ECE521 Lecture 24

Supervised learning using graphical models

UNIVERSITY OF TORONTO

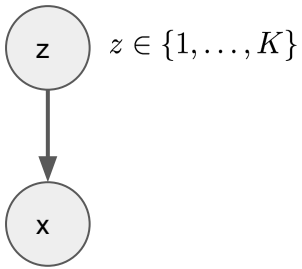
Outline

- **Supervised learning using graphical models**
  - Learning class label posteriors
  - Conditional random fields
  - Combining deep learning with graphical models
- Applications: put everything together
  - Image segmentation
  - AlphaGo

Graphical models

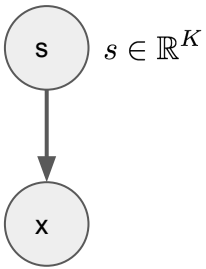
- Graphical models are useful for representing probability distributions for various learning and inference problems:
  - Specifically, in the unsupervised learning setting, we have seen the following models:

Mixture models



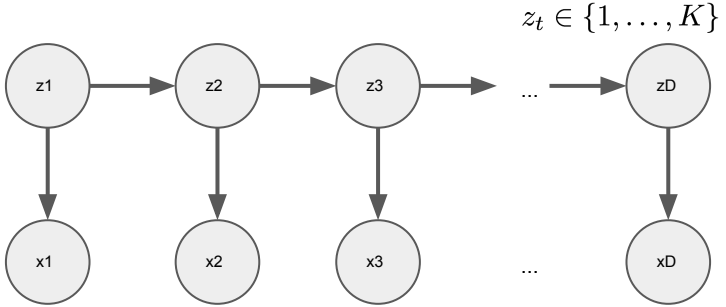
$$p(x, z) = p(z)p(x|z)$$

Continuous latent space models



$$p(x, s) = p(s)p(x|s)$$

Hidden Markov models

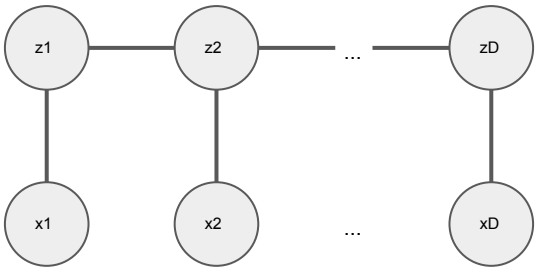


$$p(x_1, \ldots, x_D, z_1, \ldots, z_D) = p(z_1) \prod_{t=2}^{D} p(z_t|z_{t-1}) \prod_{t=1}^{D} p(x_t|z_t)$$

Graphical models

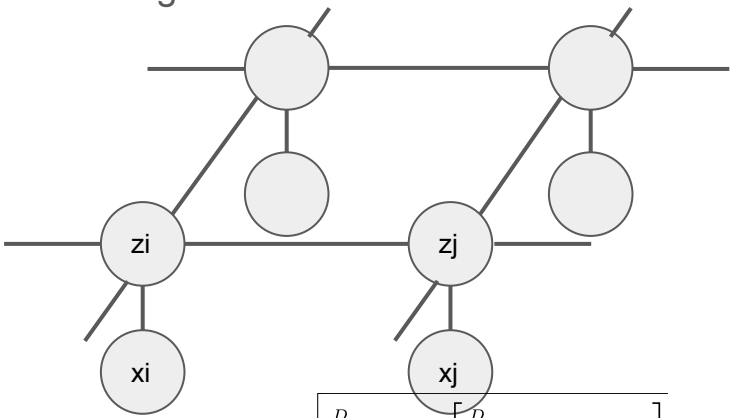
- Graphical models are useful for representing probability distributions for various learning and inference problems:
  - Specifically, in the unsupervised learning setting, we have seen the following models:

chain-structured Markov random field



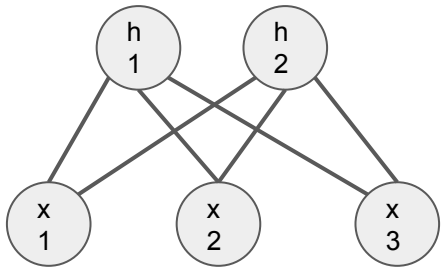
$$p(x_1, \ldots, x_D, z_1, \ldots, z_D) = \frac{1}{Z}\left[\prod_{t=2}^{D}\psi(z_{t-1}, z_t)\right]\left[\prod_{t=1}^{D}\psi(z_t, x_t)\right]$$

grid-structured MRF



$$p(x_1, \ldots, x_D, z_1, \ldots, z_D) = \frac{1}{Z}\sqrt{\prod_{t=1}^{D}\psi(z_t, x_t)\left[\prod_{t=1}^{D}\prod_{y \in Ne(z_t)}\psi(z_t, y)\right]}$$

Restricted Boltzmann Machine



$$p(x_1, \ldots, x_D, h_1, \ldots, h_M) = \frac{1}{Z}\prod_{i=1}^{D}\prod_{j=1}^{M}\psi(x_i, h_j)$$

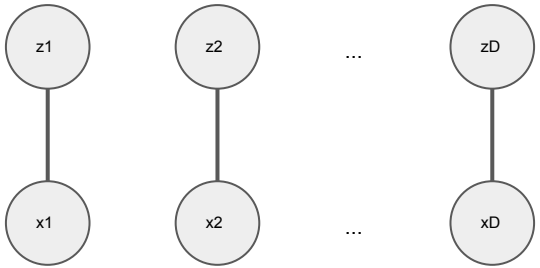
Learning graphical models

- Learning is to adjust the model parameters such that the probability distribution represented by the graphical model matches with the observed data:
  - Under the **incomplete** data setup (or the partially observed graphical model), i.e. only given the observed variable x, we need to marginalize out the latent variables or perform EM algorithm.
  - Under the **complete** data setup (or the fully observed graphical model), i.e. given both the observed and the latent variables, we can directly optimize the joint distributions represented by the graphical model. (think of learning naive Bayes model given both label/cluster assignment z and observed x)

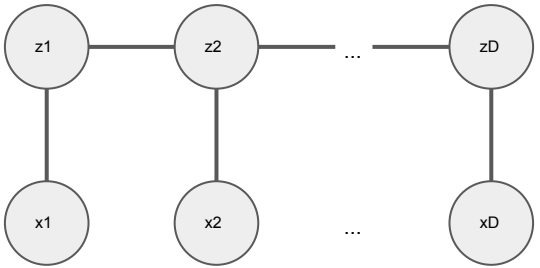
Diversion: introducing constraints to supervised learning

- The advantage of using graphical models is to represent interesting dependencies (or **soft constraints**) between the random variables. In a fully observed graphical model, useful dependencies across the class labels are captured by the edge connections in the graphical model. These dependences / constraints can help **correcting mistakes** in label prediction.
  - e.g. Zs are the class labels in a sequence prediction task.

No constraint between the labels



Chain-structured model can learn pairwise constraints between the labels



(e.g. temporal smoothing across the labels)

Learning fully observed graphical models

- Learning the fully observed graphical models or the joint distributions can be **difficult**, even if we are given the complete data of both the latent and the observed variables. Usually it is hard to learn the p(x|z) term.
  - The observed variables are too high-dimensional, e.g. high resolution images.
  - The modelling assumption of the p(x|z) term does not hold for the actual data, e.g. in the naive Bayes classifier.
- Most of the time, given the complete data setup, all we care about is a **subset of the variables** conditioned on the rest. e.g. if z is the class label, we may only care about p(z|x) for making predictions.

Learning posterior distributions

- To directly learn the conditional distribution or the **posterior** over the class label variables in the graphical model requires **inference**. We then adapt the model parameters to match the conditional data. (To avoid confusion with MAP estimation of parameters, the MAP estimation of the labels/states is often referred as “decoding”)
    - Consider the naive Bayes classifier example, where the latent variable z is the class label and the observations x are the input features.
    - We may treat the model as fully observed and learn the model parameters by maximizing the joint probability on the complete data. The class prediction p(z|x) on the new test data is given by the Bayes’ rule. (**the generative approach**, it does not work well in general)
    - Alternatively, we can directly learn the parameters by maximizing the inferred posterior probability p(z|x) on the training data. (**the discriminative approach**, this is equivalent to learning a logistic regression model, see lecture 15)

Supervised learning in graphical models

- In general, we can obtain the posterior distribution over the latent variables/class labels in a graphical model via some **inference algorithms** and θ are the model parameters.

$$p(z_1, \ldots, z_D | x_1, \ldots, x_D, \theta) = \frac{p(x_1, \ldots, x_D, z_1, \ldots, z_D | \theta)}{p(x_1, \ldots, x_D | \theta)}$$

- Learning the **log conditional/posterior distribution** of the class labels:

$$\max_{\theta} \quad \log p(z_1, \ldots, z_D | x_1, \ldots, x_D, \theta)$$

$$= \underbrace{\log p(x_1, \ldots, x_D, z_1, \ldots, z_D | \theta)}_{\text{fully-observed objective}} - \underbrace{\log p(x_1, \ldots, x_D | \theta)}_{\text{unsupervised learning objective}}$$

Learning the conditional distribution is equivalent to the generative approach regularized by the negative log marginal distribution.

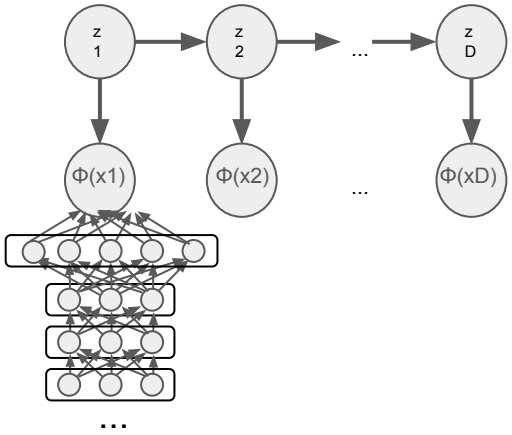
- Predicting the class label for a new test data point requires "**decoding**" the best set of labels, i.e. max-product algorithm to figure out argmax p(z|x).

Supervised learning in graphical models

- Learning the conditional/posterior class label distribution in a graphical model is also called **structured prediction**. If the graphical model is a Markov random field, it is also called the **conditional random field (CRF)** model.
    - The advantage of learning CRF models (discriminative approach) over fully observed Markov random fields (generative approach) is the same as learning logistic regression v.s. the naive Bayes classifiers. i.e. p(x|z) is difficult to model and we do not want to “waste resources” to model the input variables that are always observed.
    - The disadvantage of learning the conditional distributions directly is that it requires fully labelled training data, i.e. fully observed graphical models where generative approach can handle missing data problem or semi-supervised learning.

Combine deep learning with graphical models

- Typically the conditional distributions in a graphical model are very simple. They are either **discrete** or **Gaussians**.
- We would like to keep them this way so **inference is still easy**.
  - We can significantly improve the representational power of the conditional distribution graphical models by using non-linearly transformed observed variables.
  - i.e. have the conditional distribution on $p(\phi(\mathbf{x})|z)$ where $\phi(\cdot)$ is a neural network.
  - Jointly learn the graphical model and the neural network only when the model is grounded by training label data.



Outline

- Supervised learning using graphical models
  - Learning class label posteriors
  - Conditional random fields
  - Combining deep learning with graphical models
- **Applications: put everything together**
  - Image segmentation
  - AlphaGo

Application: image segmentation

Input

Final Output

Problem setup:

Training:

image pixels x in RGB

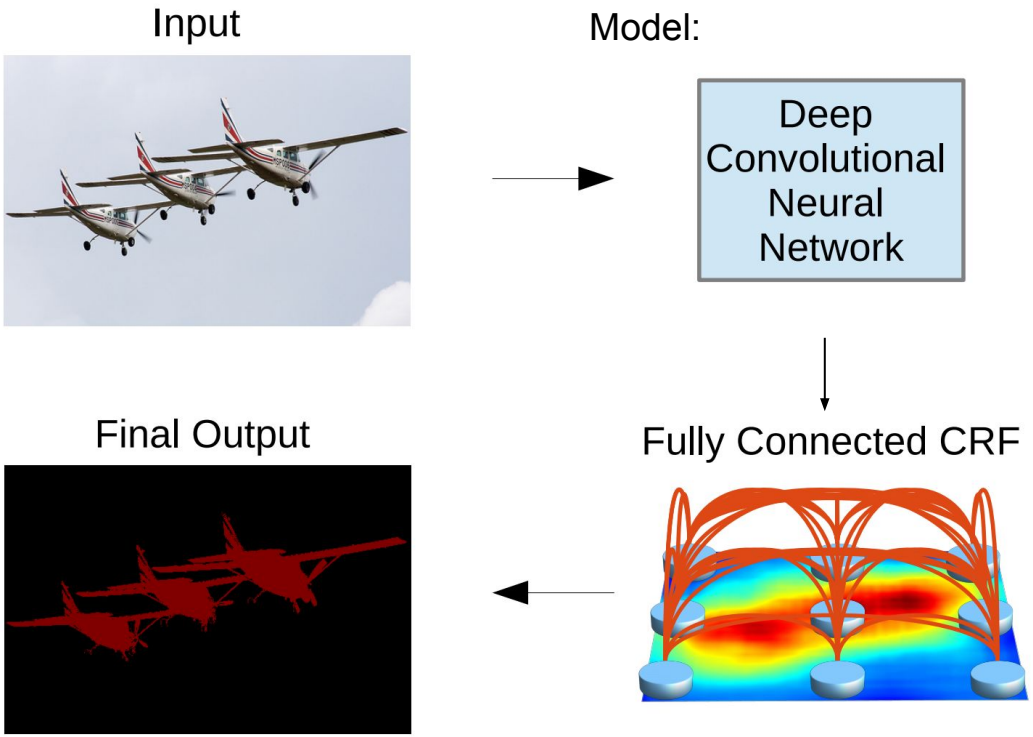
pixel labels z in {#classes}

Test:

pixel labels z in {#classes} given the input image

Chen et. al., 2015

Application: image segmentation



Problem setup:

Training:

image pixels x in RGB

pixel labels z in {#classes}

Test:

pixel labels z in {#classes} given the input image

Chen et. al., 2015

Application: image segmentation

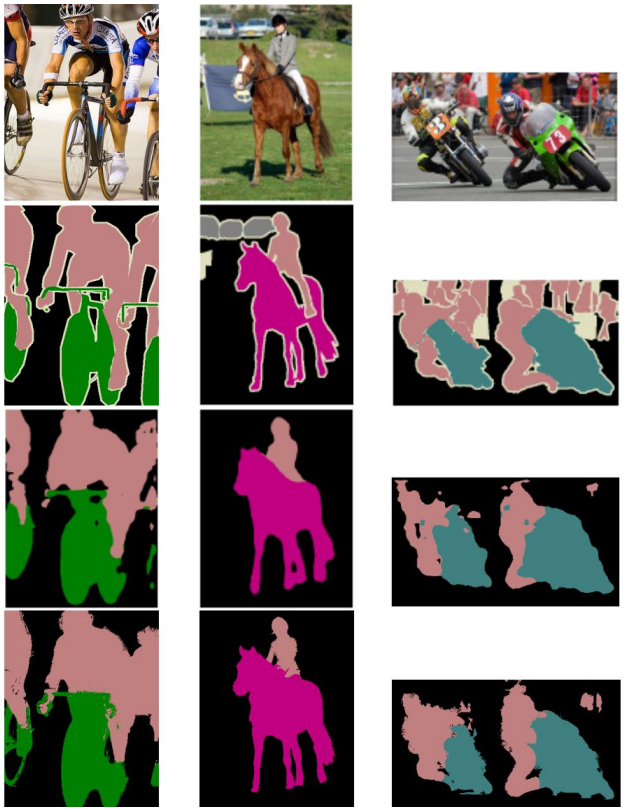
Results:

Ground truth:

without CRF:

Inference/decoding help correcting the mistakes from the independent predictions

with CRF decoding:



Problem setup:

Training:

image pixels x in RGB

pixel labels z in {#classes}

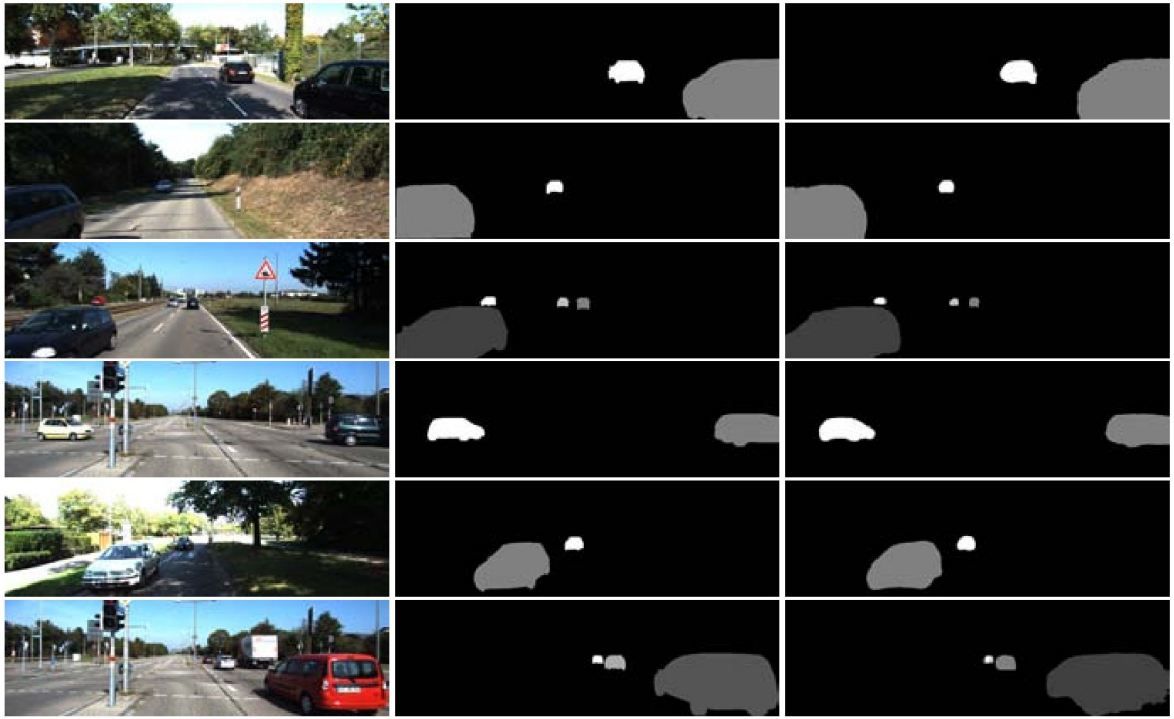
Test:

pixel labels z in {#classes} given the input image

Chen et. al., 2015

Application: instance segmentation

Results: Ground truth: with CRF:



Problem setup:

Training:

image pixels x in RGB

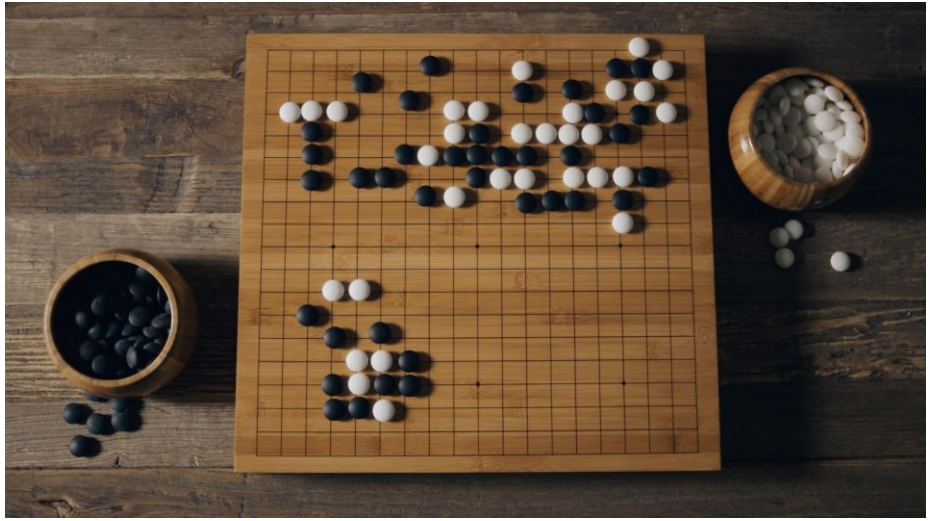
pixel labels z in {#classes x #instance}

Test:

pixel labels z in {#classes x #instance} given the input image

Zhang et. al., 2015

Putting everything together



- The entire game space is huge.
- Greedily learning the expert move is error-prone and not optimal.
- Need to perform some inference/decoding to correct the mistakes.

Problem setup:

Training:

Input: current state of the game, 18 x 18 Go board

Output: an expert move for the current game state

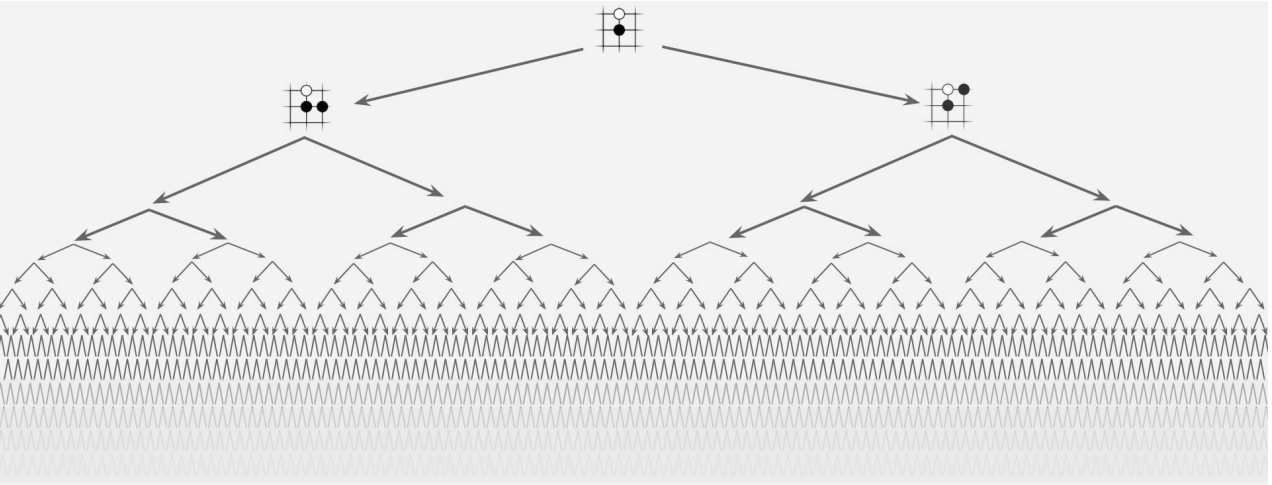
Test:

Output: a move in {#legitment moves in the current state of the game}

Silver et. al., 2016

Putting everything together

- We can think of the game of Go as a special type of graphical model.
  - Choose a move that has the highest chance of winning: argmax P(win | next_state)
  - We can run message passing algorithm to solve for this probability.

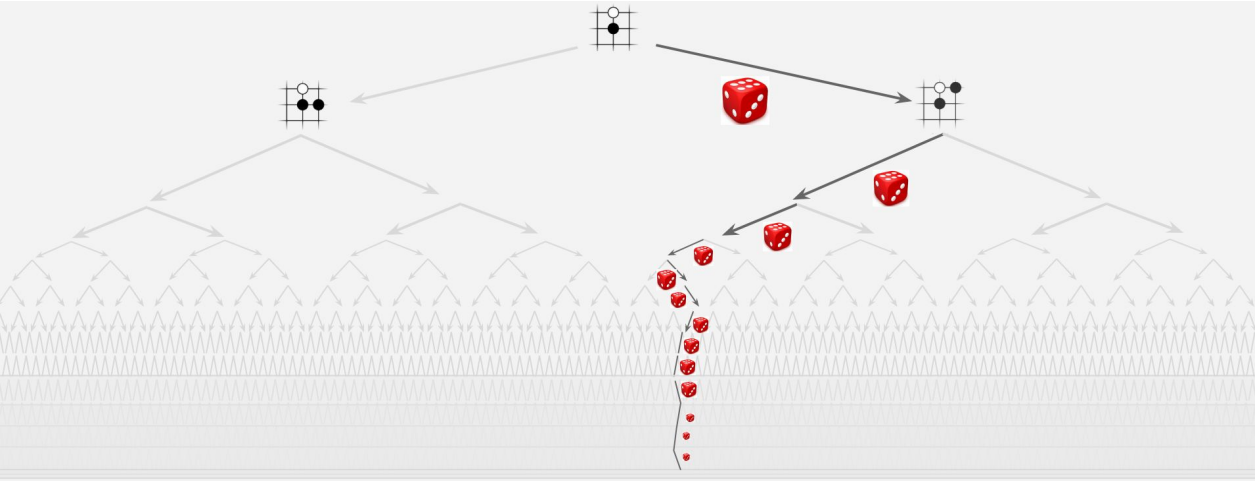


- The tree is too **wide**: too many branches at each node, which makes the summation over all those states infeasible.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Putting everything together

- We can think of the game of Go as a special type of graphical model.
  - Monte-Carlo rollouts can reduce the breath of the tree.
  - It does not help much if the prior distribution is bad.

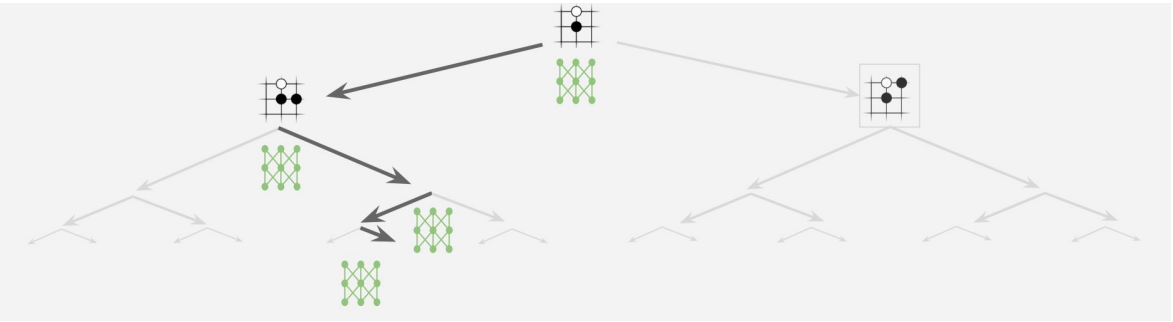


- The tree is too **wide**: too many branches at each node, which makes the summation over all those states infeasible.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Putting everything together

- We can think of the game of Go as a special type of graphical model.
  - Monte-Carlo rollouts + neural network learnt on expert moves, i.e. policy network
  - The policy network helps MC rollouts to not waste computational resources on “**bad**” moves.

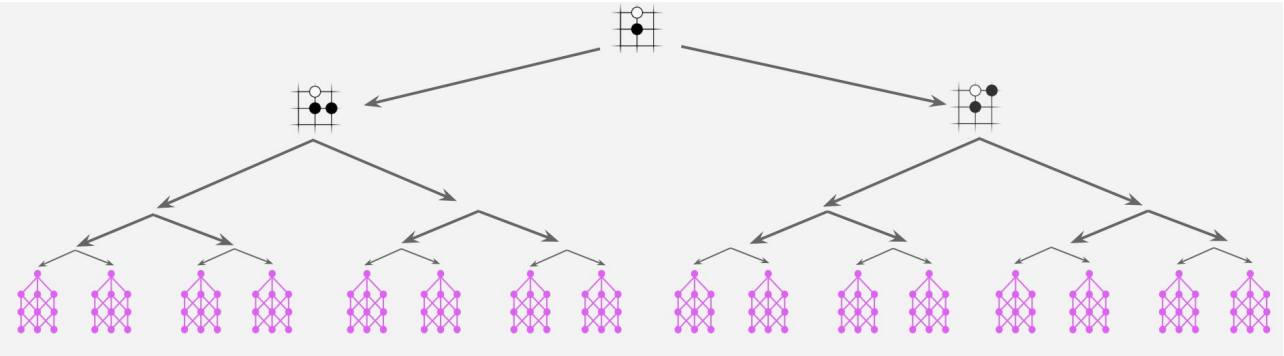


- policy network cut down the **breath** of the search tree.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Putting everything together

- We may not want to pass the messages all the way to the leaves of the tree.
  - Use a neural network to **approximate** the initial condition, i.e. value network
  - The value network learns the probability of winning at each node of the tree.

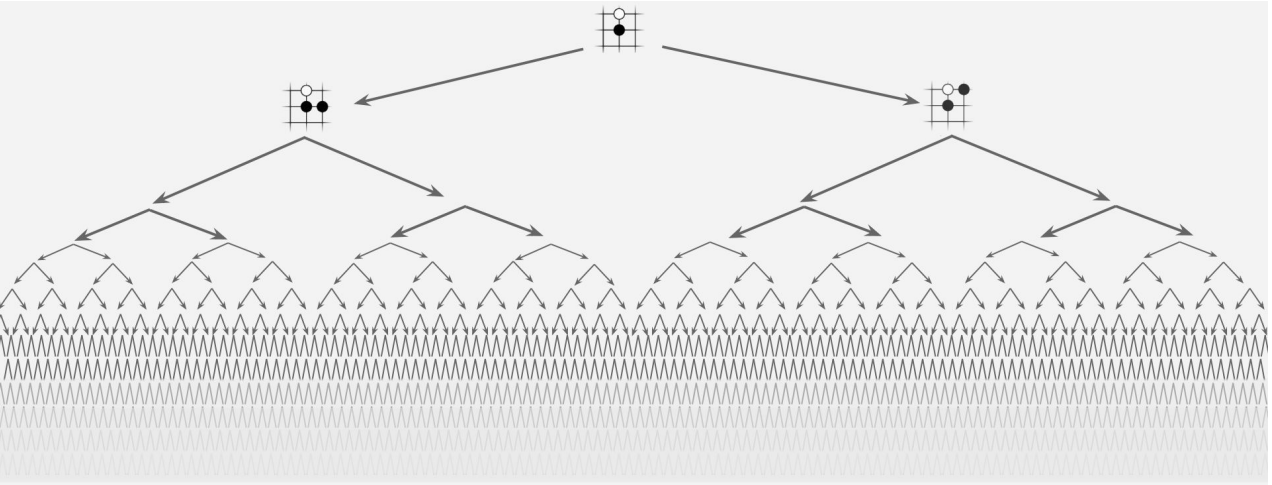


- policy network cut down the **breath** of the search tree.
- Value network cut down the **depth** of the search tree.

Silver et. al., 2016

Putting everything together

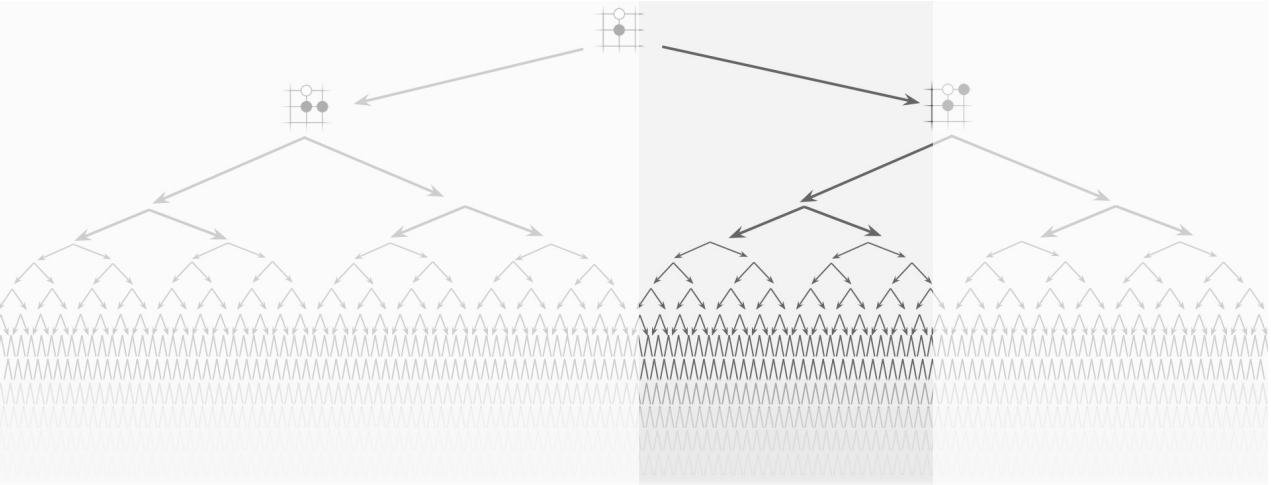
- Use both policy and value networks to significantly reduce the inference computation.



Silver et. al., 2016

Putting everything together

- Use both policy and value networks to significantly reduce the inference computation.

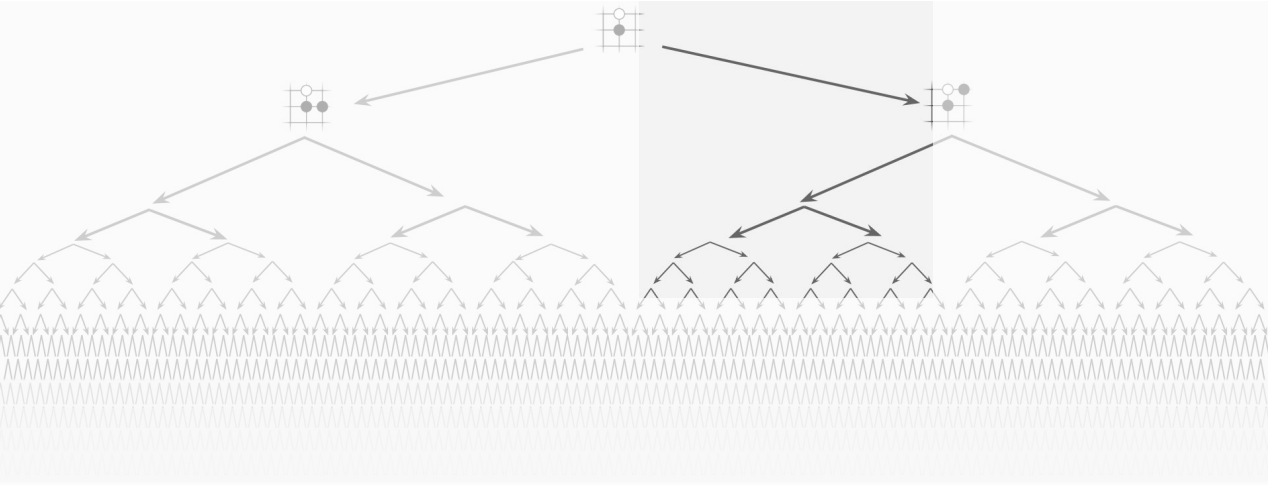


- policy network cut down the **breath** of the search tree.

Silver et. al., 2016

Putting everything together

- Use both policy and value networks to significantly reduce the inference computation.



- policy network cut down the **breath** of the search tree.
- Value network cut down the **depth** of the search tree.

Silver et. al., 2016