - c: cost
    - to satisfies fixed-point eq. → Galerkin relaxation available
    - to Q-learning: fixed point eq. for Q fct. [Watkins] ⇒ with avoidance of expectation → stochastic opt.
    - to Markovianity does not have to hold in reality: e.g. stocks!
    - to In practice: slow and unstable
  - → Recursion: $\theta(n+1) = \theta(n) + a_n f(\theta(n), X(n))$
    - a_n: gain – crucial!
    - Euler approx. analogy
    - ODE: $\frac{d}{dt}\vartheta(t) = \bar{f}(\vartheta(t)) \to \theta^*$
    - ⊕ Ulice
  - → Borkar's Monograph ⇒ best reference
    - ⊕ understanding relationship!

- Performance Eval.
  - → Finite-n bound ⇒ concentration!
  - → Asymptotic Variance → not really applicable to anything
  - ↳ MC Approx. ⇒ No concentration result; beautiful asymptotics
  - ↳ Lyapunov Eq. ⇒ not used even though it is stronger than bound!
- How to model the gain / learning rate?
  - → problem with bad emergence
  - ⇒ Least squares TD
  - → inversely proportional to optimal variance
    - → Polyak-Ruppert
    - → Stochastic Newton Raphson

$$\Sigma^* = A^{-1} \Sigma_\Delta A^{-1\top}$$

- Zoom @ learning
  - → $Q^\theta \leftrightarrow C^\theta$ ⇒ ODE: Newton-Raphson with nonlinearities!

$$C^\theta \to C^* \iff Q^\theta \to Q^* \to 1\text{-to-1 mapping!}$$

- RL Open Qs // Problems
  - → caused by dimensionality / variance
  - → finite concentration results → to be used to drive algorithm design!
  - → context-specific algo design
- Trade-off between α in $\theta \in \mathbb{R}^d$ and concentration under its covariance matrix.

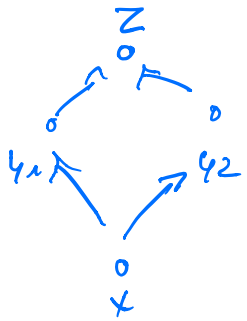
Talk 3 - Y. Bengio — Deep Learning II 28/08/17

① Backprop and graph flow

- Objective function is not underlying truth (=> data-dep & Regularizer)
  → Don't have to find exact minimum! => Approx. opt.
- Gradient => best direction to go to → look at proof!
  → Forward prop: $O(n)$ => where n: # params
  → Backprop: $O(n)$
- Alternative: Finite differences → takes $O(n^2)$

$$\frac{\partial L(\theta_i, \theta_{-i})}{\partial \theta_i} = \frac{L(\theta_i + \varepsilon, \theta_{-i}) - L(\theta_i, \theta_{-i})}{\varepsilon}$$

- Backward accumulation: Only gradients computation method
  → Chain rule: Consider intermediate values
  to learn how to change parameters by learning how to change intermediate values!



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{y_2}\frac{\partial y_2}{\partial x}$$

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{u} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

- Loss fct.: negative log LH $-\log p(y|x=x)$ → cross-entropy!
- ∇_x interesting if we want to measure sensitivity/robustness!

- Very general framework! → Just have to define forward prop ⇒ implementations define the inverse gradients needed!
- Automatic differentiation → gradient comp. can be automatically inferred from symbolic expression of f prop.
- Neural justification
  - → earlier: people were sceptical since forward and backwards computation is very different
  - → new findings suggest that brain is able to use neighbors in both stages
  - → new neural findings! ⇒ makes it more reasonable!
- Propert: Change in graph → stochastic regularization
- Theano, Tensorflow: Work on static graphs → can adapt to dynamic sequences ⇒ but an breaks down
  - ↳ pytorch: does construct graphs on the fly!
- Higher-level abstraction disentangle the factors of variation!
- Good representation? ⇒ Makes other tasks easier!
- Smoothness prior/assumption → often times not accurate for high-dimensional spaces! ⇒ make function smooth by feature construction

wifi Ku: 356E 8

Talk 4 - Y. Bengio - Deep Learning III 29/08/17

② DNNs

- Hidden units ⇒ Meaning is not directly clear
  → Learning "discovers" significant features
  → No hand engineering / pre-defined meaning for hidden units
- Universal approx. Th.: single hidden layer
  → Why deep? ⇒ Functions can be approx. better / easier (due to compositionality) with less parameters!
  → Problems:
    * reasonably deep net vs. huge shallow nets
    * not easy optimization guarantee
    * not good generalization guarantee (finite training set)

  ⇒ Only tells us existence ⇒ does not guarantee that we find weight constellation
- breakthrough ⇒ ReLU → makes training possible
  → composition of nonlinearities ⇒ no vanishing gradients
  → sigmoid ⇒ going out
  → tanh ⇒ used a lot in RNNs?
  → softmax ⇒ sometimes used in attention → not an elementwise function → involves interaction between values ⇒ competition between nearby outputs
  → ReLU can't be used in RNNs ⇒ not upper bounded so values can blow up
  → "symmetry breaking" → Maxout → off ⇒ no weights flow ⇒ strong slope → specialization of units
  → Gradients sends credit information ⇒ no diffusion of information! → no solid explanation

- Expert knowledge → does not get lost ⇏ substitution with new architecture knowledge
- Bayesian NNs → very good on small datasets where overfitting is a problem ⇒ sample from posterior gives uncertainty
  → not a good application yet!
- ReLU problem ⇒ some neurons might die → negative weights to stuck ⇒ never get out
  to In practice: Only a few neurons are exhibiting this
- Noise injection in sigmoid activation fct. to help not getting stuck with the gradients!
- Loss function: Negative log-LH ⇏ cross-entropy

$$-\log f(x)_y = -\log p(y=y \mid x)$$

  to log: simplifies numerical stability and math
- $y \in \mathbb{R}^d$ → cat. given X
  → indep.: $-\log P(y|x) = -\sum_i \log P(y_i|x)$
  add up the losses
  to missing values in y ⇒ ignore loss!
  → dep.: need to capture cat. joint distr.

$$P(y_{i1}, \dots, y_{id} \mid x)$$

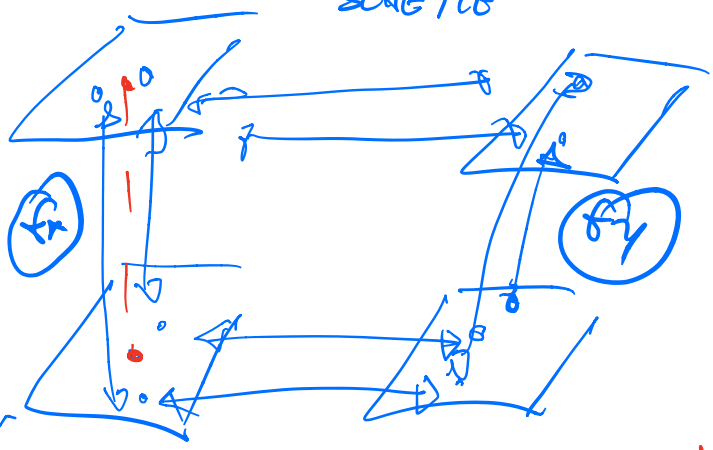
  to use factorization to sequentially predict with max-LH framework

- (cont.) afterwards use Bayesian NNs for last prediction that is hard (high var.)
- Right now not top research ⇒ no application on small datasets!

Hot Topics
- Bayesian VI for NNs ⇒ model complexity → own perspective!
- show comp. more generative modelling perspective

Bengio
- ELBO is biased Train - Val - Test better
- Use trad. framework to construct first layer text representations
  → Weights have small variance

ENCODING SCHEMES → e.g. autoencoder



Representation space

$x, y \Rightarrow$ represent different modalities

Symbolic space

combination of two "concepts"/attributes to generalize well to situations we have never see

since we have forced each attribute well

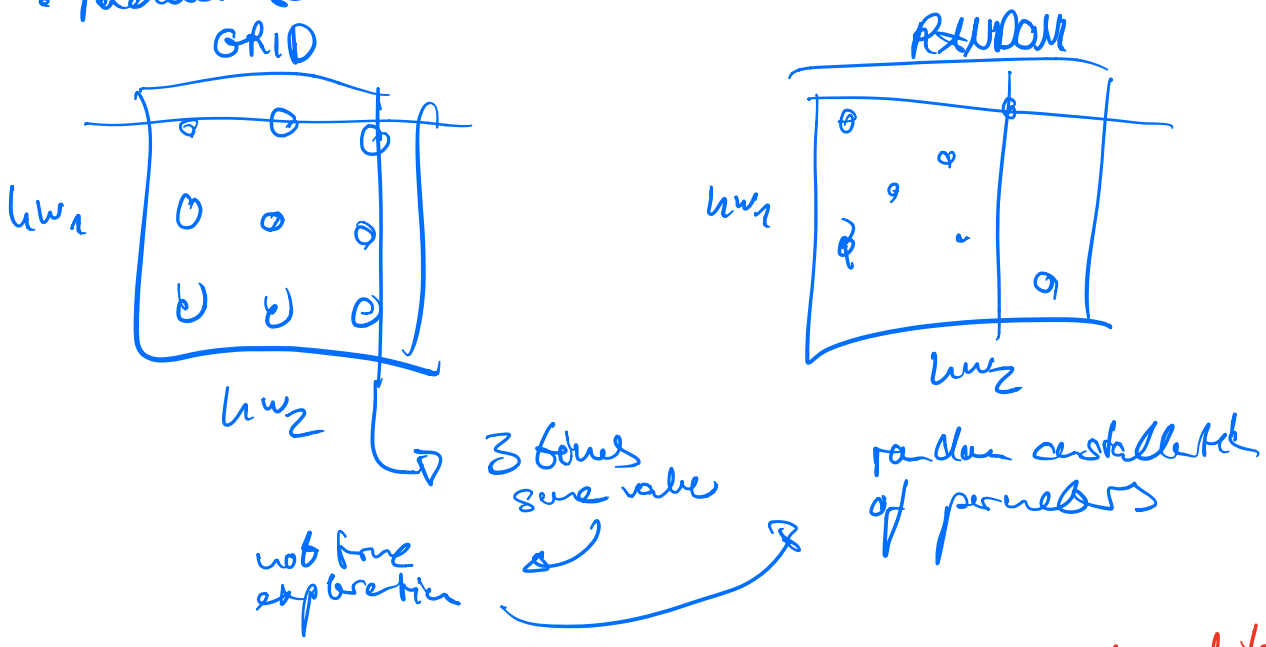
→ How to train $\Rightarrow$ No targets! → Simulate unsupervised generative modelling

- Multi-Task Learning
  - → generalization to new tasks (with few labels) $\Rightarrow$ Deep architectures learn good intermediate representations that can be shared across tasks
  - → "Good representations": disentangle underlying factors of variation $\Rightarrow$ each task contains a subset of features $\Rightarrow$ sharing of task knowledge between different tasks
  - → MTL: Learn underlying explanatory factors between tasks
  - → More tasks can lead to better performance

- Hyperparameters: Design decisions ⊕ opt. setting
  → Train ↔ Validation ↔ Test
  form of optimization!
  ↓ hyperparam tuning ≠ ↓ generalization error
  ↳ shuffle and split data ⇒ iid assumption violated
  → most important parameter: global learning rate
  ↳ In some sense equivalent to choosing the learning algorithm!
  → Minibatch: parallel CPU computation
  → Data deformations / transformations → allows to combat adversarial examples! ↳ no risk of too much noise
  → Cross-Validation: makes sense if dataset is small!
  regularization

- Sequential Evaluation: TS data
  average performance { TVT, TTVT, TTTVT } ⇒
  simulate what would have happened if we had data only up to t
  ↳ repeat with increased t and average test performance over all tests
  double CV
- Grid search ⇒ sufficient with more than 2 hyperparameters
- Random search ⇒ simple ⊕ efficient
- Bayesian optimisation → still need to set interval of values for parameters

- Random search



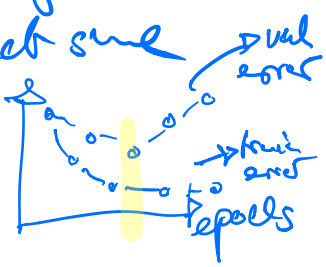
- Norm penalization
  → groups penalization:

$$\mathcal{E}(\theta) = L(y, \hat{y}; \theta) + \lambda \sum_l \sqrt{\sum_i w_{i,l}^2}$$

→ eliminate whole group of weights
→ feature selection
→ group push for uniform!

- Weight initialization: Glorot initialization
  → Biases: 0
  → weights: $\neq 0$ ⇒ propagation of info through gradients
  ↳ $\neq 0$ for tanh ⇒ as slow starts all gradients 0
  ↳ $\neq$ same value ⇒ hidden units will act same

⇒ sample $W_{i,j}^{(l)}$ ||



- Early stopping: number of training iterations
  → store training and validation error and choose sweet spot

Talk 5 – P. Ravikumar – GMs I 28/02/17

① Representation

- GMs and representation problems are complementary!
- AI → Effective reasoning in complex environments:
  I. Declarative knowledge: Structured knowledge base – e.g. prior
  II. Reasoning: derive optimal output → solve inference problem
  ↳ problem with logical reasoning: measures uncertainty
  knowledge base uncertainty
  ↳ GM provide framework to perform probabilistic reasoning in a comp. tractable way!
- GMs models declarative knowledge as a multivariate distr. over random variables ⇒ highly complicated function!
- (V, E): Each node $i \in V$ has a set of parents π_i

  ↳ DAG: $p(x_1, \dots, x_n) = \prod_{i=1}^{n} p(x_i | x_{\pi_i})$ joint factorization

  → Storage cost of general joint distr. where each var. can take r values: r^n
  → Storage cost of a DAG: for var. i, $r^{|\pi_i|}$

  ↳ as long as nodes don't have many parents (fan-in) we are able to compactly store the joint distr. ↳ go from global to local

- Undirected graph ⇒ loose symmetry with directed edges
  ⇒ Hidden Markov Model → FEEDBACK
  → Notion of locality needed for compact representation
  ⇒ Neighbors → consistency of conditional distr.
  ↳ maximal clique ⇒ set of fully connected nodes
  ↳ potential function $\phi_C(x_C)$: only depends on values x_C of max. clique variables X_C.
  ↳ non-negative, real-valued, otherwise arbitrary

$$p(x) := \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_{X_C}(x_C)$$

($\mathcal{C}$: set of max. cliques)

$$Z := \sum_x \prod_{C \in \mathcal{C}} \phi_{X_C}(x_C)$$

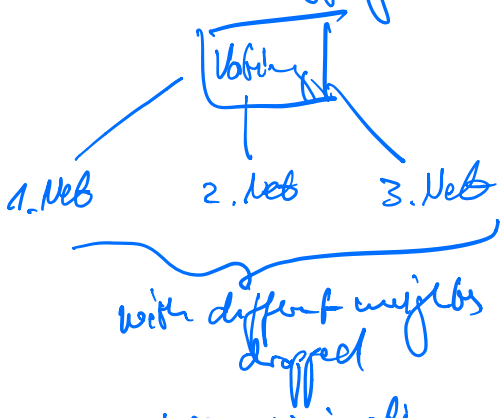
(Z → normalization constant)

↳ "partition function" ⇒ hard to compute!

Talk 6 – Y. Bengio – Deep Learning IV

23/08/17

- Dropout ⇒ Efficient Bagging → implicit ensemble method
  → only at training time
  ↳ seem to be robust to noise injection!



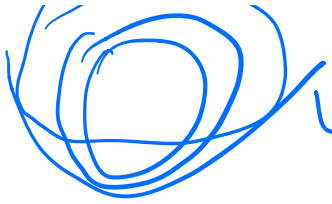
- Underfitting ⇒ better optimization
- Overfitting ⇒ more data / better regularization

- Curriculum learning
  → learn tasks sequentially! ⇒ family of objective functions

Once local minimum is found (after - medium concepts) go to next stage
↳ pretraining / guided learning
↳ use as initialization heuristic

TIPPS

- Train on small datasets ⇒ know that if we perform too well that there might be a bug!
- Track behavior of error curves during train! → Adapt code!
- Compare with simple baseline models! → is the setup right
- Look at examples that lead to overconfident mistakes! → Logistic regression
- There is no percentage for train-test ⇒ find the right split that makes things smooth!

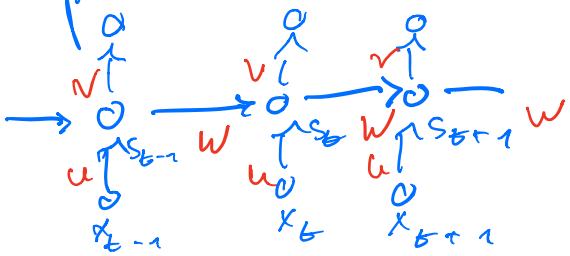


↳ thing of learning rate as a ball!
smaller l.r. make ball smaller!

- reduced variance optimization methods → then unclear how to apply!

③ RNNs

- Recursive update: $s_t = f_\theta(s_{t-1}, x_t)$



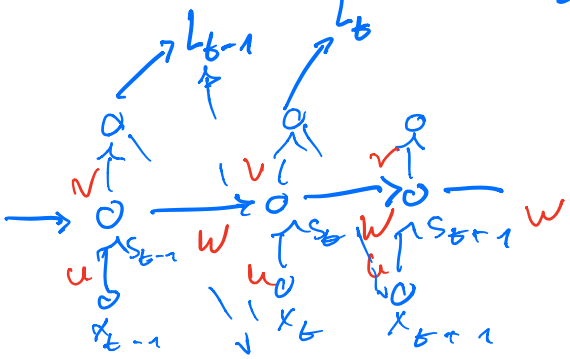
flexible output/input length

↳ unfold to obtain backprop structure

- RNN as directed generative model!

$$p(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \dots, x_1)$$

$$L_t = -\log p(x_t \mid x_{t-1}, \dots, x_1)$$



S2V
S2Sfixed
V2S
S2S

- "Teacher forcing" ↔ MLE → restrictive

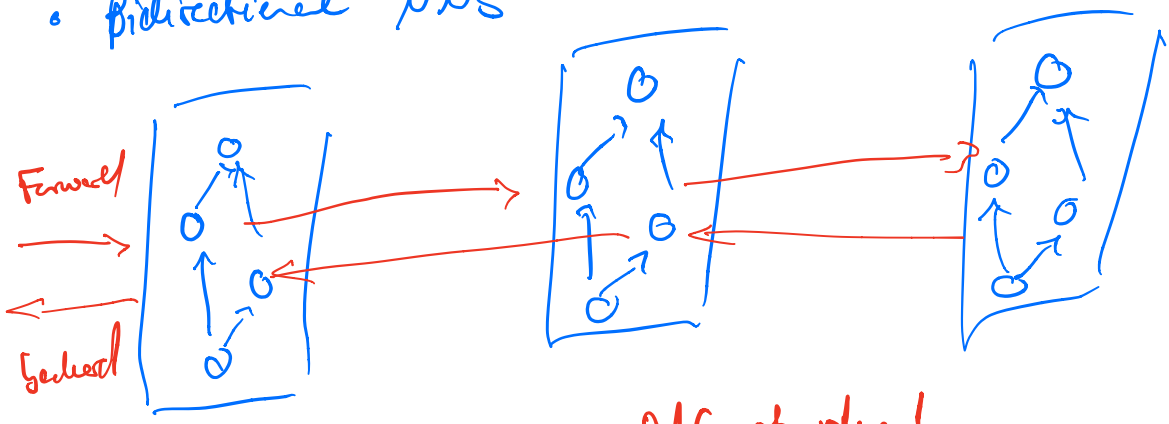
→ training vs. usage: input conditions at training time might be very different compared with test time

↳ solution: inject noise during training

methods: dithering or curriculum

→ "prefer" ⇒ use GRUs

- Skip connections: bypass layer! ⇒ shorter paths
- Bidirectional NNs



↳ backprop only requires DAG structure!

→ need to know the full sequence!

- Multiplicative interactions (dot products) seem to work well → gating
- Long-term relations ⇒ dynamical systems!
- Vanishing gradient problem!

→ different for MLP and RNNs ⇒ weight sharing

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \cdots \frac{\partial s_{t+1}}{\partial s_t}$$

→ λ(Jacobians) ≥ 1: gradient explodes } can't
λ(Jacobians) < 1: gradient vanishes } learn
long gradients!

→ gradient norm clipping ⇒ combats gradient explosion!

$\hat{g} \leftarrow \frac{\text{threshold}}{\|\hat{g}\|} \hat{g}$

- LSTMs ⇒ Gating unit → sigmoid ⇒ input, forget, output
- ~~Attention~~ mechanism ⇒ reading/writing in neural turing machine!
  - ↳ forgetting: vanishing gradients
  - ↳ Memory: higher-dim. state ⇒ avoid need for forgetting
- different time layers! ⇒ pyramidal layer structure!
- Delays ⇒ skip connections over time!
- New Situations → Search for underlying truth!
  ↳ even more abstract phenomena of world

- Good representations
  → controllable factors ⇒ want to learn jointly representations and policies! ⇒ Latent properties can be controlled independ. from other things in the environment!
  ↳ combine RL with deep learning
  ⇒ active inference & FEP

Talk - P. Kamiltner - GMs II

29/08/17

- Potential functions ⇒ cannot be cond. probabilities → do not integrate to 1!

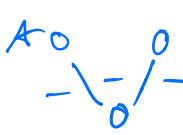
  ↳ also: marginal probs don't work! ⇒ "double counting"

  → pre-probabilistic interpretation

- Distributions are constrained in terms of conditional independence

$X_A \perp\!\!\!\perp X_B$: $p(X_A, X_B) = p(X_A)\, p(X_B)$ indep.

$X_A \perp\!\!\!\perp X_C \mid X_B$: $p(X_A, X_C \mid X_B) = p(X_A \mid X_B)\, p(X_C \mid X_B)$ cond. indep.



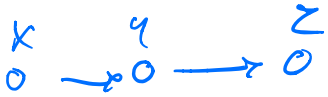
↳ general joint is a fully connected undirected graph!

⇒ no cond. indep. assertion

$$p(X_1, \dots, X_n) = \prod_{i=1}^{n} p(X_i \mid X_1, \dots, X_{i-1})$$ BCNT

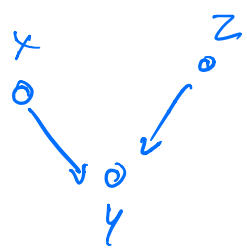
$$p(X_1, \dots, X_n) = \prod_{i=1}^{n} p(X_i \mid X_{\pi_i})$$ DAG

- graph allows us to formulate conditional independence ass. and to obtain special factorization of the joints.



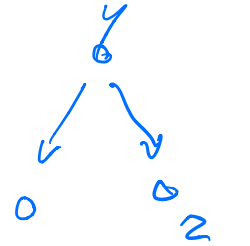
$p(Z \mid X, Y) = p(Z \mid Y)$

$\Leftrightarrow X \perp\!\!\!\perp Z \mid Y$



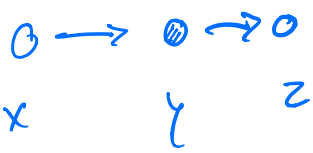
$X \perp\!\!\!\perp Z$

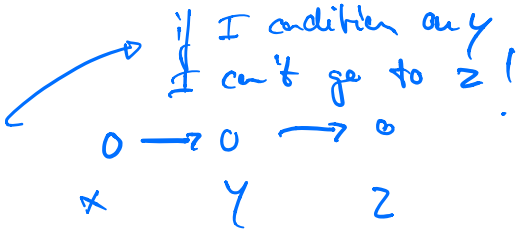
but not $X \perp\!\!\!\perp Z \mid Y$!

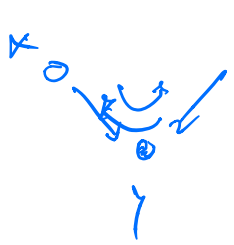


$X \perp\!\!\!\perp Z \mid Y$

- d-separation algorithm
  → blocking rules

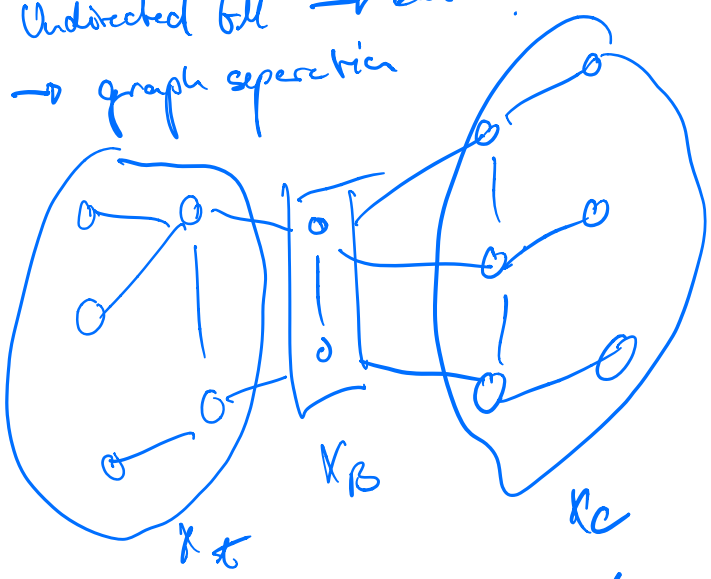






→ blocking on y allows us to go from x to z ⇒ colliders!
⇒ ("V-structure")

- Undirected GM → easier!
  → graph separation



⇒ advantage:
but factors don't have simple interpretation!

⇔ $X_A \perp\!\!\!\perp X_C \mid X_B$

Class 1: Given G, all distr. expressed as product of potential fcts. over maximal cliques ⇒ PROB. THEORY

Class 2: Given G, all distr. that satisfy all cond. indep. assertions specified by graph separation ⇒ GRAPH THEORY

⇒ Equivalence → Hammersley Clifford Theorem

- Directed GM:

  Def 1: Given G, $p(x_1, \dots, x_n) \triangleq \prod_{i=1}^{n} p(x_i \mid x_{\pi_i})$

  Def 2: Given G, all distr. that satisfy all cond. independence assumptions specified by Bayes Ball algo

  $\Rightarrow$ Equivalence

Talk 8 – P. Ravikumar – GMs III 30/08/17

- Graph $\leftrightarrow$ Locality $\leftrightarrow$ factor functions

  Joint Distr. Representation

- Edges $\rightarrow$ (X) correlation
  $\rightarrow$ (X) causation
  $\gg$ Conditional Independence

  Go Multiple Testing relation?

- Lack of edge $\rightarrow$ independence of two nodes given rest of nodes

- Factor functions $\Rightarrow$ fcts. over maximal cliques $\Rightarrow$ Undir.

  $\Rightarrow$ cond. prob. distr. of nodes given parents $\Rightarrow$ Dir.

  Def GM: Set of distr. given graph structure

  * non-parametric class of distr. $\Rightarrow$ If discrete always finite parameter
  * practice: specify parametric class to give factor fct. spec. form

- Ising Model $\Rightarrow$ binary RV $\rightarrow$ spin models

  $\rightarrow$ generalization to discrete GMs on categorical RV

  Larger cliques lead to more and more complex factor functions $\rightarrow$ Simplify

  focus on small d-size cliques!

→ pairwise function specification can be linear / non-linear / etc.

- Gaussian GM: variables are <u>fully-valued continuous</u>

$$p(x, \theta) := \exp\left\{ - \sum_{(s,t) \in E(G)} \theta_{st}\, x_s x_t + \log \text{dens}(\theta) \right\}$$

  - ↳ joint: multivariate Gaussian distribution
  - ↳ θ_{st} : scaling of quadratic ⇒ relation to precision!
    ⇒ nice interpretation of edge weights

- high-dim problem: combats by combats by compact representable of pairwise functions!

- Cont value data / skewed continuous data

⇒ Exponential family GMs

→ Node-cond. distr.: $p(x_i \mid x_{-i})$ → <u>all other nodes</u>

  - ↳ Ising: Bernoulli
  - ↳ categorical: Multinomial
  - ↳ GGM: univariate Gaussian

→ $$p(x; \theta) := \exp\left\{ \sum_{i \in I} \theta_i \phi_i(x) + C(x) - A(\theta) \right\}$$

- $p(x; \theta)$: univariate distr. family
- θ_i : params.
- $\phi_i(x)$: sufficient stats
- $C(x)$: base measure
- $A(\theta)$: log partition normalization

→ existence of consistent joint distr. if each node-cond. distr. is specified by exp. family distr.

$$p(x) = \exp\Big\{ \sum_s \sum_s \theta_s \phi_s(x_s) + \sum_{s \in V} \sum_{t \in \mathcal{N}(s)} \theta_{st} \phi_s(x_s) \phi_t(x_t) + \sum_{s \in V} \sum_{t_2, \dots, t_k} \theta_{s \dots t_k} \phi_s(x_s) \prod_{j=2}^{k} \phi_{t_j}(x_{t_j}) + \sum_s C_s(x_s) - A(\theta) \Big\}$$

↳ factor function is just product of sufficient stats at clique nodes scaled by θ's ⇒ tensor prod.

↳ nice compact factor functions ⇒ allows us to model wizard debert!

↳ $k > 2$: Is a problem ⇒ need more structure!

- Other constraints from cond. independence:
  → reduced parametrization ⇒ algebraic structure imposed
  ↳ constrain to lower order factors!

(2) LEARNING

- Goals:
  (1) Learning the graph structure → Model selection
  (2) Learning the parameters of a parametric model → Parameter Inference

$$p(x; \theta, G) = \frac{1}{Z(\theta)} \exp\Big\{ \sum_{(s,t) \in E(G)} \theta_{st} \phi_{st}(x_s, x_t) \Big\}$$

ϕ_{st}: library potential function

1a) Model Selection ↪ Undir. Case ↦ deal with normalizing constant ⇒ NP-hard!

- score-based approaches: search over space of graphs with score fct score based on penalized likelihood
- constraint-based approaches: hypothesis test for cond. indep.
  → needs a lot of samples if we cond. on a lot of variables ⇒ sometimes only constraint on subsets
  → PC: Multiple testing error correction!
- Estimating neighborhoods: go back to locality!
  → high-dim stat. model est.
  → high-dim hyp. testing

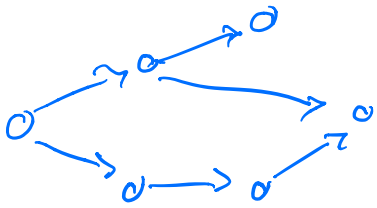
↳ Sparse logistic regression! for each node ↦ regularization
⇒ get neighborhood by combining neighborhood estimates in a consistent manner (AND, OR)

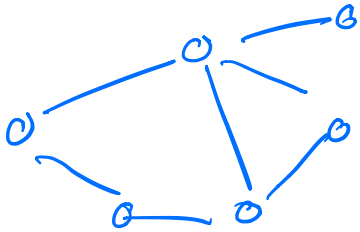
↳ regularization: $\lambda \gtrsim \sqrt{\frac{\log p}{n}}$ ⇒ consistent result

recovers exact graph structure with very very high prob.

1b) Model Selection ↪ directed case

- Moralization ⇒ Marry all parents ↦ fully connect parents





↦ learn undir. graph "signal hidden inside"

- Orient edges using cond. independence tests
  ⇒ PC algo ↦ not very scalable!

2 Inference

- Need to approx. ⇒ otherwise inference is NP-hard
- We now structure, edge weights and exp. family distr.

$$p(x_F \mid x_E) \rightarrow p(x_E, x_F) = \int p(x_E, x_F, x_R)\, dx_R$$

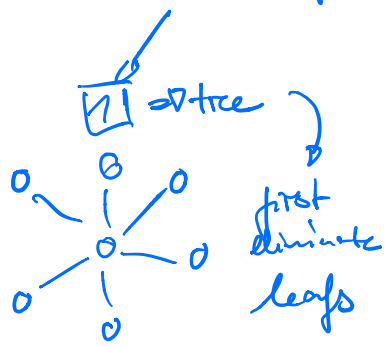
$$= \frac{p(x_E, x_F)}{p(x_E)} \qquad p(x_E) = \int p(x_E, x_F)\, dx_F$$

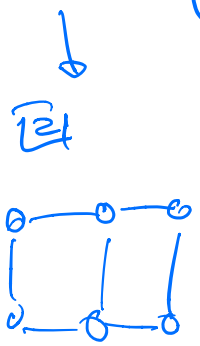
↳ can take exponential time if R is very large! ⇒ high-dim integral

- Exact Algos
  - → exp. in tree-width of graph
  - → Variable Elimination ⇒ leverages locality
  - → Sum-product for tree-structured graphs

↳ Variable Elimination: variable-wise marginalization
to keep creating new local factors!

- → Elimination ⇔ connection of all its neighbors
- → Comp. complexity is exp. in size of largest clique obtained when eliminating variables
- → Tree-width: 1 minus size of largest clique size
  to NP hard to find best possible ordering of elimination order





↳ Sum-product:

$$p(\mathbf{x}) \propto \prod_{i \in V} \phi^{E}(x_i) \prod_{(i,j) \in E} \phi(x_i, x_j)$$

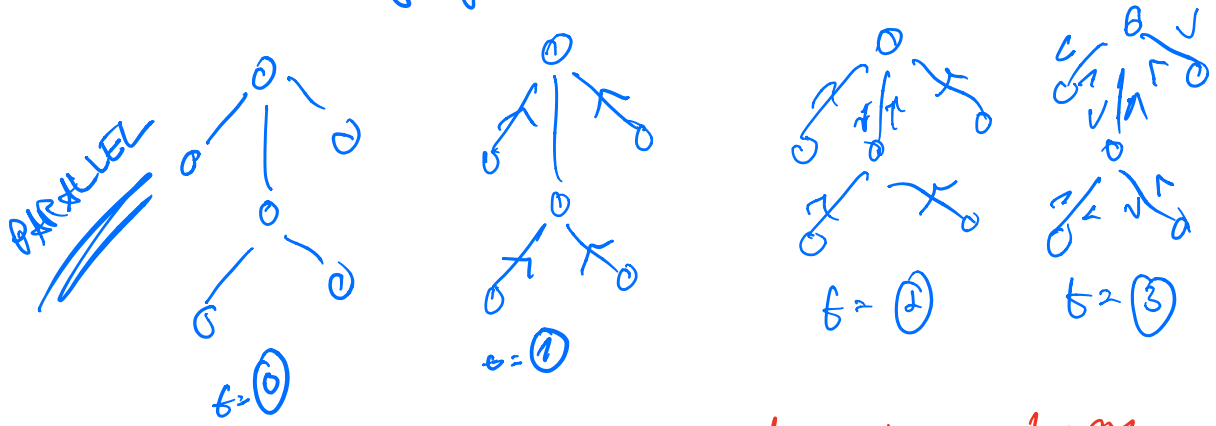
Message Passing Algo:

product of local factors

messages of other neighbors

$$m_{i \to j}(x_j) = \sum_{x_i} \left(\phi^{E}(x_i)\, \phi(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k i}(x_i) \right)$$

Node can send message to neighbors when it has received messages from all of its remaining neighbors.



↳ Exact for non-tree structured graphs ⇒ Loopy belief propagation → variants: tree-re-weighted belief prop. / convex belief prop.

- MAP → similar to marginal computation via Elimination
  ↳ maximize instead of sum / integrating

$$\mu_{ji}^{\max}(x_i) = \max_{x_j} \left(\phi^E(x_j)\, \psi(x_i, x_j) \prod_{i' \neq i} \mu_{i'j}(x_j) \right)$$

$$\max_x p^E(x) = \max_{x_i} \left(\phi^E(x_i) \prod_{j \in N(i)} \mu_{ji}^{\max}(x_i) \right)$$

- Variational inference $\leftrightarrow$ mean field / structured MF
  - $\rightarrow$ convex opt. that is intractable since there are exp. many constraints! $\Rightarrow$ connection of local to global
  - $\rightarrow$ connection to sum-product

Take home: Exact inference is intractable!

- Message passing algos are very scalable since we perform many locally simple operations!

Talk 9 - P. Richtárik - Random Optimization 30/08

- Empirical Risk Minimization:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \text{loss}\left(h_x(a_i), b_i\right) + g(x)$$

↳ Monte Carlo integration of risk:

$$\min_{x \in \mathbb{R}^d} \mathbb{E}_{(a_i, b_i) \sim \mathcal{D}} \left[\text{loss}(h_x(a_i), b_i) \right]$$

- Linear System Problem Formulation:

LOSS $\quad \mathbb{1}_{\{b_i\}}(t) = \begin{cases} 0, & t = b_i \\ +\infty & \text{otw.} \end{cases}$

REG $\quad \frac{1}{2} \|x - x^0\|_B^2$

- Legendre - Fenchel Transform $\Leftrightarrow$ Convex conjugate $\rightarrow$ Dual
  $\rightarrow$ Map from fct. to another fct.

↳ Continued notes in the slides

Talk 10 – S. Russell – Open-universe prob. models 31108

- First-order logic ⇒ rules too unwieldy // necessary to learn! rules using propositional logic – circuits / GMs
- Probabilistic GMs ⇒ Bayesian Nets (DAG) ↦ BUGS → STAN
- First order prob. languages ⇒ combine Bayes Nets with logical notation

World has things in it and we don't know what they are ⇒ 1st order open universe prob. languages

↳ closed-universe: unique names, domain closure ⇒ PROLOG

↳ open-universe: allow uncertainty over existence ⇒ BLOG

Bayesian Logic

Talk 11 – C. Szepesvári – Bandits I — 31/08/17

① Stochastic finite-armed bandits

- Hostelles, Bush (1953) – Animal Learning (Mouse in T-maze)
- Thompson (1933) – Medical Trial // A/B Testing
  → don't throw away information → sequential design
- Lai, Robbins (1985) – UCB ⇒ asy. regret optimality
  Lo Auer et al (2002) – finite iter. result

- Bandits ⇔ Simplest version of problem of AI
  ⇒ Decision making under uncertainty → Ē vs. E
- Policy: Mapping of all possible histories to future actions! Ugo
- Min regret: 0 has special meaning ⇒ shifting invariant!
  Lo still: scaling invariant
- 0-regret learner: $\frac{R_n}{n} \to 0 \iff R_n = o(n)$

speed → How slow does R_n grow?

- Time horizon is important: No future? Exploration does not make sense!

$A_t \sim \pi(\cdot \mid A_1, X_1, \dots, A_{t-1}, X_{t-1})$ ⇒ current action follows policy given history!

⇒ ν: environment ⇒ (ν, π) joint distr. of interest!

- Regret decomposition:
  - $\nu = (P_1, \dots, P_K)$: bandit environment
  - $\Delta_i(\nu) = \mu^*(\nu) - \mu_i(\nu)$: sub-optimality gap of arm i
  - $T_i(t) = \sum_{s=1}^{t} \mathbb{1}_{\{A_t = i\}}$ → random due to reward
  - ⇒ fixed π, ν, then regret R_n:

$$R_n = \sum_{i=1}^{K} \Delta_i \, \mathbb{E}[T_i(n)]$$

→ don't observe, have to estimate!

- Concentration of measure
  - → Markov: $\mathbb{P}(|X| \geq \varepsilon) \leq \frac{\mathbb{E}[|X|]}{\varepsilon}$
  - → Chebyshev: $\mathbb{P}(|X - \mathbb{E}[X]| \geq \varepsilon) \leq \frac{\mathrm{Var}(X)}{\varepsilon^2}$
  - → Subgaussian: X is σ-subgaussian if $\forall \lambda \in \mathbb{R}$:

$$\mathbb{E}[\exp(\lambda X)] \leq \exp(\lambda^2 \sigma^2 / 2)$$

  - → tails are decaying faster! → from Chernoff bounding!
  - → If X

$$\mathbb{P}(X \geq \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

  - → Ass.: RVs in bandit problem are sub-gaussian!

(1).1 Explore-then-commit ⇒ follow the leader after some exploration of all arms
⇒ ETC

→ ε-greedy: part of the journey!
→ w.r. number of iter. we do exploring

(1).2 Instance-adaptive algos

(2).1 UCB ⇒ Optimism in the face of uncertainty
→ Choose actions as if environment is as nice as possible
→ In beginning: each arm is explored once!

Talk 12 - C. Szepesvári - Bandits II 01.10.16

- Recurring theme: For each A, $\exists C > 0$ s.t.

$$R_n \leq C\sqrt{n}$$

↳ balance / trade-off

- UCB ⇒ many different versions depending on how we accumulate data!

↳ simulation: balancing of upper bounds

$UCB_i(t-1,\delta) = \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}}$ ← $2\log(Conf)$

$A_t = \arg\max_i UCB_i(t-1,\delta)$ ← pulls us to exploration

→ bad performance if environment is non-stationary
↳ bad can modify if we now the speed of change!

- Asympt. Conv.
- Finite time
- Worst Case

→ Tend to result in good algos!

→ Dangerous current: Might be back to "only" minimize bound on expected regret! ⇒ variance matters!

② Stochastic Bandits

→ Adversarial Environment:

ν point in $([0,1]^K)^n$: $\nu = (x_1, \dots, x_n)$ where $x_t \in [0,1]^K$

↳ $t=0$: ν chooses $x_1, \dots, x_n$

$t = 1, 2, \dots$:

* Based on history, choose action $A_t \in [K]$ to send to ν
* Env. sends reward $X_t = x_{t, A_t}$ to learner

⇒ Action chosen to minimize the worst-case!

$$R_n = \max_i \sum_{t=1}^{n} x_{t,i} - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right]$$

↓ rewards are non-random

↳ results from random action I_t

choosing learners vs. choosing actions } ONLINE !!! $[r, A]$

important! ⇒ otherwise adversarial would simply exploit your deterministic behavior! ⇒ actually have to randomize!

↳ reactive opponents → MDP formulation

$$R_n^*(\pi) = \sup_{r \in [0,1]^{nK}} R_n(\pi, r)$$ → worst-case regret

⇒ Existence: $R_n^*(\pi) = o(n)$ → sublinear regret!

$$\mathbb{E}\left[\max_i \sum_{t=1}^{n} X_{t,i} - X_{t,I_t}\right] \geq \max_i \mathbb{E}\left[\sum_{t=1}^{n} X_{t,i} - X_{t,I_t}\right]$$

Exp. Adversarial regret (left) — Stochastic regret (right)

↳ It does not really matter if world is truly stochastic !?

- Exp 3: Exponential-weights algo for Exploration a. Exploitation
  ⇒ Estimation of rewards ⇒ Only $X_t = X_{t,I_t}$ is seen
- Minimax Regret: $R_n^*(\mathcal{E}) = \inf_{\pi \text{ strategy}} \sup_{r \in \mathcal{E}} R_n(\pi, r)$

Talk 18 - S. Nowozin - Probabilistic DL 01/09/17

- Evolution was not able / did not get enough time
  ⇒ It is all about representations and adaptive learning
- Lower dimensional subspaces capture abstract structures!
  - accessible and controllable representations
    - meaning : word2vec e.g.
  - weak / implicit supervision
  - robust learning
- Advances in Density Est.

① Integral Prob. Metrics

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{f \in \mathcal{F}} \left| \int f dP - \int f dQ \right|$$

↳ Wasserstein GANs

- depending on class of functions you get different metric

② Proper Scoring Rules

↳ Var. Inference

$$S(P,Q) = \int s(P,x)\, d\,Q(x)$$

- most used: log likelihood!

③ f-divergences

↳ GANs

$$D_f(P\|Q) = \int q(x)\, f\left(\frac{p(x)}{q(x)}\right) dx$$

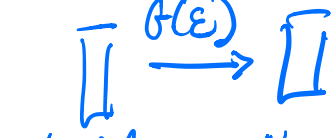
($\frac{p(x)}{q(x)} \geq 0$)

f: generator fct.

- KL divergences belong to this class

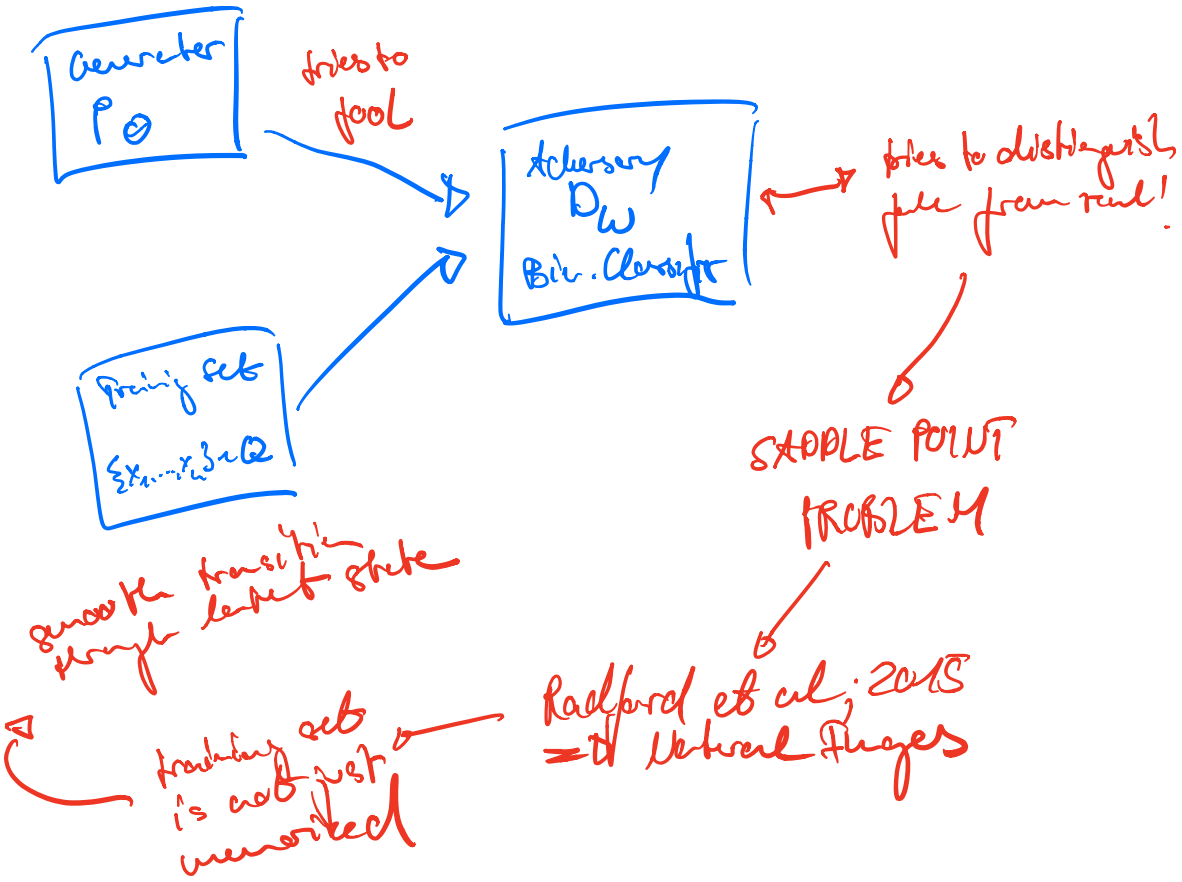
* $f(1) = 0$
* convex

- Implicit Model ⇒ fixed prob. distr. → apply det. to obtain a sample → density fct. might not be defined explicitly
  - ↳ Vanilla sampler / Likelihood-free models
  - → flexible modelling ⇒ very flexible
  - → Problem: $p(x)$ might not be defined
    - ↳ can't reason about all events ⇒ Non-Bayesian
    - not all dimensions can be recovered
    - ↳ possible solution: Hold out → But: how to choose?



↳ det. map // has not to be invertible

- GAN = Implicit model ⊕ Estimation procedure



- Relation to f-divergences
  → Every convex fct. has a Fenchel conjugate!

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t \frac{p(x)}{q(x)} - f^*(t) \right\} dx$$

$$\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p(x) T(x)\, dx - \int_{\mathcal{X}} q(x) f^*(T(x))\, dx \right)$$

$$= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x \sim P}[T(x)] - \mathbb{E}_{x \sim Q}[f^*(T(x))] \right)$$

to approx. by MC integration / samples

$\Rightarrow$ GAN → Jensen-Shannon Divergence!

$$\min_\theta \max_w \mathbb{E}_{x \sim P_\theta}[\log D_w(x)] + \mathbb{E}_{x \sim Q}[\log(1 - D_w(x))]$$

$\Rightarrow$ f-GAN

$$\min_\theta \max_w \left(\mathbb{E}_{x \sim P_\theta}[T_w(x)] - \mathbb{E}_{x \sim Q}[f^*(T_w(x))] \right)$$

- Why does it work if $p(x)$ is not defined?
  → generalize f-div. by introducing geometry of space to consider kernel ⇒ extend! → add noise
- GAN training is not always stable
  → numerical issues for 0-player non-smooth game ⇒ solution: regularizers!
- Variational Autoencoders → more stable ⇒ Related to Classical ML

$$\boxed{p(x|\theta)} = \int p(x|z,\theta)\boxed{p(z)}\,dz \rightarrow \text{intractable integral}$$

$\boxed{p(x|\theta)}$ → Mixture of Normals

$\boxed{p(z)}$ → multivariate Normal ⇒ fixed prior

→ MLE training ⇒ VI formulation

$$\log \int p(x|z,\theta)\,\frac{p(z)}{q(z)}\,q(z)\,dz$$

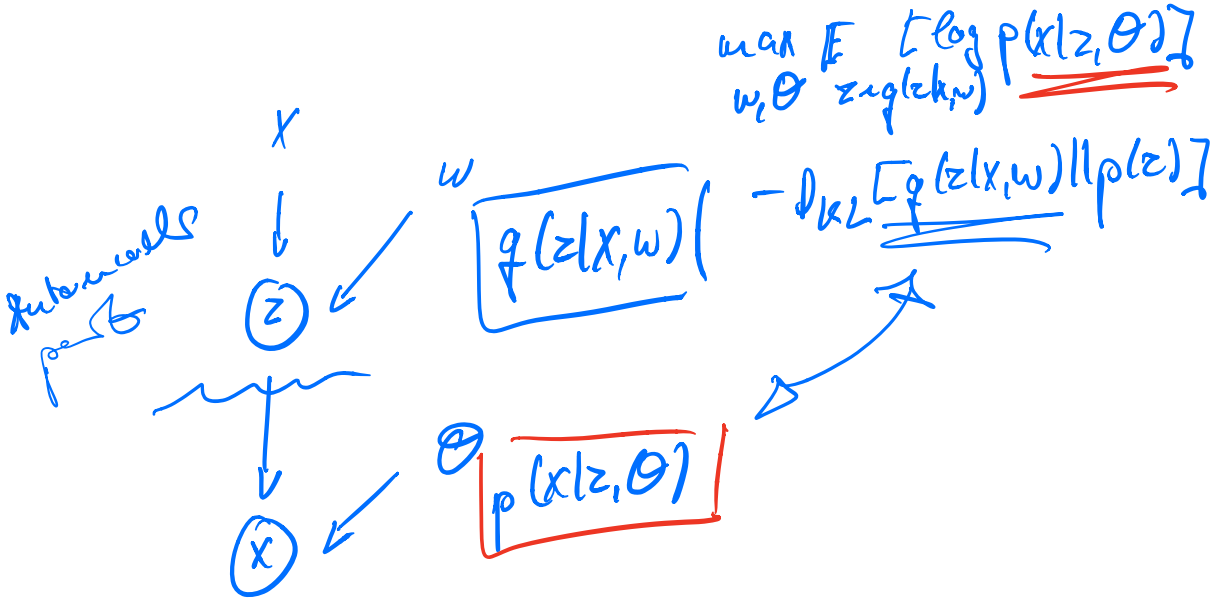
$$= \log \mathbb{E}_{z\sim q(z)}\left[p(x|z,\theta)\,\frac{p(z)}{q(z)}\right]$$

$$\geq \mathbb{E}_{z\sim q(z)}\left[\log p(x|z,\theta)\,\frac{p(z)}{q(z)}\right] \quad \text{by Jensen's Ineq.}$$

$$= \mathbb{E}_{z\sim q(z)}\left[\log p(x|z,\theta)\right] - D_{KL}(q(z)\|p(z))$$

$\mathbb{E}_{z\sim q(z)}[\log p(x|z,\theta)]$ → Exp. Var. Free Energy → Use samples in practice!

→ Inference Network ⇒ new network that learns all variational parameters ⇒ no direct optimization → network does the optimization
↳ "Amortization"



→ Variance reduction ⇒ Reparametrization Trick
↳ Doersch: "Tutorial on VAE"
↳ makes the gradients flow more smoothly!

- Problems:
  - ELBO can be very loose!
  - Image generation of VAEs ⇒ can be very blurry