ORIE 4741: Learning with Big Messy Data

Looking backward, looking forward

Professor Udell

Operations Research and Information Engineering
Cornell

December 7, 2021

Announcements 12/7/21

- homework 6 due 9:15am Tuesday 12/7/21
- project peer review due Sunday 12/12/21 11:59pm
- extra credit: $+1\%$ for filling out course evaluation

Poll

Next year, should this class have

- quizzes
- a midterm and final exam

Poll

Next year, should this class have

- more homeworks
- the same number of homeworks
- fewer homeworks

Poll

Next year, should this class have

- shorter homeworks
- longer (and fewer) homeworks

Poll

Next year, should this class have a homework drop (in addition to slip days)

- yes
- no

Poll

Next year, should this class be offered

- hybrid sync
- hybrid async (like this year)
- only in-person

Outline

Learning Big Messy Data

How to study (if there were an exam)

What did we not talk about?

Review

- learning
- big
- messy
- data

Review: data

- first of all: look at it!
- are there missing values?
- decide what you want to learn or predict
- input space $\mathcal{X}$, output space $\mathcal{Y}$
  - real, boolean, nominal, ordinal, text, ...

Review: messy

- probabilistic model: $(x, y) \sim P(x, y)$
- deterministic model: $y = f(x)$
- additive noisy model: $y = f(x) + \varepsilon$
  - additive noise model makes no sense for non-real data types (boolean, ordinal, nominal)

Review: messy features

- feature engineering
  - can convert other data to real valued features
  - allows linear models to handle nonlinear data
- unsupervised learning
  - can use to fill in missing data
  - can use to reduce dimensionality of feature space
- regularization
  - can reduce sensitivity of estimate to corrupted or novel feature values
  - can perform feature selection

Feature engineering

examples:

- adding offset
- standardizing features
- polynomial fits
- missing values: imputation, boolean indicator
- nonlinear transformations
- autoregressive models
- transforming Booleans, ordinals, nominals
- pretrained neural nets for images and text
- decision trees
- all of the above

`https://xkcd.com/2048/`

Review: messy labels

- robust loss functions
  - huber, ℓ_1, quantile
- loss functions for classification
  - hinge, logistic
- loss functions for ordinals and nominals
  - by learning a vector
  - by learning probabilities
- how to impute data in correct domain (eg, ordinal)

Review: messy labels

- robust loss functions
  - huber, ℓ_1, quantile
- loss functions for classification
  - hinge, logistic
- loss functions for ordinals and nominals
  - by learning a vector
  - by learning probabilities
- how to impute data in correct domain (eg, ordinal)
  - use the loss function! $\hat{y} = \operatorname{argmin}_a \ell(x^T w, a)$

Review: learning

- view data as samples from $P(x, y)$
- goal is to learn $f : \mathcal{X} \to \mathcal{Y}$
- complex models fit both data and noise better
- diagnose underfitting vs overfitting
- generalization: how do we know if we're overfitting?
  - bootstrap: how big are the error bars?
  - (cross)validate: how big are the out-of-sample errors?
  - compute error on test set + use Hoeffding bound
  - posit a probabilistic model + use bias variance tradeoff
- correcting overfitting: regularize or add data
- correcting underfitting: add new (engineered or measured) features

Review: learning a mapping

how to learn $f : \mathcal{X} \to \mathcal{Y}$?

- using a greedy iterative procedure, like the **perceptron** method or **decision trees**
- by **regularized empirical risk minimization**
  - minimizing some **loss function** to fit the data
  - $+$ some **regularizer** to generalize well

Review: learning by optimizing

how to solve an optimization problem:

- is the objective quadratic?
  - set the gradient to 0; solve resulting system of equations
- is the objective (sub)differentiable?
  - use gradient descent
- do you have lots of data?
  - use stochastic gradients

Review: big

- algorithms for big data should be **linear** in the number of samples n
- three big data algorithms for least squares:
  - gradient descent ($O(nd)$ per iteration)
  - QR ($O(nd^2)$)
  - SVD ($O(nd^2)$) (mostly used as analysis tool)
- decision trees ($O(nd\log(d))$)
- parallelize!
  - gradient calculation
  - Gram matrix calculation

Review: unsupervised

- use unsupervised learning to
  - reduce dimensionality
  - reduce noise in data
  - fill in missing entries
  - plot or cluster messy high-dimensional features
  - find vector representation for nominal features
- Generalized low rank models
  - includes PCA, non-negative matrix factorization, matrix completion, ...
  - can fit using alternating proximal gradient method

Review: interpretability

- what kinds of models are interpretable? (justify)

Review: interpretability

- what kinds of models are interpretable? (justify)
    - linear models with few features
    - decision trees
    - generalized additive models
- what kinds of models are not interpretable? (justify)

Review: interpretability

- what kinds of models are interpretable? (justify)
  - linear models with few features
  - decision trees
  - generalized additive models
- what kinds of models are not interpretable? (justify)
  - deep neural nets
  - random forests
- why use an interpretable model?

Review: interpretability

- what kinds of models are interpretable? (justify)
  - linear models with few features
  - decision trees
  - generalized additive models
- what kinds of models are not interpretable? (justify)
  - deep neural nets
  - random forests
- why use an interpretable model?
  - facilitate use of model
  - identify biases in data
  - (possible) causal insights

Review: limits of big data

- does your data have any systematic biases? could you tell?
- what are the human consequences if your model is right or wrong?
- is your model a Weapon of Math Destruction (WMD)?
  - are outcomes hard to measure?
  - could its predictions harm anyone?
  - could it create a feedback loop?
- is the current decision making procedure a WMD?

Review: fairness

- why is algorithmic fairness important?

Review: fairness

- why is algorithmic fairness important?
- fair for whom? **protected attributes**

Review: fairness

- why is algorithmic fairness important?
- fair for whom? **protected attributes**
- why might a learned model be unfair even if the protected attribute is not in the model?
  - proxies

Review: fairness

- why is algorithmic fairness important?
- fair for whom? **protected attributes**
- why might a learned model be unfair even if the protected attribute is not in the model?
  - proxies
- what might we mean by unfairness? which definition is appropriate (for a given application)?
  - Unawareness / anti-classification
  - Demographic parity
  - Equalized odds, Equality of opportunity
  - Predictive Rate Parity
  - Individual Fairness
  - Counterfactual fairness

Outline

Learning Big Messy Data

How to study (if there were an exam)

What did we not talk about?

How to study for the exam

- reread your notes and the lecture slides

How to study for the exam

- reread your notes and the lecture slides
- reread homework questions and solutions

How to study for the exam

- reread your notes and the lecture slides
- reread homework questions and solutions
- make up exam questions
  - what do you want to remember from this class?
  - what will you need to remember to learn from big messy data effectively?
  - what will you need to remember to learn from big messy data without learning things that are false?

How to study for the exam

- reread your notes and the lecture slides
- reread homework questions and solutions
- make up exam questions
  - what do you want to remember from this class?
  - what will you need to remember to learn from big messy data effectively?
  - what will you need to remember to learn from big messy data without learning things that are false?
- ask when. *e.g.*,
  - when to use a decision tree vs linear model?
  - when should we use different losses and regularizers?
  - when to use word embeddings vs bag of words?
  - when should we cross validate?
  - when to bootstrap?
  - when to use PCA?
  - when would you want sparsity?

How to study for the exam

- reread your notes and the lecture slides
- reread homework questions and solutions
- make up exam questions
  - what do you want to remember from this class?
  - what will you need to remember to learn from big messy data effectively?
  - what will you need to remember to learn from big messy data without learning things that are false?
- ask when. *e.g.*,
  - when to use a decision tree vs linear model?
  - when should we use different losses and regularizers?
  - when to use word embeddings vs bag of words?
  - when should we cross validate?
  - when to bootstrap?
  - when to use PCA?
  - when would you want sparsity?
- then ask **how** to do these and **why** these methods work.

How to study for the exam

- reread your notes and the lecture slides
- reread homework questions and solutions
- make up exam questions
  - what do you want to remember from this class?
  - what will you need to remember to learn from big messy data effectively?
  - what will you need to remember to learn from big messy data without learning things that are false?
- ask when. *e.g.*,
  - when to use a decision tree vs linear model?
  - when should we use different losses and regularizers?
  - when to use word embeddings vs bag of words?
  - when should we cross validate?
  - when to bootstrap?
  - when to use PCA?
  - when would you want sparsity?
- then ask **how** to do these and **why** these methods work.

do these **out loud, with a friend**

Outline

Learning Big Messy Data

How to study (if there were an exam)

What did we not talk about?

What did we not talk about? I

more learning techniques

- nearest neighbors and exemplar methods
- decision trees, random forests (bagging), gradient boosted trees (boosting)
- fancier neural nets: frameworks (TensorFlow and PyTorch), convolution, transformers

resources:

- CS machine learning class
- Hastie, Tibshirani and Friedman, "Elements of Statistical Learning"
- distributed ML frameworks: TensorFlow, PyTorch, H2O, MLLib, ...
- ML conferences: NeurIPS, ICML, AAAI, KDD, FAccT, ...

What did we not talk about? II

more optimization techniques

- solvers for non-differentiable problems like LASSO, *e.g.*, proximal gradient
- more stable, reliable methods, *e.g.*, interior point
- more parallel distributed methods, *e.g.*, ADMM
- other iterative methods for huge data, *e.g.*, ADAM, AdaGrad

resources:

- ORIE optimization classes
- CS machine learning classes

What did we not talk about? III

more domain specific techniques

- text
- images, video
- time series
- signal processing
- speech

Questions?

Questions?

my question for you:

- what other topics should we cover next year?