CS 583: Probabilistic Graphical Models

Topic: Learning - Overview

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

WHY LEARN?

- Alternative
  - Elicit the structure and the parameters from experts
- Problems
  - Knowledge might not exist or might be insufficient
  - Might take weeks/months
  - The model might be different for different places
    - A model in the US vs. Europe might be different
  - The model might change over time
- It is, however, often easier to collect data
  - E.g., patient records are often available

P*

- Assume the domain is governed by P^*
- A network model $\mathcal{M}^*=(\mathcal{K}^*, \theta^*)$
- We are given a dataset $\mathcal{D} = \{x[1], \ldots x[n]\}$
  - n samples from P^*
- We typically assume that the data is *IID*
  - *Independent and identically distributed*
- We can learn
  - Parameters for a given structure
  - Parameters and the structure of the network

Ideally

- Ideally, we want to learn $\mathcal{M}$ that captures P^* precisely
- Unfortunately, this goal is often not achievable due to
  - Computational reasons, but more importantly,
  - Limited data
- Examples:
  - You want to estimate P(Heads), and you want to see each heads and each tails at least 100 times
    - How much data (in expectation) do you need?
  - You have k binary variables. You want to see the least likely case at least 100 times. How much data (in expectation) do you need in the best case where each case is uniformly distributed?

Goals of learning

- Because we have to learn approximate models, we have to define the goal of learning
  - One approximate model for a purpose might not be the best approximate model for another purpose
- Goals
  1. Density estimation
  2. Specific prediction tasks
  3. Knowledge discovery

1. Density estimation

- Goal: Use the network for various inference tasks
  - E.g., missing data prediction
- Learn $\tilde{\mathcal{M}}$ such that $\tilde{P}$ is as close as possible to P^*
- How do evaluate the quality of $\tilde{\mathcal{M}}$? That is, how do we define close?
- Relative entropy/Kullback-Liebler (KL) divergence

$$KLD(P^* \| \tilde{P}) = E_{x \sim P^*}\left[\log\left(\frac{P^*(x)}{\tilde{P}(x)}\right)\right]$$

KL-Divergence

$$KLD(P^* \| \tilde{P}) = E_{x \sim P^*}\left[\log\left(\frac{P^*(x)}{\tilde{P}(x)}\right)\right]$$

- We need to know *P**
- *P** is not known in real-world settings

$$KLD(P^* \| \tilde{P}) = -H_{P^*}(\mathrm{X}) - E_{x \sim P^*}\left[\log\left(\tilde{P}(x)\right)\right]$$

Proof?

EXPECTED LOG-LIKELIHOOD

$$KLD(P^* \| \tilde{P}) = -H_{P^*}(\mathrm{X}) - E_{x \sim P^*}\left[\log\left(\tilde{P}(x)\right)\right]$$

- To find $\tilde{\mathrm{P}}$ to minimize *KLD*($\mathrm{P}^* \mid \tilde{\mathrm{P}}$), we can maximize the second term $\mathrm{E}_{\mathrm{P}*}[\log(\tilde{\mathrm{P}})]$
- $\mathrm{E}_{\mathrm{P}*}[\log(\tilde{\mathrm{P}})]$ is also called *expected log-likelihood*

Likelihood/Log-likelihood

- Likelihood of the data given model $\mathcal{M}$
  - $P(\mathcal{D} : \mathcal{M})$

$$P(\mathcal{D}:\mathcal{M}) = \prod_{i=1}^{n} P\left(x[i]:\mathcal{M}\right)$$

- Log-likelihood of the data given model $\mathcal{M}$:
  - $\log P(\mathcal{D} : \mathcal{M}))$

$$\log P(\mathcal{D}:\mathcal{M}) = \sum_{i=1}^{n} \log P\left(x[i]:\mathcal{M}\right)$$

2. Specific prediction tasks

- We are interested in predicting a set of variables $\boldsymbol{Y}$ given another set $\boldsymbol{X}$
  - We are interested in $P(\boldsymbol{Y} \mid \boldsymbol{X})$
- Expected conditional log-likelihood

$$E_{(\boldsymbol{x},\boldsymbol{y})}\left[\log\left(\tilde{P}(\boldsymbol{y} \mid \boldsymbol{x})\right)\right]$$

- Like log-likelihood, except, we don't require our model to provide a distribution over only $\boldsymbol{X}$

3. Knowledge discovery

- We would like to discover knowledge about P^*
- E.g.,
  - Direct and indirect dependencies
  - Nature of the dependencies (positive/negative correlation)
- Goal is not to model just P^* but it is to extract $\mathcal{M}^*$
- The correct model might not be identifiable
  - Remember that given a Bayesian network structure, there might be many I-Equivalent versions of it
- Inferring weak edges requires a lot of data
- Inferring the lack of edges requires a lot of data

Learning as optimization

- We define a criterion that we want to optimize
  - Log-likelihood, conditional log-likelihood, etc.
- We have a hypothesis space
  - A set of candidate models
- An objective function
  - Quantifies our preferences over candidates
- Learning = finding a high-scoring model within the hypothesis space

EMPIRICAL RISK

- We don't have access to P*
- We can instead construct an empirical distribution $\tilde{P}$ using the data training data $\mathcal{D}$
  - $\tilde{P}(x) = \text{Count}(x \in \mathcal{D}) \,/\, |\mathcal{D}|$
- Let's analyze $\tilde{P}$
  - What's good about it?
    - It is the distribution with the highest log-likelihood on $\mathcal{D}$
      - Make sure you see that
  - What's wrong about it?

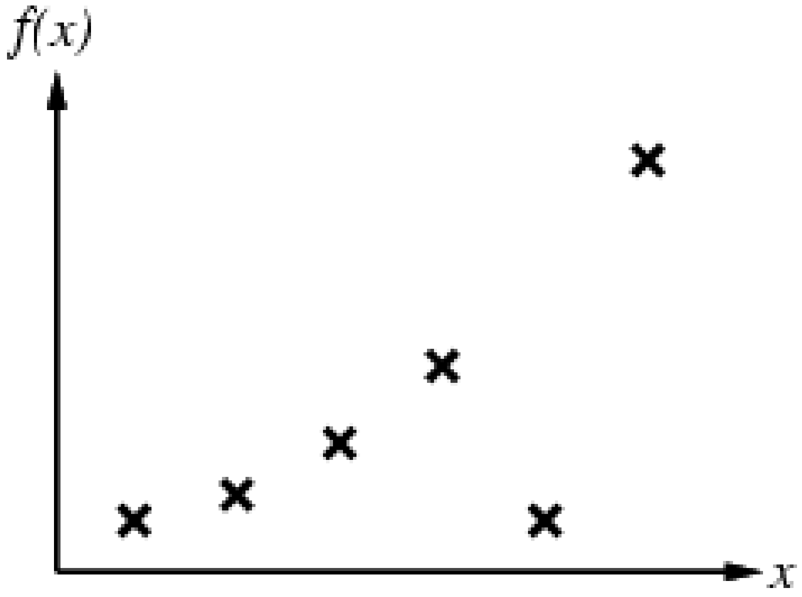
Overfitting / Generalization

- $\tilde{P}$ *overfits* the data: it doesn't capture the regularities, rather it captures every bit of detail
  - If we had an infinite mount of data (or close to it), then this would not be a problem, but
  - In limited data, the details can be accidental
- We want our model to *generalize* instead
  - Be able to perform well with unseen data

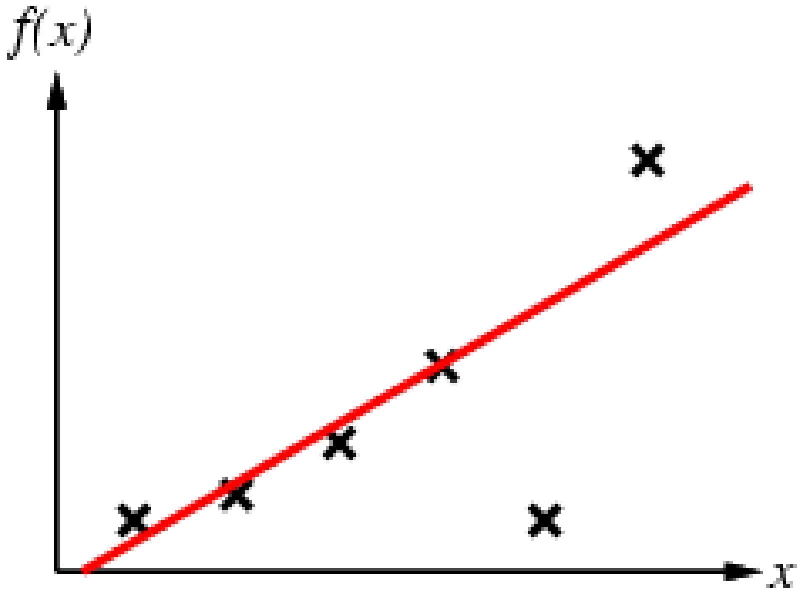
Bias/Variance trade-off

- If we choose a simple hypothesis class
  - E.g., if we assume every variable is marginally independent, no matter how much data, we might not be able to capture P*
  - High *bias*
- If we choose a complex hypothesis class
  - E.g., if we assume every variable is connected to every other variable, then there won't be enough data to estimate the parameters correctly. Moreover, the estimated parameters will vary drastically if the training data is only slightly different
  - High *variance*
- We need to strike a balance between *bias* and *variance* to achieve better *generalization*

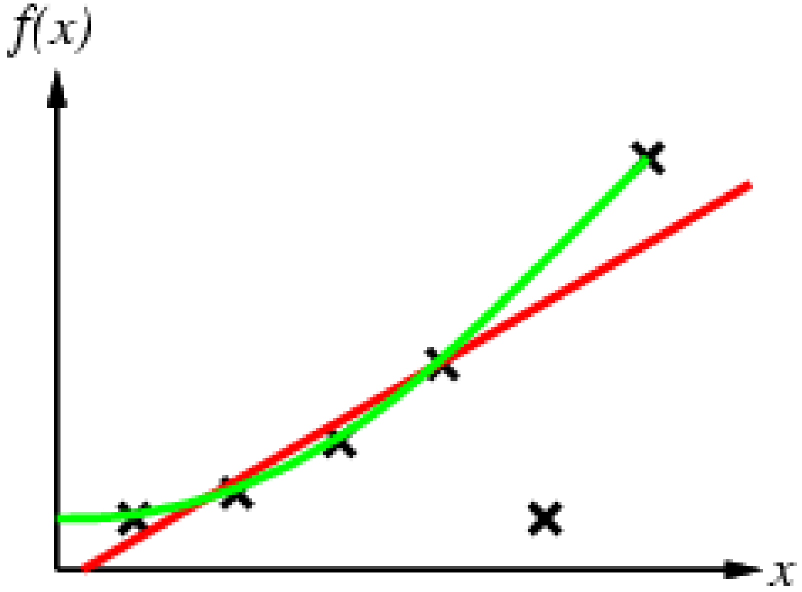
CURVE FITTING



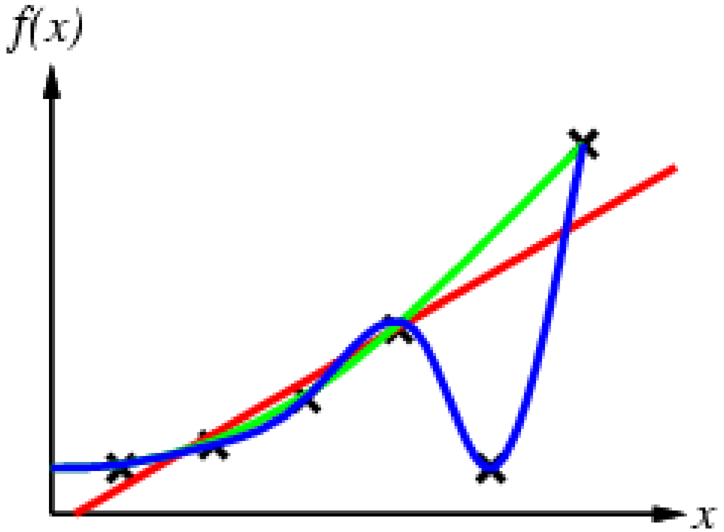
Curve Fitting



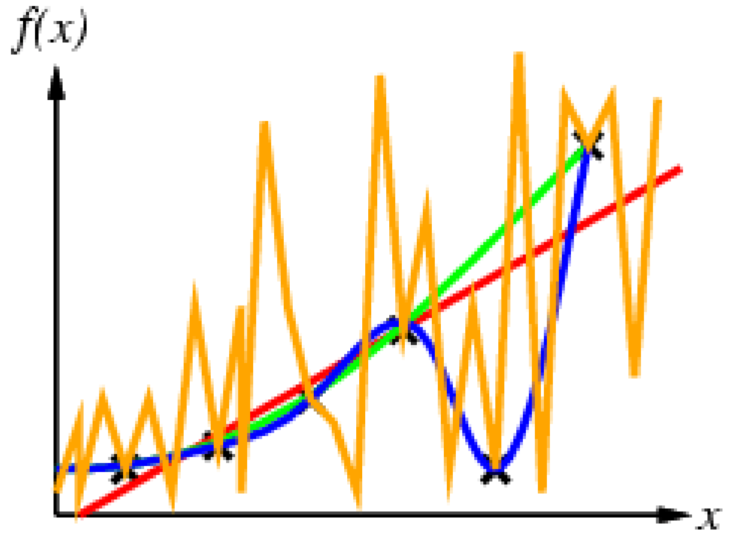
Curve Fitting



CURVE FITTING



Curve Fitting



EVALUATING GENERALIZATION PERFORMANCE

- *Hold-out testing*
  - Split the data into training set and test set
  - Learn on the training set, evaluate on the test set
  - Need large data
- *Cross-validation*
  - Split the data into k
  - Each time, every split gets a turn to be the test set while the others act as the training set