Review of CS486/686

Wenhu Chen, Pascal Poupart

Lecture 24

Outline

Search Algorithm

Uncertainty Estimation

Machine Learning

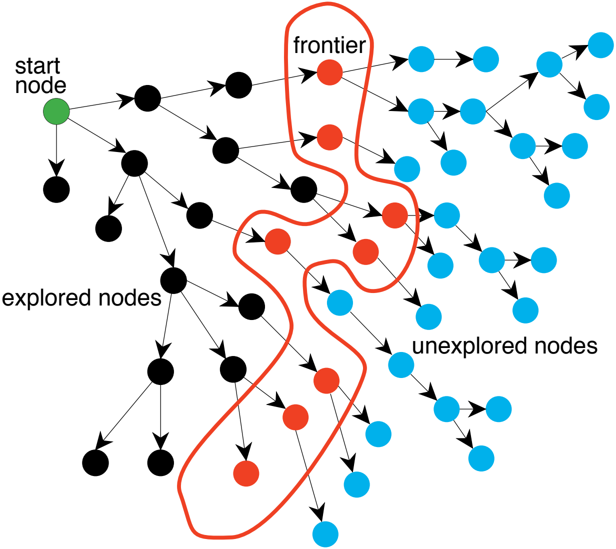
Deep Learning

Miscellaneous

Uninformed Search

- How to formulate a search problem?
- What is a search tree?
- What is generic Search algorithm?
- What is DFS and what is BFS?
- What is the space/time complexity of DFS and BFS?
- What is the iterative deepening space complexity?

The Search Tree



Generic Search Algorithm

Algorithm 1 A Generic Search Algorithm

procedure SEARCH(Graph, Start node s, Goal test $goal(n)$)
    frontier := $\{\langle s \rangle\}$
    while frontier is not empty **do**
        select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from frontier
        if goal(n_k) **then**
            return $\langle n_0, \ldots, n_k \rangle$
        for every neighbour n of n_k **do**
            add $\langle n_0, \ldots, n_k, n \rangle$ to frontier
    return no solution

How to use heuristic search?

- What is LCFS (lowest-cost first)?
- What is GBFS (lowest-heuristic first)?
- What is A* search (combination of two)?

A* Search Algorithm

- Space and Time Complexities.
- Completeness and Optimality.
- Admissible Heuristics $\rightarrow$ Optimality
- Consistent Heuristics $\rightarrow$ Multi-Path Pruning
- Prove the admissibility and consistency criterion

A* is Optimal with admissibility constraint

- Assuming you have many paths in the frontier: $(S \to G : C^*, \cdots, S \to N : C^n)$, and $C^* \leq C^n$.
- If there a path through N to G has a lower cost of $C' < C^*$.
- According to admissibility, $C^n \leq C' < C^*$.
- It's contradictory to our assumption.

Summary of Search Strategies

Strategy	Frontier Selection	Halts?	Space	Time
Depth-first	Last node added	No	Linear	Exp
Breadth-first	First node added	Yes	Exp	Exp
Lowest-cost-first	min $cost(n)$	Yes	Exp	Exp
Greedy Best-first	min $h(n)$	No	Exp	Exp
A*	min $cost(n) + h(n)$	Yes	Exp	Exp

Constraint Satisfaction Problem

- Why do we need to model the internal structure of the state?
- What is Backtracking Algorithm?
- What is Arc consistency Algorithm?
- AC-3 Algorithm, using Arc consistency to eliminate Arc
- AC-3 Algorithm complexity
- Combine Backtracking with AC-3 algorithm

Backtracking Search

Algorithm 2 BACKTRACK(assignment, csp)

```
if assignment is complete then return assignment
Let var be an unassigned variable
for every value in the domain of var do
    if adding {var = value} satisfies every constraint then
        add {var = value} to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ failure then return result
    remove {var = value} from assignment if it was added
return failure
```

The AC-3 Arc Consistency Algorithm

Algorithm 3 The AC-3 Algorithm

put every arc in the set S.
while S is not empty **do**
    select and remove $\langle X, c(X,Y)\rangle$ from S
    remove every value in D_X that doesn't have a value in D_Y that satisfies the constraint $c(X,Y)$
    if D_X was reduced **then**
        if D_X is empty **then return** false
        for every $Z \neq Y$, add $\langle Z, c'(Z,X)\rangle$ to S
return true

Local Search

- Why do we need local search?
- How do we perform greedy descent?
- How can we avoid local minima?
- What is Simulated Annealing?
- What is Genetic Algorithm?

Simulated Annealing Algorithm

Algorithm 4 Simulated Annealing

current $\leftarrow$ initial-state
T $\leftarrow$ a large positive value
while $T > 0$ **do**
    next $\leftarrow$ a random neighbour of current
    $\Delta C \leftarrow$ cost(next) - cost(current)
    if $\Delta C < 0$ **then**
        current $\leftarrow$ next
    else
        current $\leftarrow$ next with probability $p = e^{\frac{-\Delta C}{T}}$
    decrease T
return current

Search Algorithm

Uncertainty Estimation

Machine Learning

Deep Learning

Miscellaneous

Independence

- What is unconditional independence?
- What is conditional independence?
- What is chain rule/product rule/sum rule/bayes rule?
- Universal appraoch to calculate a probability.

Independence

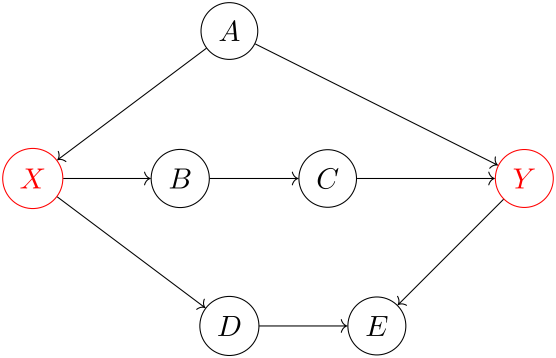
- Given joint probability distribution, derive the independence step by step.
- How are independence verified quantitatively.
- Why do we need to use Bayesian Networks?
- How can we compute joint probability over a Bayesian Network?

D-Separation

- What is D-Separation Rule 1?
- What is D-Separation Rule 2?
- What is D-Separation Rule 3?
- How do you apply these D-Sesparation rules to understand independence between different nodes?

D-Separation

- Un-directed paths between X and Y.
- Multiple paths need to be considered if they exist.
- One of the nodes on all the paths blocking the connection.



Constructing Bayesian Network

- Identify all the (conditional) independence relationships
- Pick an order
- Add nodes to the graph
- Pick the minimum subset as parents according to the (conditional) independence relationships
- Form a Bayesian Network

Search Algorithm

Uncertainty Estimation

Machine Learning

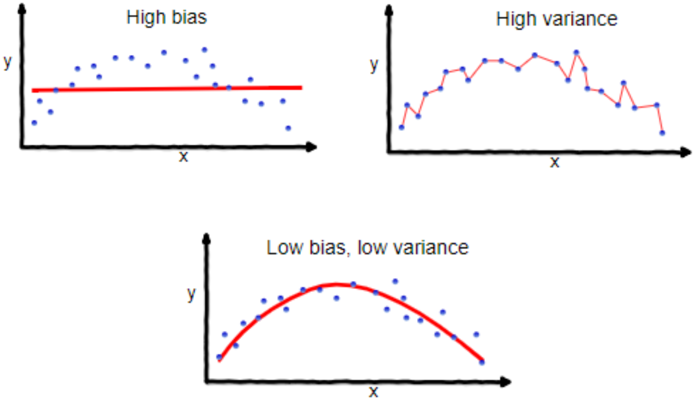
Deep Learning

Miscellaneous

Supervised Learning

- Classification vs. Regression
- Cross-Validation
- How to avoid Over-fitting?
- How to derive Bias-Variance equation?
- Trade-offs between bias and variance.

Bias-Variance Trade-off



Bias-Variance proof (1)

Let's denote: - Y as the true output. - X as the input feature. - $f(X)$ as the true function (which we aim to approximate). - $\hat{f}(X)$ as the predicted function from our model. - ϵ as the irreducible error or noise, with $\mathbb{E}[\epsilon] = 0$ and $\text{Var}(\epsilon) = \sigma^2$.

The true error is given by the expected value of the squared difference between the predicted value and the true value:

$$\text{True Error} = \mathbb{E}[(Y - \hat{f}(X))^2]$$

First, express Y in terms of the true function and noise:

$$Y = f(X) + \epsilon$$

Substitute this into the expression for the true error:

$$\text{True Error} = \mathbb{E}[(f(X) + \epsilon - \hat{f}(X))^2]$$

Expand the squared term:

$$\text{True Error} = \mathbb{E}[(f(X) - \hat{f}(X))^2 + 2(f(X) - \hat{f}(X))\epsilon + \epsilon^2]$$

Bias-Variance proof (2)

Since ϵ is noise with $\mathbb{E}[\epsilon] = 0$:

$$\mathbb{E}[2(f(X) - \hat{f}(X))\epsilon] = 2\mathbb{E}[(f(X) - \hat{f}(X))]\mathbb{E}[\epsilon] = 0$$

So, the true error simplifies to:

$$\text{True Error} = \mathbb{E}[(f(X) - \hat{f}(X))^2] + \mathbb{E}[\epsilon^2]$$

Since $\mathbb{E}[\epsilon^2] = \sigma^2$, we have:

$$\text{True Error} = \mathbb{E}[(f(X) - \hat{f}(X))^2] + \sigma^2$$

Now, decompose the term $\mathbb{E}[(f(X) - \hat{f}(X))^2]$:

$$\mathbb{E}[(f(X) - \hat{f}(X))^2] = \mathbb{E}[(f(X) - \mathbb{E}[\hat{f}(X)] + \mathbb{E}[\hat{f}(X)] - \hat{f}(X))^2]$$

Using the linearity of expectation:

$$= \mathbb{E}[(f(X) - \mathbb{E}[\hat{f}(X)])^2 + (\mathbb{E}[\hat{f}(X)] - \hat{f}(X))^2 + 2(f(X) - \mathbb{E}[\hat{f}(X)])(\mathbb{E}[\hat{f}(X)] - \hat{f}(X))]$$

Bias-Variance proof (3)

Since $\mathbb{E}[(f(X) - \mathbb{E}[\hat{f}(X)])(\mathbb{E}[\hat{f}(X)] - \hat{f}(X))] = 0$:

$$= \mathbb{E}[(f(X) - \mathbb{E}[\hat{f}(X)])^2] + \mathbb{E}[(\mathbb{E}[\hat{f}(X)] - \hat{f}(X))^2]$$

The first term is the **bias squared**:

$$\mathbb{E}[(f(X) - \mathbb{E}[\hat{f}(X)])^2] = (\text{Bias}[\hat{f}(X)])^2$$

The second term is the **variance**:

$$\mathbb{E}[(\mathbb{E}[\hat{f}(X)] - \hat{f}(X))^2] = \text{Var}[\hat{f}(X)]$$

Thus, we have:

$$\mathbb{E}[(f(X) - \hat{f}(X))^2] = (\text{Bias}[\hat{f}(X)])^2 + \text{Var}[\hat{f}(X)]$$

Therefore, the true error can be written as:

$$\text{True Error} = (\text{Bias}[\hat{f}(X)])^2 + \text{Var}[\hat{f}(X)] + \sigma^2$$

Neural Networks I

- Approximating Different Gate function with Neural Network.
- What is activation function and what qualifies as activation functions?

Neural Networks II

- What is gradient descent?
- What is loss function?
- What's the rule of Backward propagation?
- How to perform backpropagation on 1 or 2 layers of neural network?
- Understand the computation/memory complexity of backpropagation

The recursive relationship

Backward Propagation Algorithm:

- Initialize W_i for all the layers (from 1 to n).
- Feedforward x into neural network and save intermediate values $g(x^{(1)}), g(x^{(2)}), \cdots$.
- Compute $\delta_n = \frac{\partial E}{\partial z} \cdot \frac{\partial g(x^{(n)})}{\partial x^{(n)}}$.
- For i = n $\rightarrow$ 2; do
  - $\delta_{i-1} = \delta_i \cdot W_i^T \cdot \frac{\partial g(x^{(i-1)})}{\partial x^{(i-1)}}$
  - Compute $\frac{\partial E}{\partial W_i} = g(x^{i-1}) \otimes \delta_i$
- $\frac{\partial E}{\partial W_1} = x^0 \otimes \delta_1$, where x^0 is the input.
- Obtain all $\frac{\partial E}{\partial W_i}$ for gradient descent.

Neural Networks III

- What is stochastic gradient descent?
- What is momentum method?
- What is adaptive method?
- Understand how to use Adam optimizer.

Adam: ADAptive Moments

Algorithm 5 ADAptive Moments

Require: Learning Rate ϵ, Decay rates $\rho_1, \rho_2, \theta, \delta$

1: Initialize $\boldsymbol{s} = 0$, $\boldsymbol{r} = 0$, time step $t = 0$
2: **while** stopping criteria not met **do**
3: Sample example $(x^{(i)}, y^{(i)})$ from training set
4: Compute gradient estimate: $\hat{g} \leftarrow +\nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$
5: $t \leftarrow t + 1$
6: Update: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\hat{g}$
7: Update: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\hat{g} \odot \hat{g}$
8: Correct Biases: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}, \hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
9: Compute Update: $\Delta\theta = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$
10: Apply Update: $\theta \leftarrow \theta + \Delta\theta$

Search Algorithm

Uncertainty Estimation

Machine Learning

Deep Learning

Miscellaneous

Unsupervised Learning

- How to do K-means clustering?
- What is Principled component Analysis?
- Understanding the two algorithms of PCA.
- Draw the connection between the two algorithms of PCA.
- What is auto-encoder?
- What is the optimization goal of GANs?

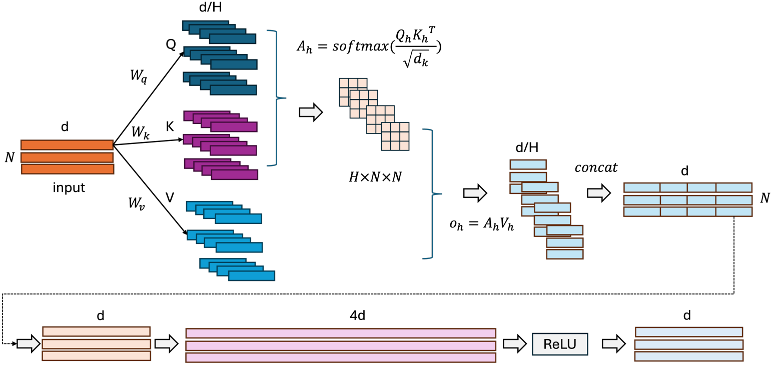
CNN & RNNs

- Understand RNN architecture.
- Perform backward propagation through time with two steps.
- Understand CNN architecture.
- Perform backward propagation on 2x2 kernel on 3x3 image.

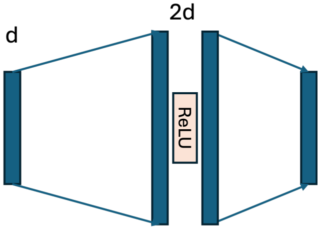
Transformers

- Understand the motivation of self-attention.
- Understand the difference between self-attention and cross-attention.
- Understand the computation flow of a single-layer self-attention.
- Understand the computation and memory complexity of single-layer self-attention.

Self-Attention Layer



Neural Networks Computation Complexity (Forward)



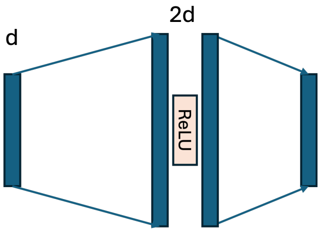
- Computation cost is mostly coming from matrix multiplication. Activations have negligible computation.
- In the forward propagation, there will be Bxdx2d and Bx2dxd matrix multiplications.
- Total cost is $4Bd^2$.

Neural Networks Computation Complexity (Backward)

Assuming we have one layer of d by d' dimensions (weight matrix of W), where the input is h_t and the output is h_{t+1}.

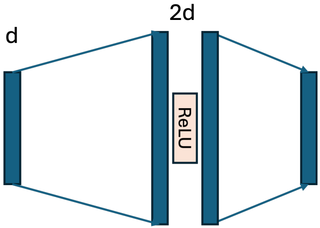
- We need $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_{t+1}} \cdot h_t$. This operation consists of a matrix multiplication of $\mathbb{R}^{B \times d' \times 1}$ and $\mathbb{R}^{B \times 1 \times d}$, which consumes Bdd' multiplication.
- We also need to use chain rule to compute $\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \cdot W$. This operation consists of a matrix multiplication of $\mathbb{R}^{B \times d'}$ and $\mathbb{R}^{d' \times d}$, which consumes another Bdd' multiplication.
- The forward is pass is Bdd', then the backward is roughly doubling it to $2Bdd'$. Therefore, in order to know the backward computation, then we can just double that.
- In the previous network, the total will be $8Bd^2$.

Neural Networks Memory Complexity (Forward)



- Memory complexity is the peak memory we use.
- In the forward scenario, let's say that we don't do backprop and we don't want to cache anything.
- We can implement everything in-place. The memory cost is basically the largest intermediate. It's 2Bd in this case.
- Model weights have to stay in the memory, therefore, the total is $4d^2 + 2Bd$.

Neural Networks Memory Complexity (Backward)



- We need to cache every hidden vectors and input and output in the neural network. This is for preparing backward propagation. Therefore, the total is Bd + B2d + B2d + Bd = 6Bd.
- We need to store all the weights in the memory, which is $4d^2$.
- If we are using Adam, there will first-order and second-order momentum, which are exactly the same size of weight $8d^2$.
- The total is $12d^2 + 6Bd$.

Exam

- Non-programmable calculator.
- Single-sided A4 Study Note.
- A total of 8 problems.
- 5 problems from Wenhu.
- 3 problems from Pascal.
- 100 marks in total.
- If you can't attend the final exam, we will mark as INC to allow you to take the final in the following term.
- Pass the exam to pass the course.

Lecture 24: Other AI Courses
CS486/686 Intro to Artificial Intelligence

Pascal Poupart
David R. Cheriton School of Computer Science
CIFAR AI Chair at Vector Institute

UNIVERSITY OF WATERLOO

Other AI Courses

- CS480/680: Intro to Machine Learning
  - Support vector machines, logistic regression, Gaussian processes, linear regression, CNNs, RNNs, transformers, variational autoencoders, generative adversarial networks, graph neural networks, normalizing flows, diffusion models, bagging/boosting, transfer learning, fairness, etc.
- CS485/685: Learning theory
- CS484/684: Computer vision
- CS479: Biologically plausible neural networks
- CS794: Optimization for Data Science
- CS885: Reinforcement Learning (Winter 2025, instructor: Pascal Poupart)
- CS886: Advanced topics in AI
  - Graph neural networks, NLP, Vision, multiagent systems, robust ML, learning theory