10-607 Computational Foundations for Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Course Overview

+

Propositional Logic

Matt Gormley
Lecture 1
Oct. 22, 2018

ABOUT THIS COURSE

How to describe 606/607 to a friend

606/607 is…

a **formal** presentation of **mathematics** and **computer science**…

motivated by (carefully chosen) **real-world problems** that arise in **machine learning**…

where the **broader picture** of how those problems arise is treated **somewhat informally.**

How to describe 606/607 to a friend

606/607 is…

The class you should take to prepare for future coursework in machine learning

Machine Learning is NOT a self-driving car

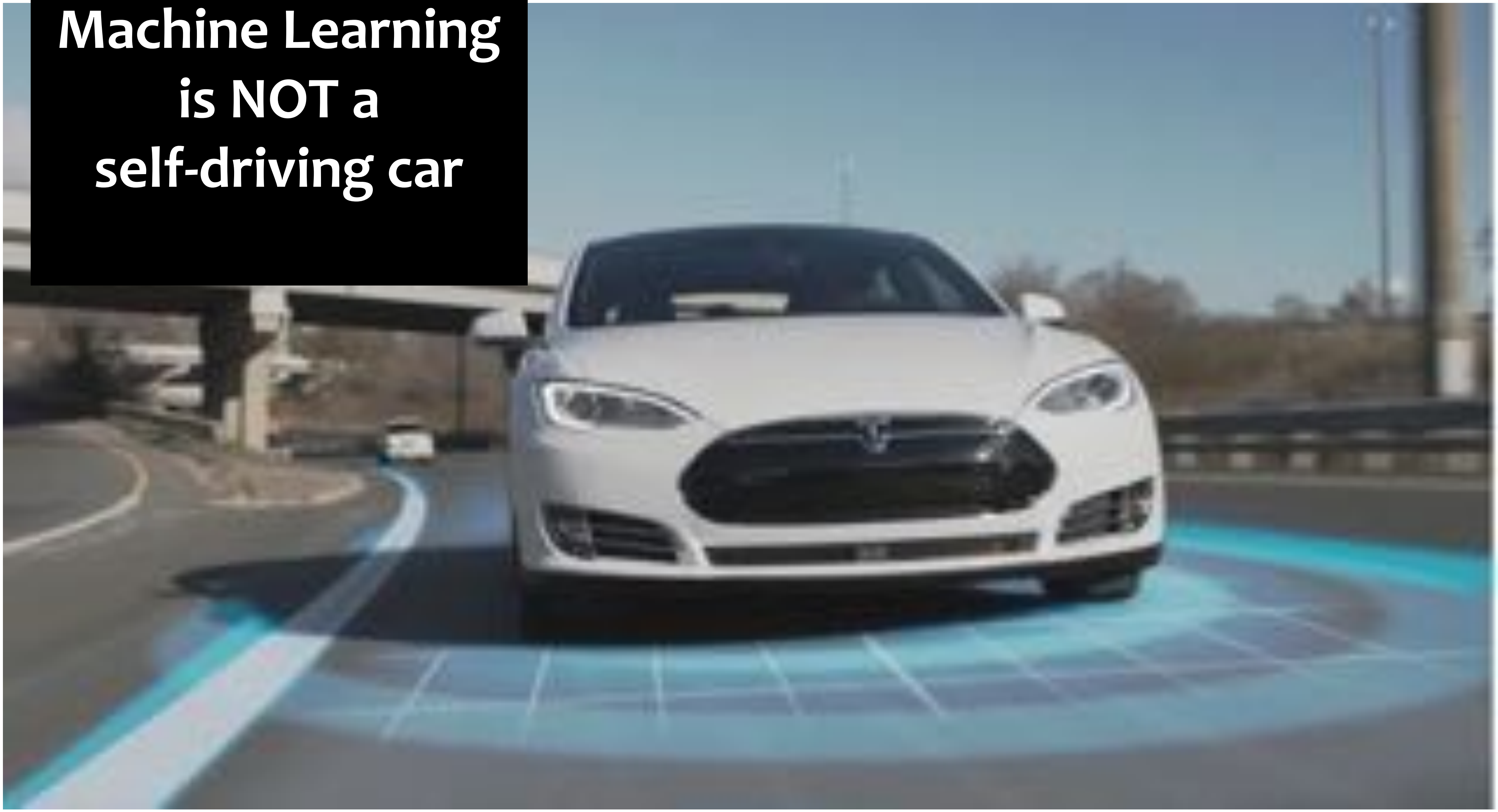


Image from https://www.greencarreports.com/

Reorganization of Today's Lecture

- **10-606 Students:**
  - Don't worry! We're going to leave the entire next section (course overview, syllabus until the end)
  - That way, you can leave class early if you don't want to hear it all over again
- **Non-10-606 Students:**
  - We'll save the big introduction until later and jump right into some core content
  - Save your questions about course orginization until later

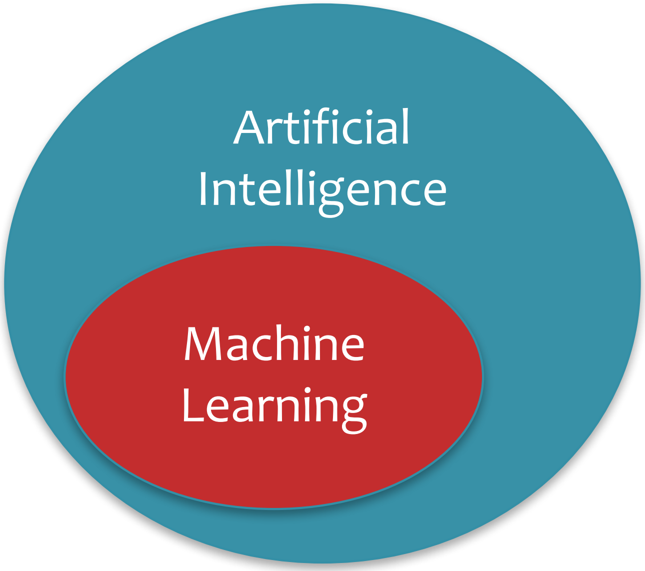
WHAT IS MACHINE LEARNING?

Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

- Perception
- Reasoning
- Control / Motion / Manipulation
- Planning
- Communication
- Creativity
- Learning



What is Machine Learning?

The goal of this course is to provide you with a toolbox:

Machine Learning

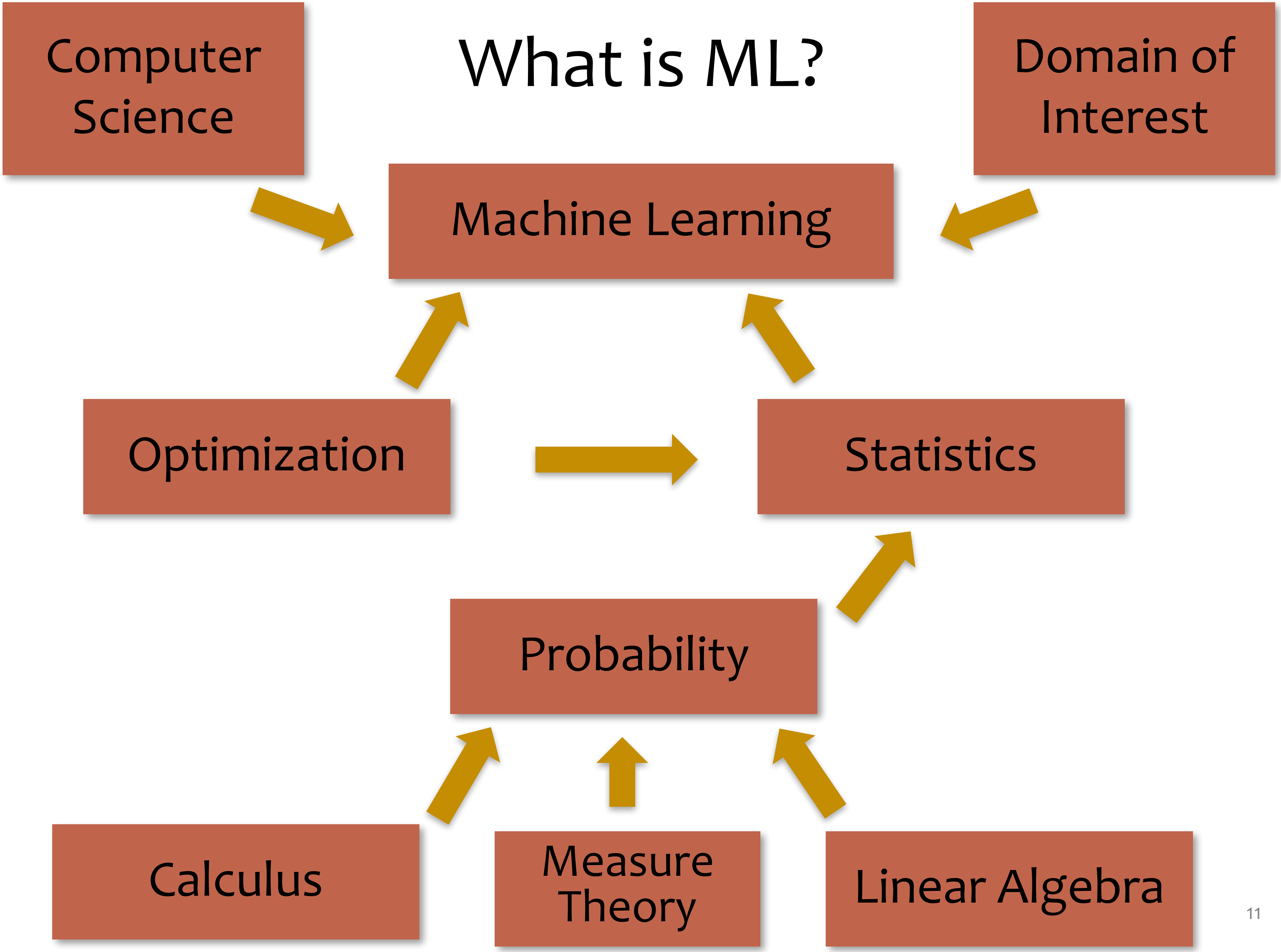
Statistics

Probability

Computer Science

Optimization

What is ML?



Computer Science

Domain of Interest

Machine Learning

Optimization

Statistics

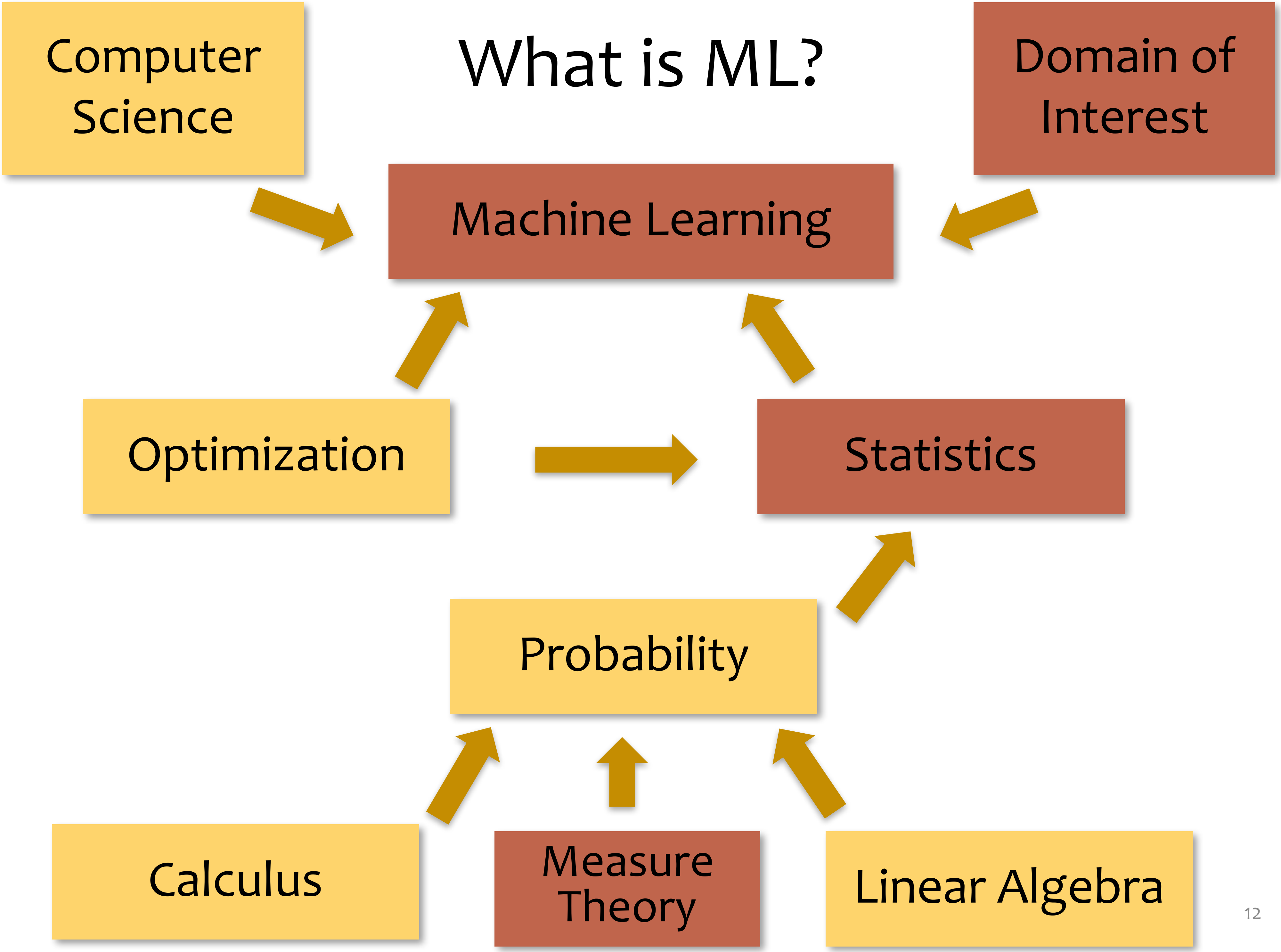
Probability

Calculus

Measure Theory

Linear Algebra

What is ML?



Computer Science

Domain of Interest

Machine Learning

Optimization

Statistics

Probability

Calculus

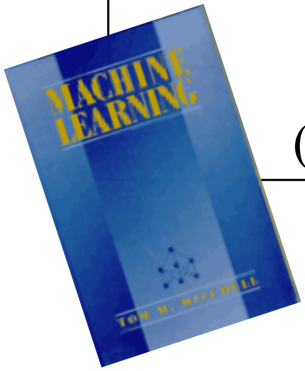
Measure Theory

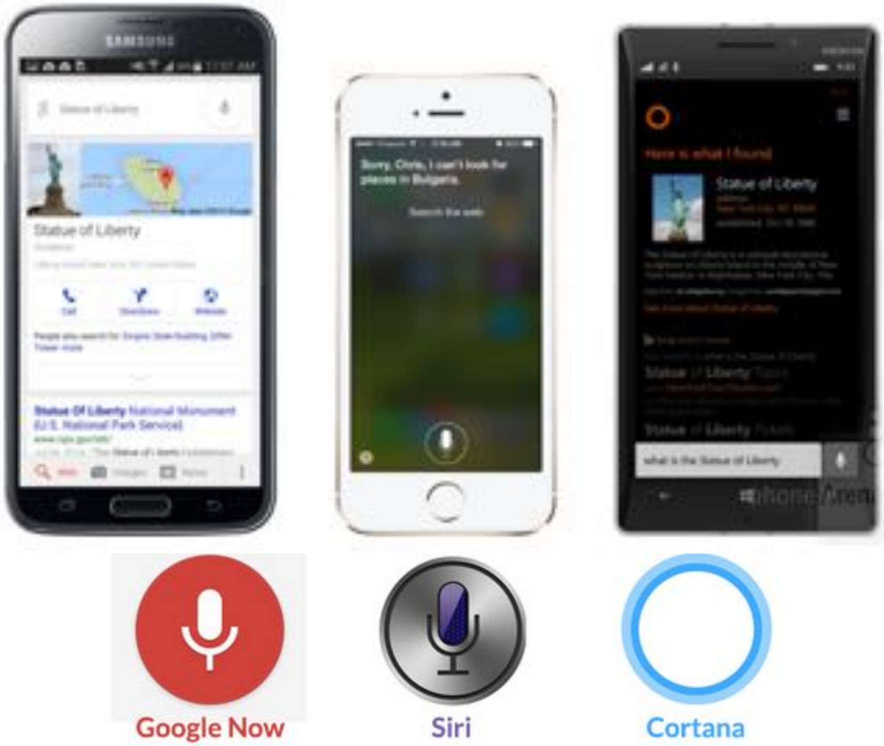
Linear Algebra

Speech Recognition

1. Learning to recognize spoken words

THEN	NOW
“…the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal…neural network methods…hidden Markov models…” (Mitchell, 1997)	

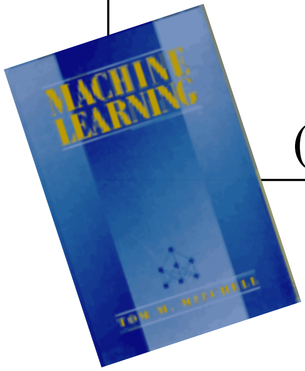




Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults

Robotics

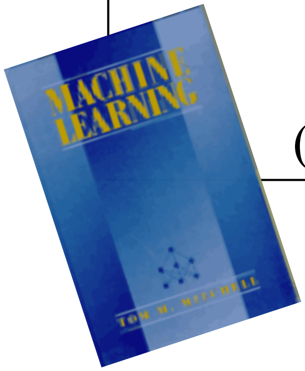
2. Learning to drive an autonomous vehicle

THEN	NOW
"…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…" (Mitchell, 1997)  	 waymo.com

Robotics

2. Learning to drive an autonomous vehicle

THEN	NOW
"…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…" (Mitchell, 1997)	 https://www.geek.com/wp-content/uploads/2016/03/uber.jpg



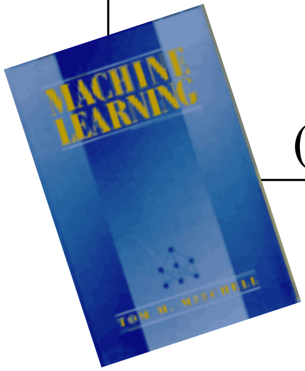
Games / Reasoning

3. Learning to beat the masters at board games

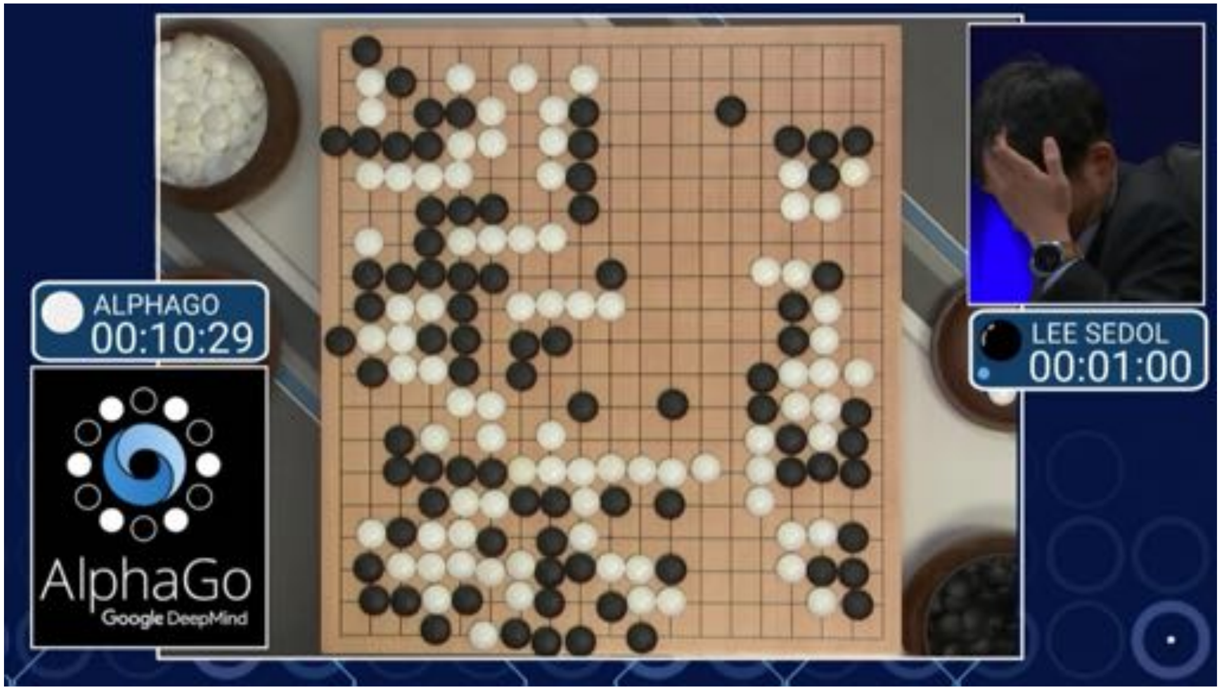
THEN

“…the world’s top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself…”

(Mitchell, 1997)



NOW

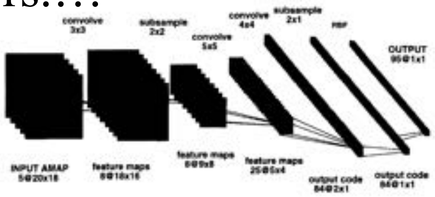


Computer Vision

4. Learning to recognize images

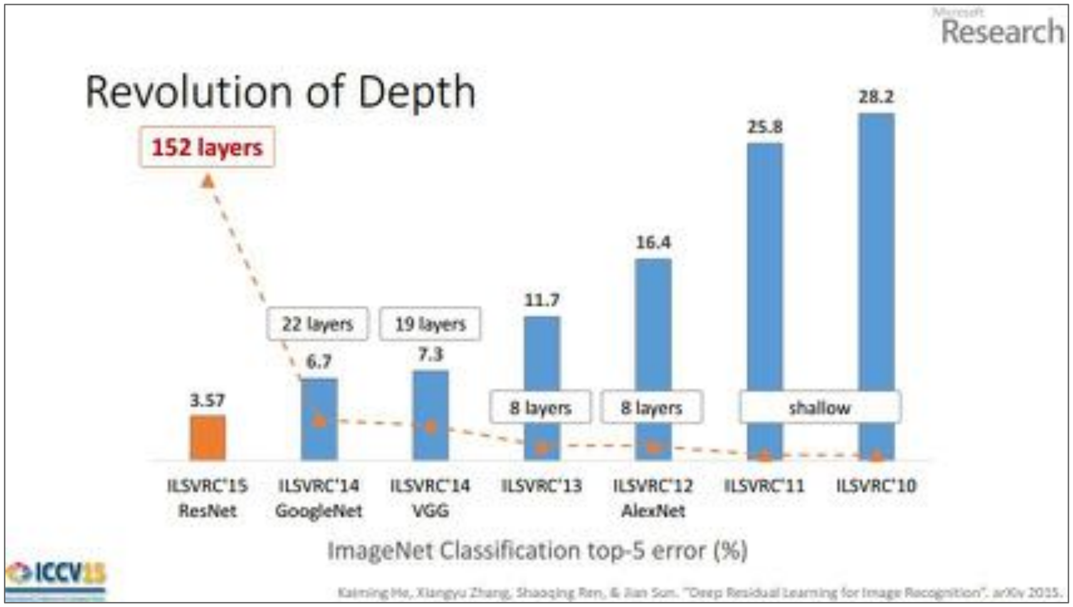
THEN

"…The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors.…"



(LeCun et al., 1995)

NOW

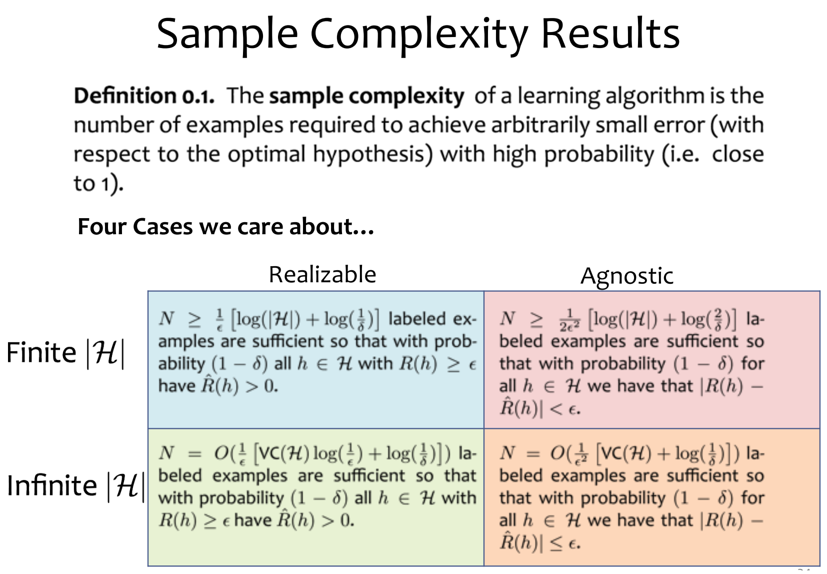


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Images from https://blog.openai.com/generative-models/

Learning Theory

- 5. In what cases and how well can we learn?

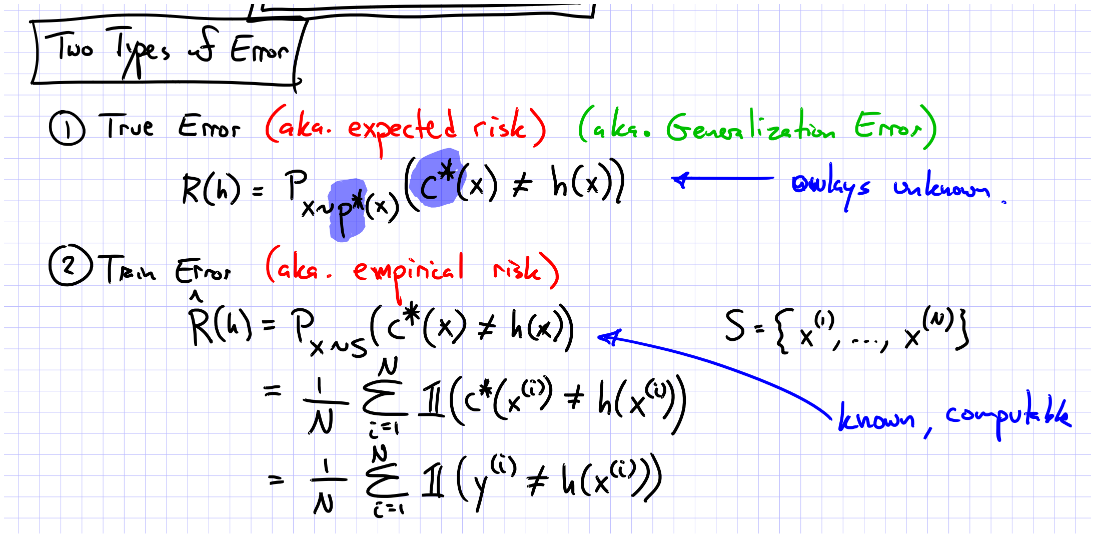


Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about…

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	$N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $R(h) \geq \epsilon$ have $\hat{R}(h) > 0$.	$N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert < \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$	$N = O(\frac{1}{\epsilon}\left[\text{VC}(\mathcal{H})\log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $R(h) \geq \epsilon$ have $\hat{R}(h) > 0$.	$N = O(\frac{1}{\epsilon^2}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.



Two Types of Error

① True Error (aka. expected risk) (aka. Generalization Error)

$R(h) = P_{x\sim p^*(x)}(c^*(x) \neq h(x))$ ← always unknown.

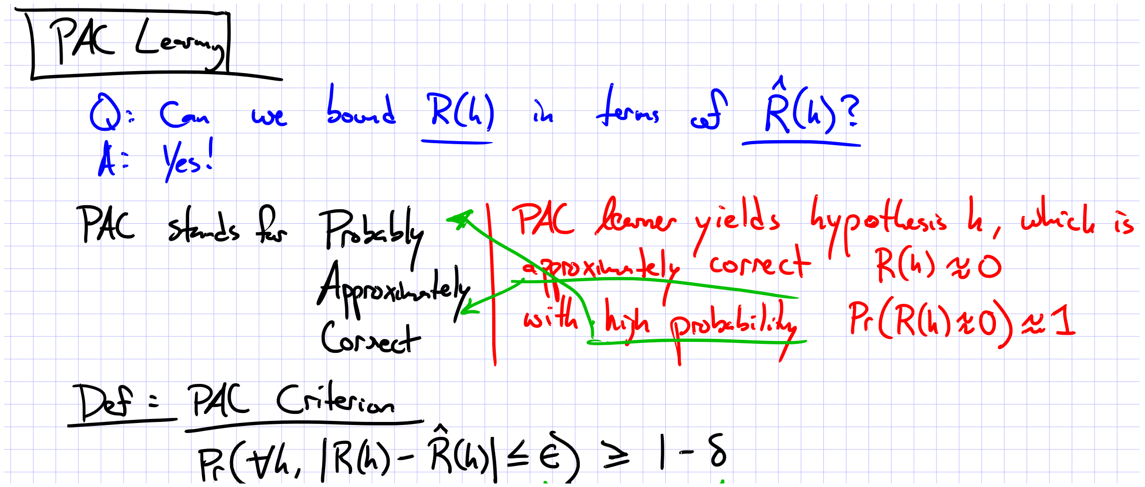
② Train Error (aka. empirical risk)

$\hat{R}(h) = P_{x\sim S}(c^*(x) \neq h(x))$ ← known, computable

$S = \{x^{(1)}, \ldots, x^{(N)}\}$

$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(c^*(x^{(i)}) \neq h(x^{(i)}))$

$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(y^{(i)} \neq h(x^{(i)}))$



PAC Learning

Q: Can we bound $\underline{R(h)}$ in terms of $\underline{\hat{R}(h)}$?

A: Yes!

PAC stands for Probably Approximately Correct

PAC learner yields hypothesis h, which is approximately correct $R(h) \approx 0$ with high probability $Pr(R(h) \approx 0) \approx 1$

Def: PAC Criterion

$Pr(\forall h, |R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$

1. How many examples do we need to learn?
2. How do we quantify our ability to generalize to unseen data?
3. Which algorithms are better suited to specific learning settings?

What is Machine Learning?

The goal of this course is to provide you with a toolbox:

Machine Learning

Statistics

Probability

Computer Science

Optimization

To solve all the problems above and more

Topics Covered in an ML Course

(e.g. 10-601)

- Foundations
  - Probability
  - MLE, MAP
  - Optimization
- Classifiers
  - KNN
  - Naïve Bayes
  - Logistic Regression
  - Perceptron
  - SVM
- Regression
  - Linear Regression
- Important Concepts
  - Kernels
  - Regularization and Overfitting
  - Experimental Design
- Unsupervised Learning
  - K-means / Lloyd's method
  - PCA
  - EM / GMMs
- Neural Networks
  - Feedforward Neural Nets
  - Basic architectures
  - Backpropagation
  - CNNs
- Graphical Models
  - Bayesian Networks
  - HMMs
  - Learning and Inference
- Learning Theory
  - Statistical Estimation (covered right before midterm)
  - PAC Learning
- Other Learning Paradigms
  - Matrix Factorization
  - Reinforcement Learning
  - Information Theory

ML Big Picture

Learning Paradigms:

What data is available and when? What form of prediction?

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

Theoretical Foundations:

What principles guide learning?

❑ probabilistic

❑ information theoretic

❑ evolutionary search

❑ ML as optimization

Problem Formulation:

What is the structure of our output prediction?

boolean	Binary Classification
categorical	Multiclass Classification
ordinal	Ordinal Classification
real	Regression
ordering	Ranking
multiple discrete	Structured Prediction
multiple continuous	(e.g. dynamical systems)
both discrete & cont.	(e.g. mixed graphical models)

Application Areas

Key challenges?

NLP, Speech, Computer Vision, Robotics, Medicine, Search

Facets of Building ML Systems:

How to build systems that are robust, efficient, adaptive, effective?

1. *Data prep*
2. *Model selection*
3. *Training (optimization / search)*
4. *Hyperparameter tuning on validation data*
5. *(Blind) Assessment on test data*

Big Ideas in ML:

Which are the ideas driving development of the field?

- *inductive bias*
- *generalization / overfitting*
- *bias-variance decomposition*
- *generative vs. discriminative*
- *deep nets, graphical models*
- *PAC learning*
- *distant rewards*

WHY DO WE NEED MATH / CS FOR MACHINE LEARNING?

10-606

Why Math for ML?

To best understand __A__ we need __B__

A	B

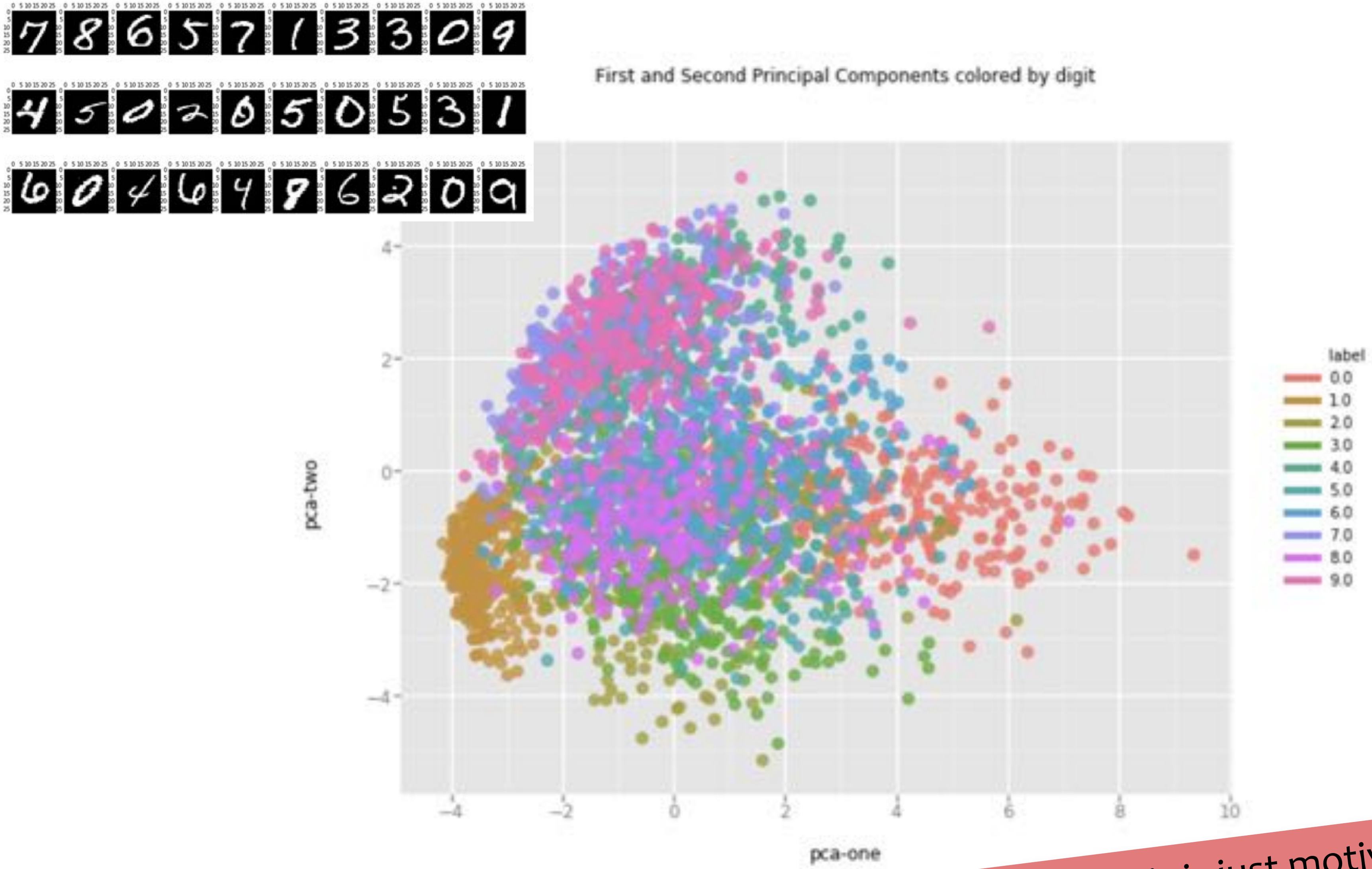
Why Math for ML?

10-606

To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition

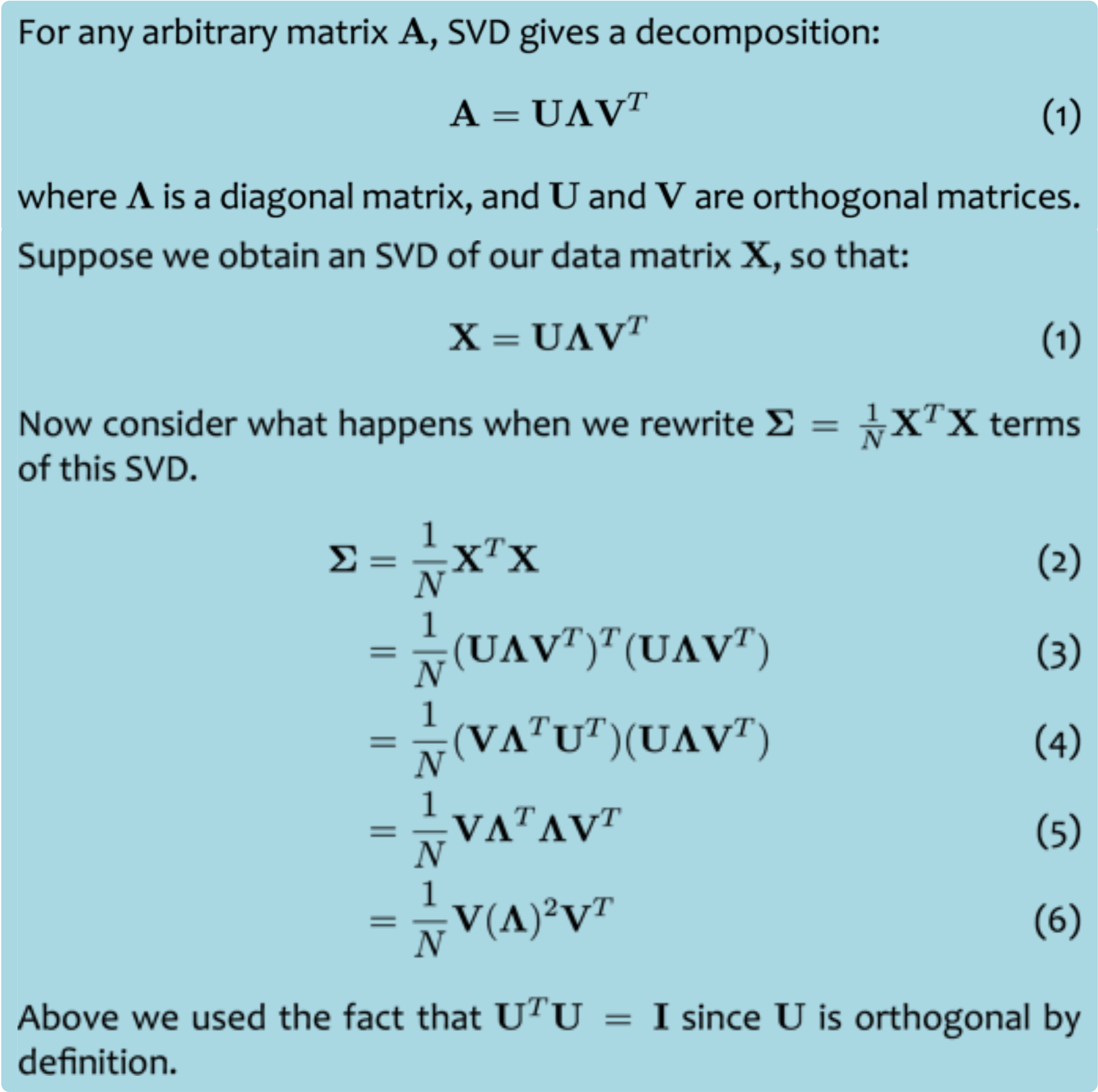
Principle Component Analysis (PCA)



Note: This is just motivation – we'll cover the math need to understand these topics later!

Figure from https://medium.com/@luckylwk/visualising-high-dimensional-datase
python-8ef87e7915b

SVD for PCA



For any arbitrary matrix $\mathbf{A}$, SVD gives a decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{1}$$

where $\mathbf{\Lambda}$ is a diagonal matrix, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

Suppose we obtain an SVD of our data matrix $\mathbf{X}$, so that:

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{1}$$

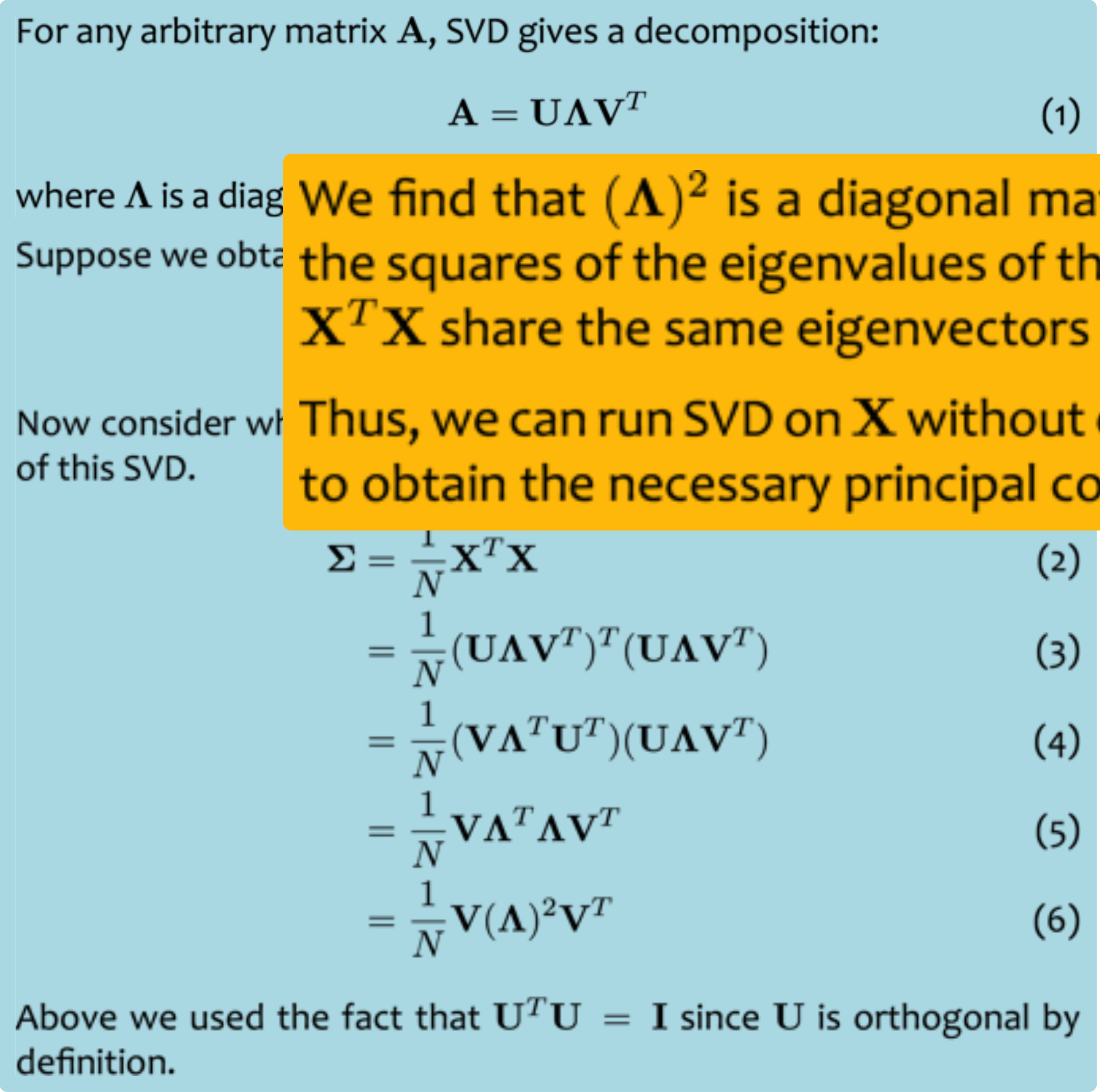
Now consider what happens when we rewrite $\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$ terms of this SVD.

$$\begin{aligned}
\mathbf{\Sigma} &= \frac{1}{N}\mathbf{X}^T\mathbf{X} && (2)\\
&= \frac{1}{N}(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T) && (3)\\
&= \frac{1}{N}(\mathbf{V}\mathbf{\Lambda}^T\mathbf{U}^T)(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T) && (4)\\
&= \frac{1}{N}\mathbf{V}\mathbf{\Lambda}^T\mathbf{\Lambda}\mathbf{V}^T && (5)\\
&= \frac{1}{N}\mathbf{V}(\mathbf{\Lambda})^2\mathbf{V}^T && (6)
\end{aligned}$$

Above we used the fact that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ since $\mathbf{U}$ is orthogonal by definition.

Note: This is just motivation – we'll cover the math need to understand these topics later!

SVD for PCA



For any arbitrary matrix $\mathbf{A}$, SVD gives a decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{1}$$

where $\mathbf{\Lambda}$ is a diag

Suppose we obta

Now consider wh
of this SVD.

$$\begin{aligned}
\mathbf{\Sigma} &= \frac{1}{N}\mathbf{X}^T\mathbf{X} && (2)\\
&= \frac{1}{N}(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T) && (3)\\
&= \frac{1}{N}(\mathbf{V}\mathbf{\Lambda}^T\mathbf{U}^T)(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T) && (4)\\
&= \frac{1}{N}\mathbf{V}\mathbf{\Lambda}^T\mathbf{\Lambda}\mathbf{V}^T && (5)\\
&= \frac{1}{N}\mathbf{V}(\mathbf{\Lambda})^2\mathbf{V}^T && (6)
\end{aligned}$$

Above we used the fact that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ since $\mathbf{U}$ is orthogonal by definition.

We find that $(\mathbf{\Lambda})^2$ is a diagonal matrix whose entries are $\Lambda_{ii} = \lambda_i^2$ the squares of the eigenvalues of the SVD of $\mathbf{X}$. Further, both $\mathbf{X}$ and $\mathbf{X}^T\mathbf{X}$ share the same eigenvectors in their SVD.

Thus, we can run SVD on $\mathbf{X}$ without ever instantiating the large $\mathbf{X}^T\mathbf{X}$ to obtain the necessary principal components more efficiently.

Note: This is just motivation – we'll cover the math need to understand these topics later!

Why Math for ML?

To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition

10-606

Why Math for ML?

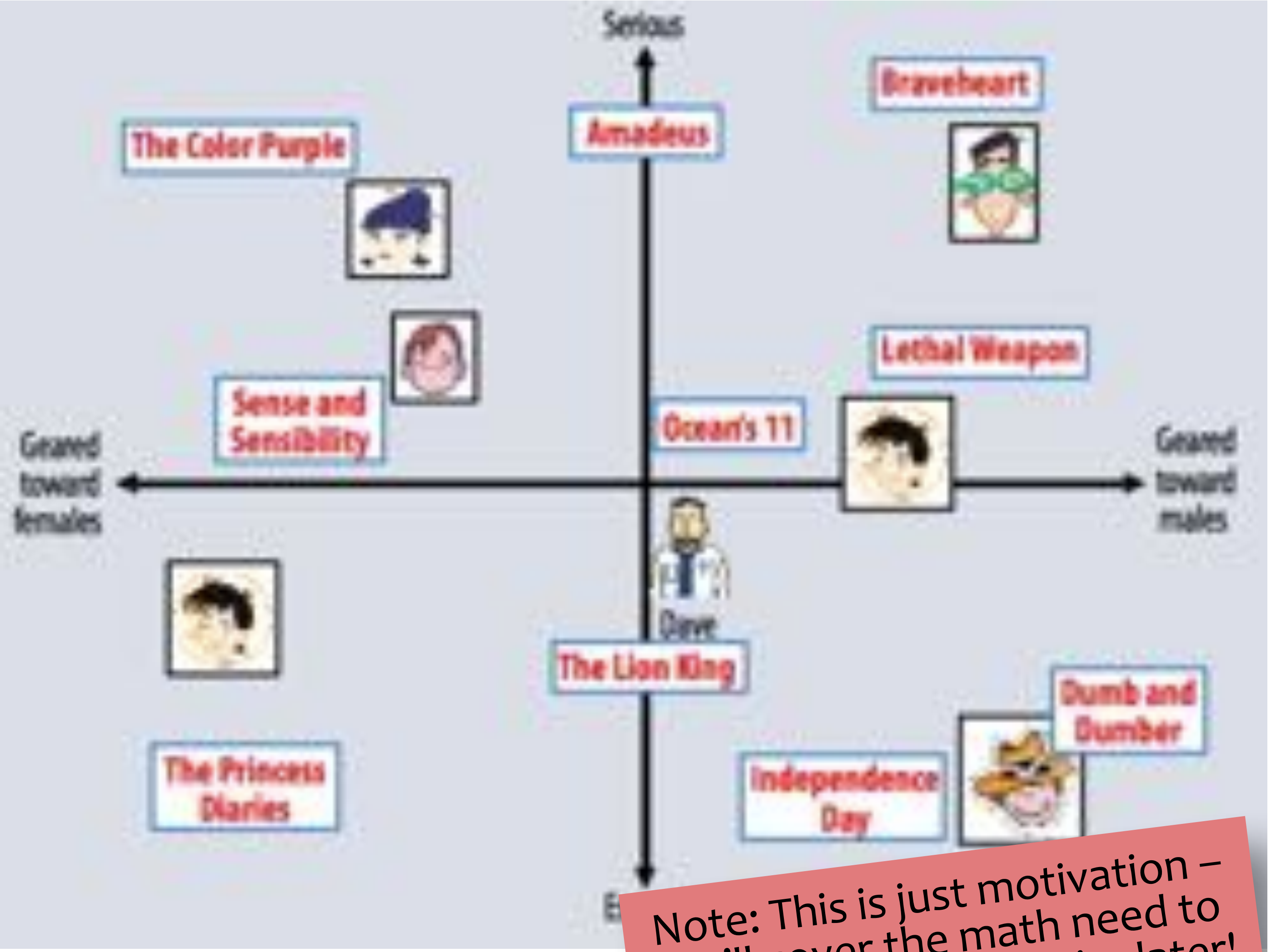
To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition
Gradient-based Matrix Factorization and Collaborative Filtering	Calculus • Chain-rule; Partial Derivatives; • Matrix Differentials; Second and Higher Differentials

Collaborative Filtering

Latent Factor Methods

- Assume that both movies and users live in some **low-dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space

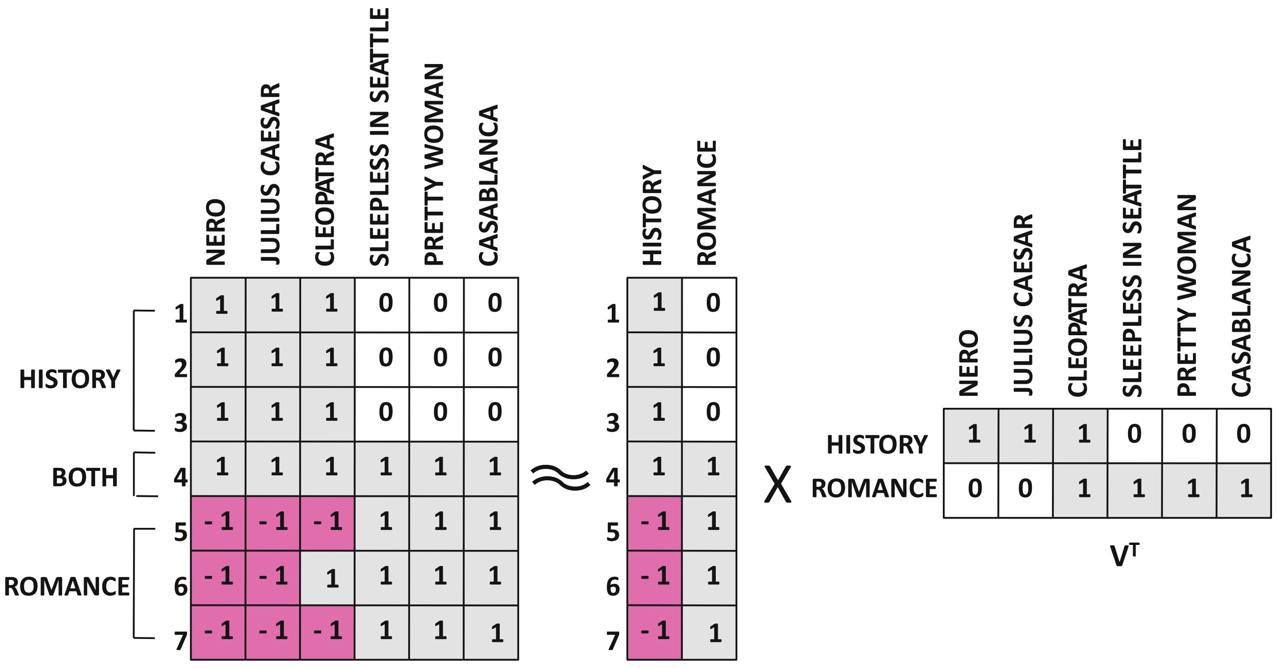


Note: This is just motivation – we'll cover the math need to understand these topics later!

Figures from Koren et al. (2009)

Example: Matrix Factorization

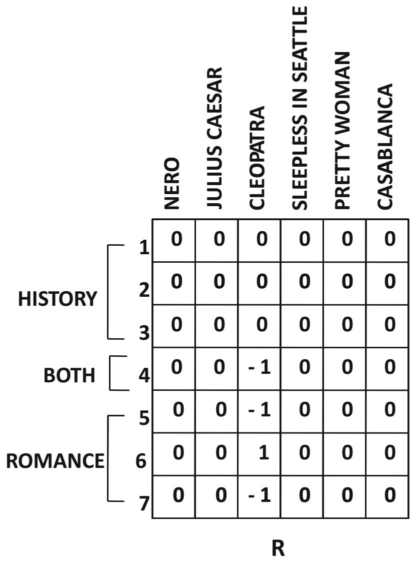
for the Netflix Problem



$$
\begin{array}{ll|cccccc}
 & & \text{NERO} & \text{JULIUS CAESAR} & \text{CLEOPATRA} & \text{SLEEPLESS IN SEATTLE} & \text{PRETTY WOMAN} & \text{CASABLANCA} \\
\hline
\text{HISTORY} & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\
 & 3 & 1 & 1 & 1 & 0 & 0 & 0 \\
\text{BOTH} & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\
\text{ROMANCE} & 5 & -1 & -1 & -1 & 1 & 1 & 1 \\
 & 6 & -1 & -1 & 1 & 1 & 1 & 1 \\
 & 7 & -1 & -1 & -1 & 1 & 1 & 1
\end{array}
\approx
\begin{array}{l|cc}
 & \text{HISTORY} & \text{ROMANCE} \\
\hline
1 & 1 & 0 \\
2 & 1 & 0 \\
3 & 1 & 0 \\
4 & 1 & 1 \\
5 & -1 & 1 \\
6 & -1 & 1 \\
7 & -1 & 1
\end{array}
\times
\begin{array}{l|cccccc}
 & \text{NERO} & \text{JULIUS CAESAR} & \text{CLEOPATRA} & \text{SLEEPLESS IN SEATTLE} & \text{PRETTY WOMAN} & \text{CASABLANCA} \\
\hline
\text{HISTORY} & 1 & 1 & 1 & 0 & 0 & 0 \\
\text{ROMANCE} & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

$$\text{R} \qquad \text{U} \qquad V^T$$

(a) Example of rank-2 matrix factorization



$$
\begin{array}{ll|cccccc}
 & & \text{NERO} & \text{JULIUS CAESAR} & \text{CLEOPATRA} & \text{SLEEPLESS IN SEATTLE} & \text{PRETTY WOMAN} & \text{CASABLANCA} \\
\hline
\text{HISTORY} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{BOTH} & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
\text{ROMANCE} & 5 & 0 & 0 & -1 & 0 & 0 & 0 \\
 & 6 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & 7 & 0 & 0 & -1 & 0 & 0 & 0
\end{array}
$$

$$\text{R}$$

(b) Residual matrix

Note: This is just motivation – we'll cover the math need to understand these topics later!

Figures from Aggarwal (2016)

Matrix Factorization
(with matrices)

- User vectors:

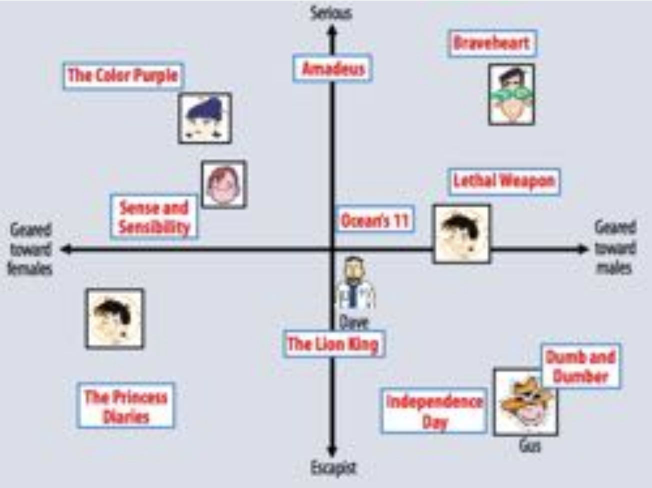
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

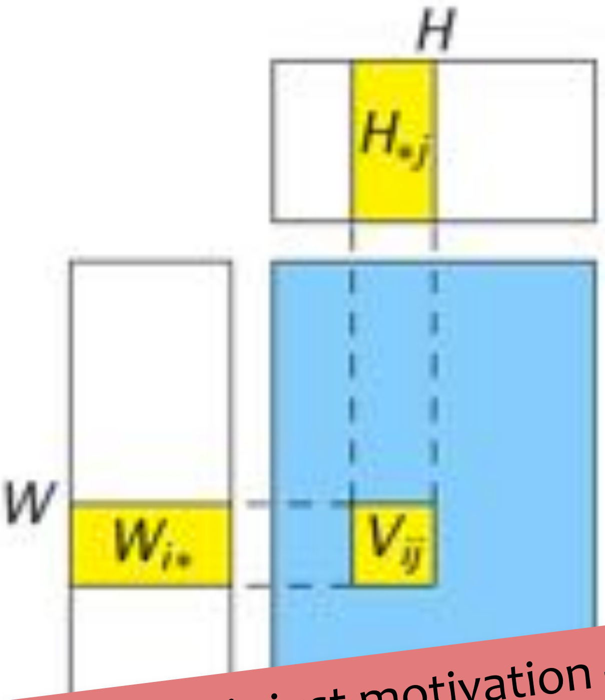
$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$\begin{aligned} V_{ui} &= W_{u*}H_{*i} \\ &= [WH]_{ui} \end{aligned}$$



Figures from Koren et al. (2009)



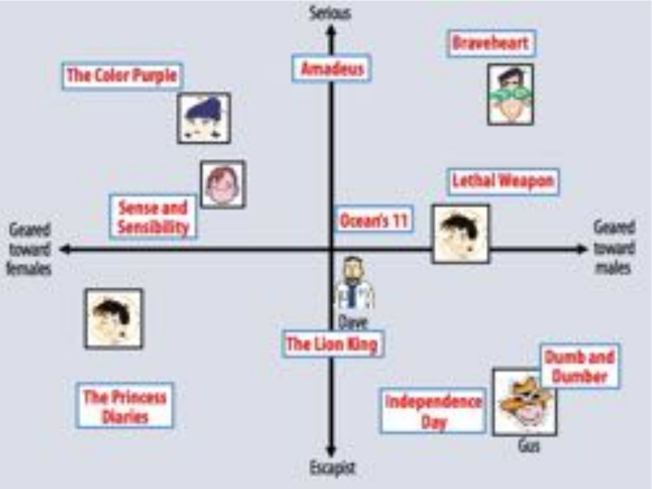
et al. (2011)

Note: This is just motivation – we'll cover the math need to understand these topics later!

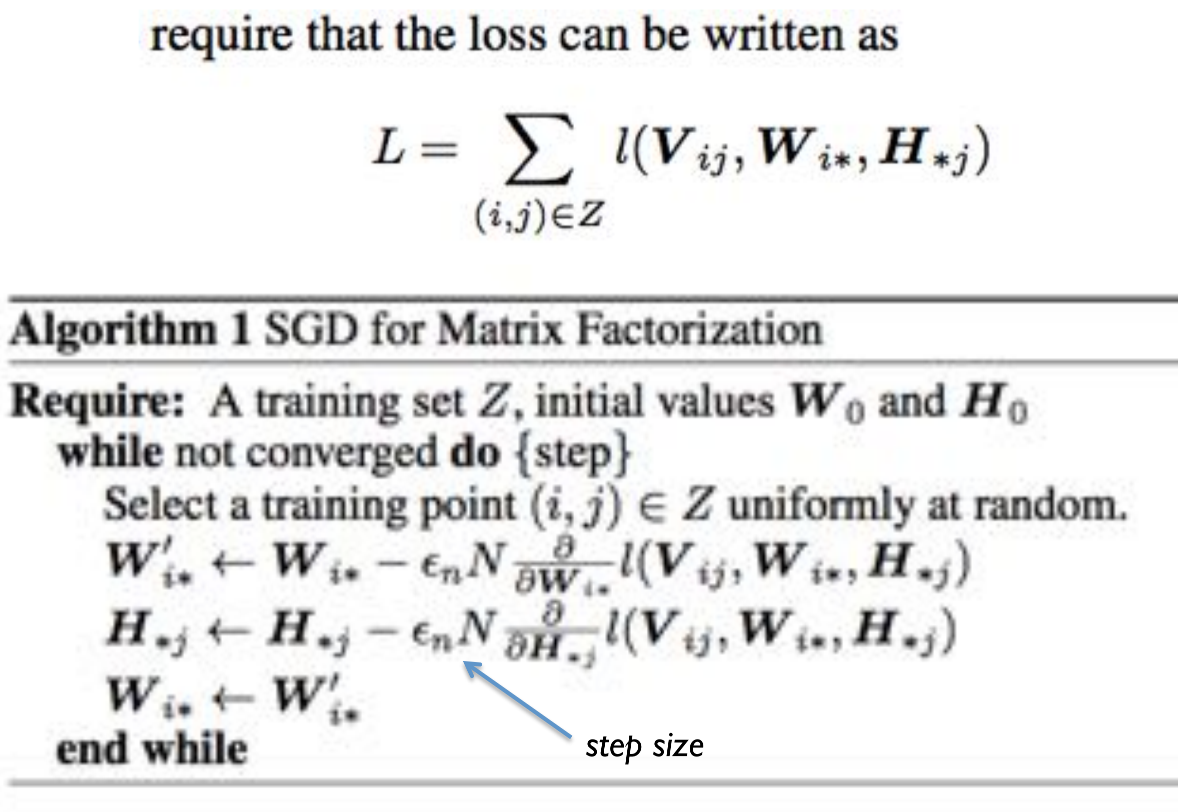
Matrix Factorization

(with matrices)

- Stochastic Gradient Descent



Figures from Koren et al. (2009)



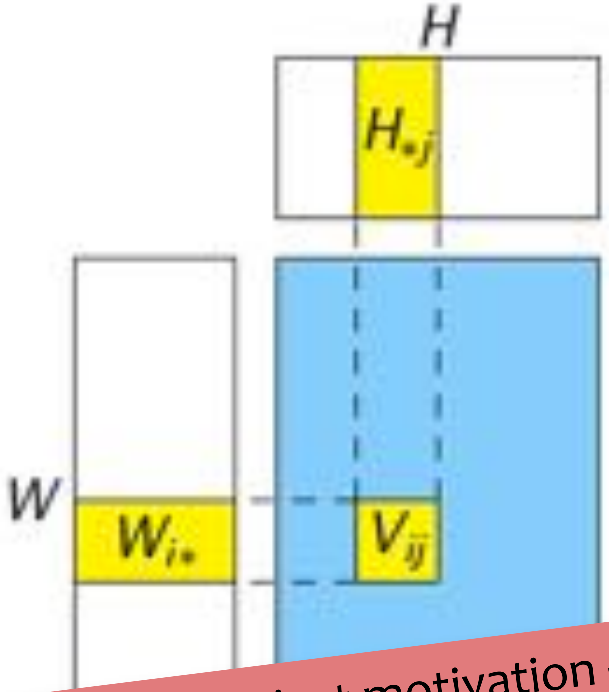
require that the loss can be written as

$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

```
Algorithm 1 SGD for Matrix Factorization
Require: A training set $Z$, initial values $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$
  while not converged do {step}
    Select a training point $(i, j) \in Z$ uniformly at random.
    $\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$
  end while
```

step size

Figure from Gemulla et al. (2011)



Note: This is just motivation – we'll cover the math need to understand these topics later!

et al. (2011)

10-606

Why Math for ML?

To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition
Gradient-based Matrix Factorization and Collaborative Filtering	Calculus • Chain-rule; Partial Derivatives; • Matrix Differentials; Second and Higher Differentials

Why Math for ML?

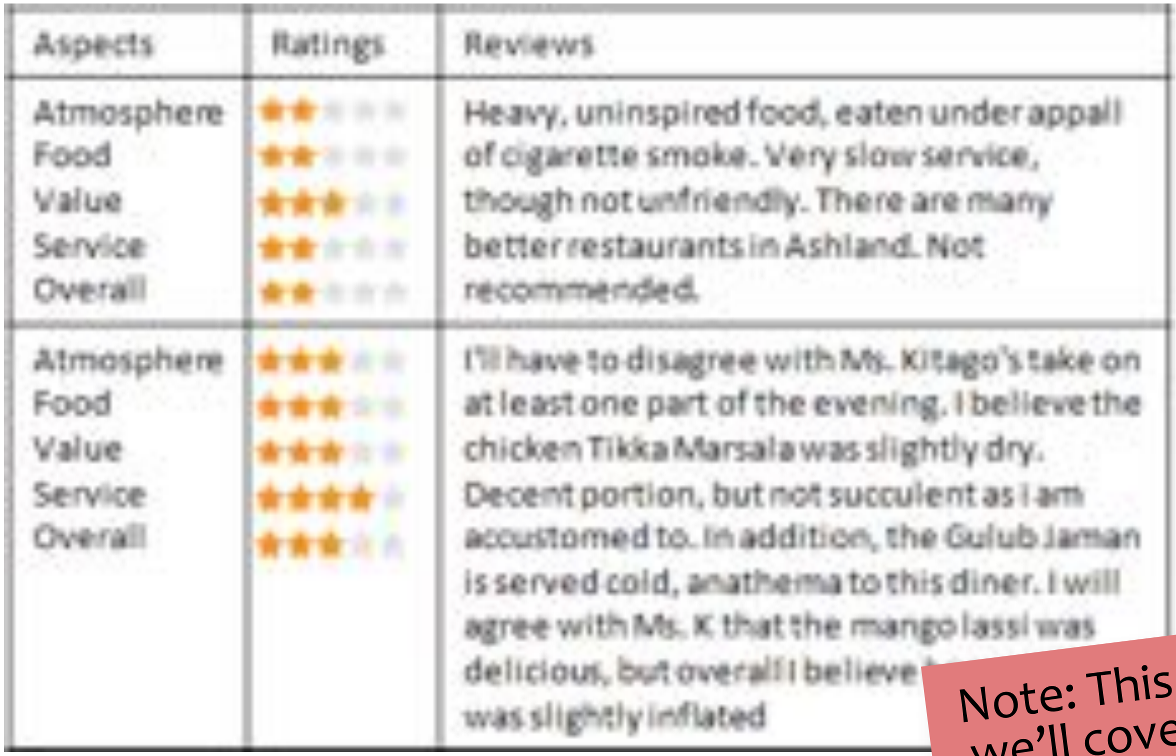
10-606

To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition
Gradient-based Matrix Factorization and Collaborative Filtering	Calculus • Chain-rule; Partial Derivatives; • Matrix Differentials; Second and Higher Differentials
Probabilistic Study of Ordinal Regression	Probability • Events • Discrete/Continuous Random Variables • Mean & Variance; Factorization • Multivariate Distributions

Sentiment Analysis

- *Task*: **Given** a restaurant description, **predict** how many stars the author would give for the "Overall" rating

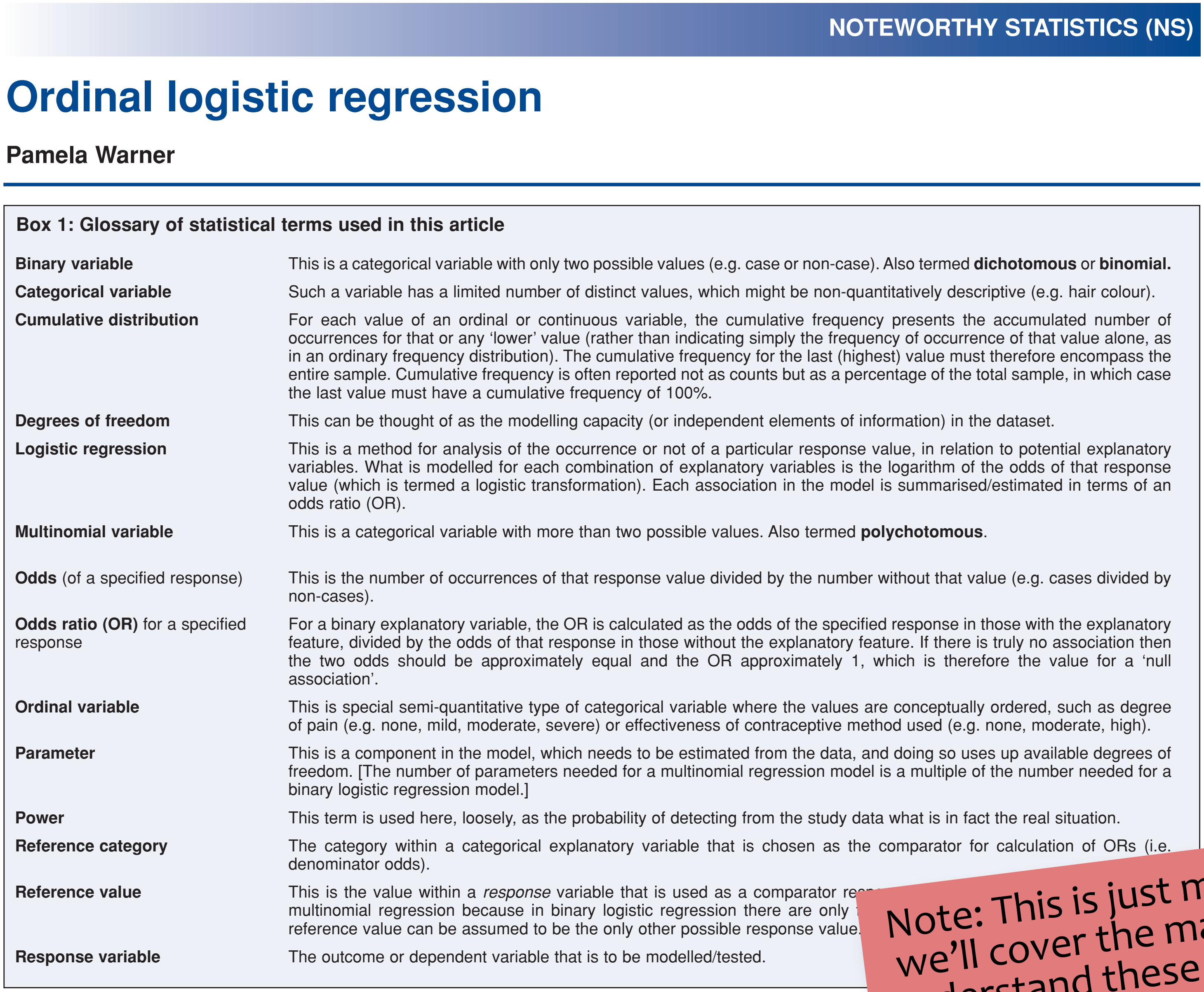


Aspects	Ratings	Reviews
Atmosphere Food Value Service Overall	★★☆☆☆ ★★☆☆☆ ★★★☆☆ ★★☆☆☆ ★★☆☆☆	Heavy, uninspired food, eaten under appall of cigarette smoke. Very slow service, though not unfriendly. There are many better restaurants in Ashland. Not recommended.
Atmosphere Food Value Service Overall	★★★☆☆ ★★★☆☆ ★★★☆☆ ★★★★☆ ★★★☆☆	I'll have to disagree with Ms. Kitago's take on at least one part of the evening. I believe the chicken Tikka Marsala was slightly dry. Decent portion, but not succulent as I am accustomed to. In addition, the Gulub Jaman is served cold, anathema to this diner. I will agree with Ms. K that the mango lassi was delicious, but overall I believe was slightly inflated

Note: This is just motivation – we'll cover the math need to understand these topics later!

Figures from Gupta et al. (2010)

Ordinal Regression



NOTEWORTHY STATISTICS (NS)

Ordinal logistic regression

Pamela Warner

Box 1: Glossary of statistical terms used in this article

Binary variable This is a categorical variable with only two possible values (e.g. case or non-case). Also termed **dichotomous** or **binomial.**

Categorical variable Such a variable has a limited number of distinct values, which might be non-quantitatively descriptive (e.g. hair colour).

Cumulative distribution For each value of an ordinal or continuous variable, the cumulative frequency presents the accumulated number of occurrences for that or any 'lower' value (rather than indicating simply the frequency of occurrence of that value alone, as in an ordinary frequency distribution). The cumulative frequency for the last (highest) value must therefore encompass the entire sample. Cumulative frequency is often reported not as counts but as a percentage of the total sample, in which case the last value must have a cumulative frequency of 100%.

Degrees of freedom This can be thought of as the modelling capacity (or independent elements of information) in the dataset.

Logistic regression This is a method for analysis of the occurrence or not of a particular response value, in relation to potential explanatory variables. What is modelled for each combination of explanatory variables is the logarithm of the odds of that response value (which is termed a logistic transformation). Each association in the model is summarised/estimated in terms of an odds ratio (OR).

Multinomial variable This is a categorical variable with more than two possible values. Also termed **polychotomous**.

Odds (of a specified response) This is the number of occurrences of that response value divided by the number without that value (e.g. cases divided by non-cases).

Odds ratio (OR) for a specified response For a binary explanatory variable, the OR is calculated as the odds of the specified response in those with the explanatory feature, divided by the odds of that response in those without the explanatory feature. If there is truly no association then the two odds should be approximately equal and the OR approximately 1, which is therefore the value for a 'null association'.

Ordinal variable This is special semi-quantitative type of categorical variable where the values are conceptually ordered, such as degree of pain (e.g. none, mild, moderate, severe) or effectiveness of contraceptive method used (e.g. none, moderate, high).

Parameter This is a component in the model, which needs to be estimated from the data, and doing so uses up available degrees of freedom. [The number of parameters needed for a multinomial regression model is a multiple of the number needed for a binary logistic regression model.]

Power This term is used here, loosely, as the probability of detecting from the study data what is in fact the real situation.

Reference category The category within a categorical explanatory variable that is chosen as the comparator for calculation of ORs (i.e. denominator odds).

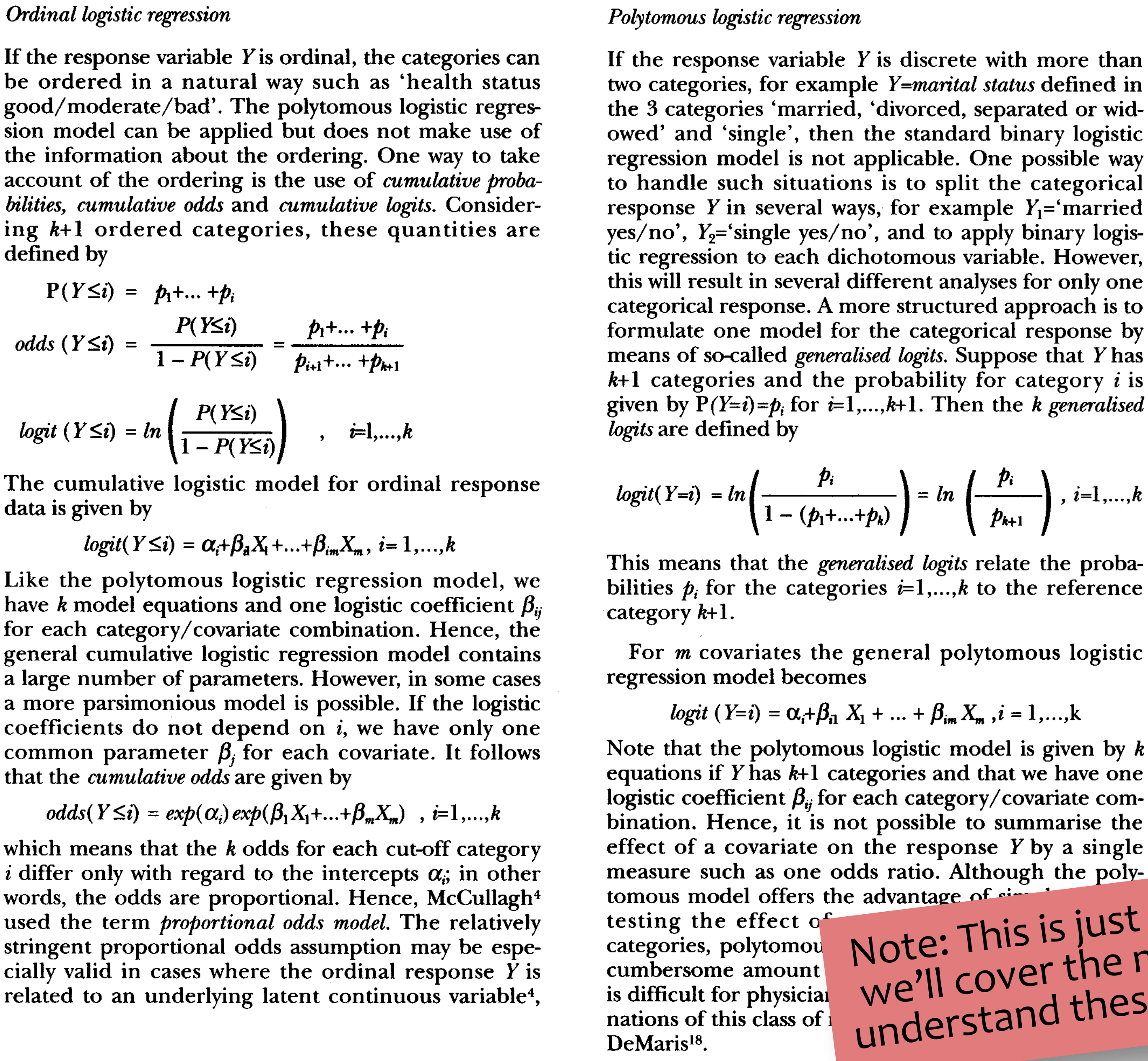
Reference value This is the value within a *response* variable that is used as a comparator res multinomial regression because in binary logistic regression there are only reference value can be assumed to be the only other possible response value.

Response variable The outcome or dependent variable that is to be modelled/tested.

Note: This is just motivation – we'll cover the math need to understand these topics later!

Figures from Warner (2008)

Ordinal Regression



Ordinal logistic regression

If the response variable Y is ordinal, the categories can be ordered in a natural way such as 'health status good/moderate/bad'. The polytomous logistic regression model can be applied but does not make use of the information about the ordering. One way to take account of the ordering is the use of *cumulative probabilities, cumulative odds* and *cumulative logits*. Considering k+1 ordered categories, these quantities are defined by

$$\mathrm{P}(Y \leq i) = p_1 + \ldots + p_i$$

$$odds\ (Y \leq i) = \frac{P(Y \leq i)}{1 - P(Y \leq i)} = \frac{p_1 + \ldots + p_i}{p_{i+1} + \ldots + p_{k+1}}$$

$$logit\ (Y \leq i) = ln\left(\frac{P(Y \leq i)}{1 - P(Y \leq i)}\right) \quad , \quad i{=}1,\ldots,k$$

The cumulative logistic model for ordinal response data is given by

$$logit(Y \leq i) = \alpha_i + \beta_{i1} X_1 + \ldots + \beta_{im} X_m,\ i = 1,\ldots,k$$

Like the polytomous logistic regression model, we have k model equations and one logistic coefficient β_{ij} for each category/covariate combination. Hence, the general cumulative logistic regression model contains a large number of parameters. However, in some cases a more parsimonious model is possible. If the logistic coefficients do not depend on i, we have only one common parameter β_j for each covariate. It follows that the *cumulative odds* are given by

$$odds(Y \leq i) = exp(\alpha_i)\, exp(\beta_1 X_1 + \ldots + \beta_m X_m) \quad , i{=}1,\ldots,k$$

which means that the k odds for each cut-off category i differ only with regard to the intercepts α_i; in other words, the odds are proportional. Hence, McCullagh[4] used the term *proportional odds model*. The relatively stringent proportional odds assumption may be especially valid in cases where the ordinal response Y is related to an underlying latent continuous variable[4],

Polytomous logistic regression

If the response variable Y is discrete with more than two categories, for example Y=*marital status* defined in the 3 categories 'married, 'divorced, separated or widowed' and 'single', then the standard binary logistic regression model is not applicable. One possible way to handle such situations is to split the categorical response Y in several ways, for example Y_1='married yes/no', Y_2='single yes/no', and to apply binary logistic regression to each dichotomous variable. However, this will result in several different analyses for only one categorical response. A more structured approach is to formulate one model for the categorical response by means of so-called *generalised logits*. Suppose that Y has k+1 categories and the probability for category i is given by $\mathrm{P}(Y{=}i){=}p_i$ for i=1,…,k+1. Then the k *generalised logits* are defined by

$$logit(Y{=}i) = ln\left(\frac{p_i}{1 - (p_1 + \ldots + p_k)}\right) = ln\left(\frac{p_i}{p_{k+1}}\right),\ i{=}1,\ldots,k$$

This means that the *generalised logits* relate the probabilities p_i for the categories i=1,…,k to the reference category k+1.

For m covariates the general polytomous logistic regression model becomes

$$logit\ (Y{=}i) = \alpha_i + \beta_{i1} X_1 + \ldots + \beta_{im} X_m,\ i = 1,\ldots,\mathrm{k}$$

Note that the polytomous logistic model is given by k equations if Y has k+1 categories and that we have one logistic coefficient β_{ij} for each category/covariate combination. Hence, it is not possible to summarise the effect of a covariate on the response Y by a single measure such as one odds ratio. Although the polytomous model offers the advantage of
testing the effect of
categories, polytomou
cumbersome amount
is difficult for physicia
nations of this class of
DeMaris[18].

Note: This is just motivation – we'll cover the math need to understand these topics later!

Figures from Bender & Grouven (1997)

Why Math for ML?

10-606

To best understand __A__ we need __B__

A	B
Derivation of Principal Component Analysis (PCA)	Linear Algebra • Vector spaces, Functions and Function Spaces • Matrices and linear operators • Matrix decomposition
Gradient-based Matrix Factorization and Collaborative Filtering	Calculus • Chain-rule; Partial Derivatives; • Matrix Differentials; Second and Higher Differentials
Probabilistic Study of Ordinal Regression	Probability • Events • Discrete/Continuous Random Variables • Mean & Variance; Factorization • Multivariate Distributions

The core content for this course is the **mathematics** (Column B), but you will apply what you learn to **real problems in machine learning** (Column A)

Why Computer Science for ML?

10-607

To best understand __A__ we need __B__

A	B

Why Computer Science for ML?

10-607

To best understand __A__ we need __B__

A	B
Analysis of Exact Inference in Graphical Models	Computation • Computational Complexity • Recursion; Dynamic Programming • Data Structures for ML Algorithms

Factor Graph Notation

- Variables:

$$\mathcal{X} = \{X_1, \ldots, X_i, \ldots, X_n\}$$

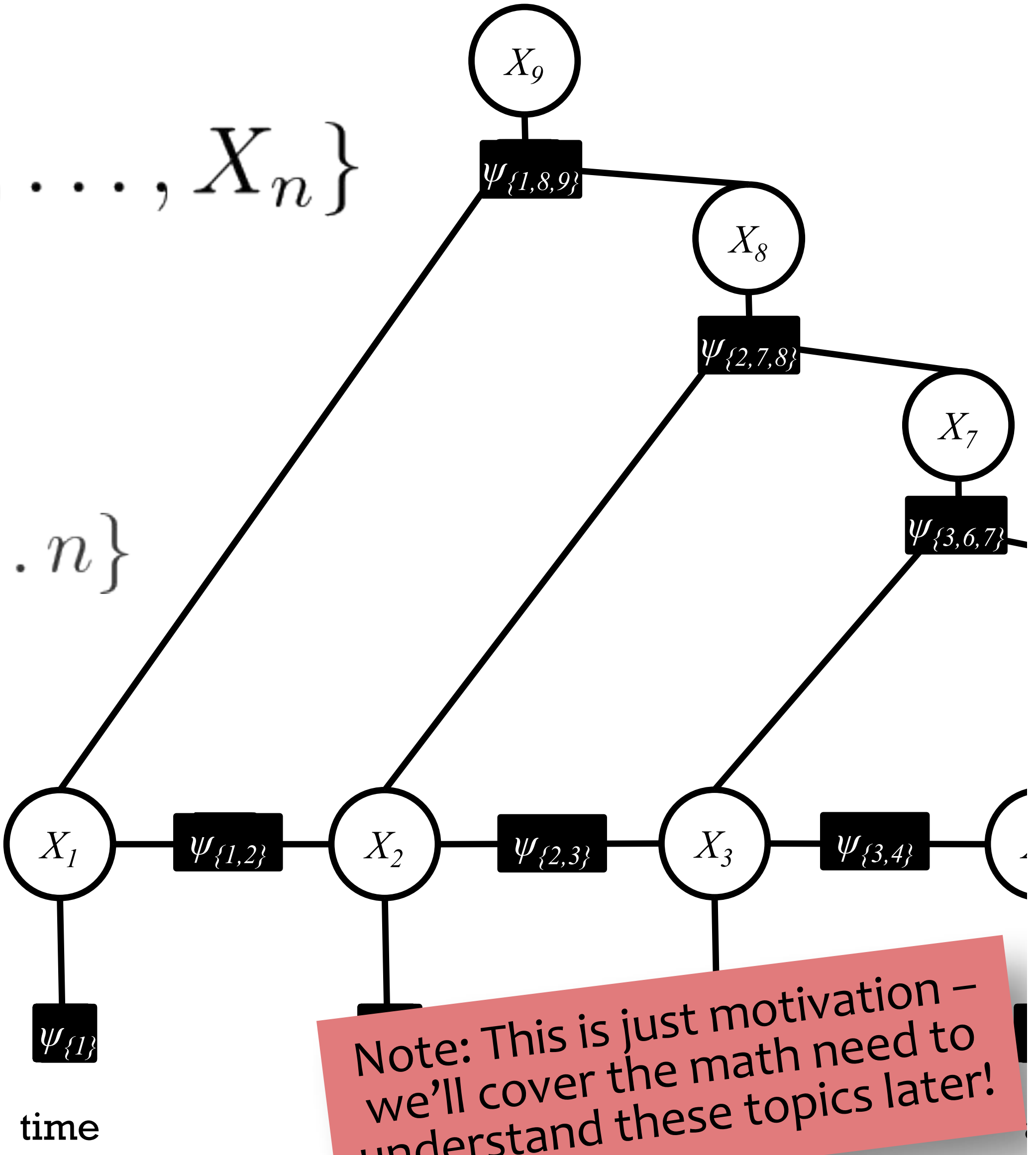
- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

where $\alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\}$

Joint Distribution

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_\alpha \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$

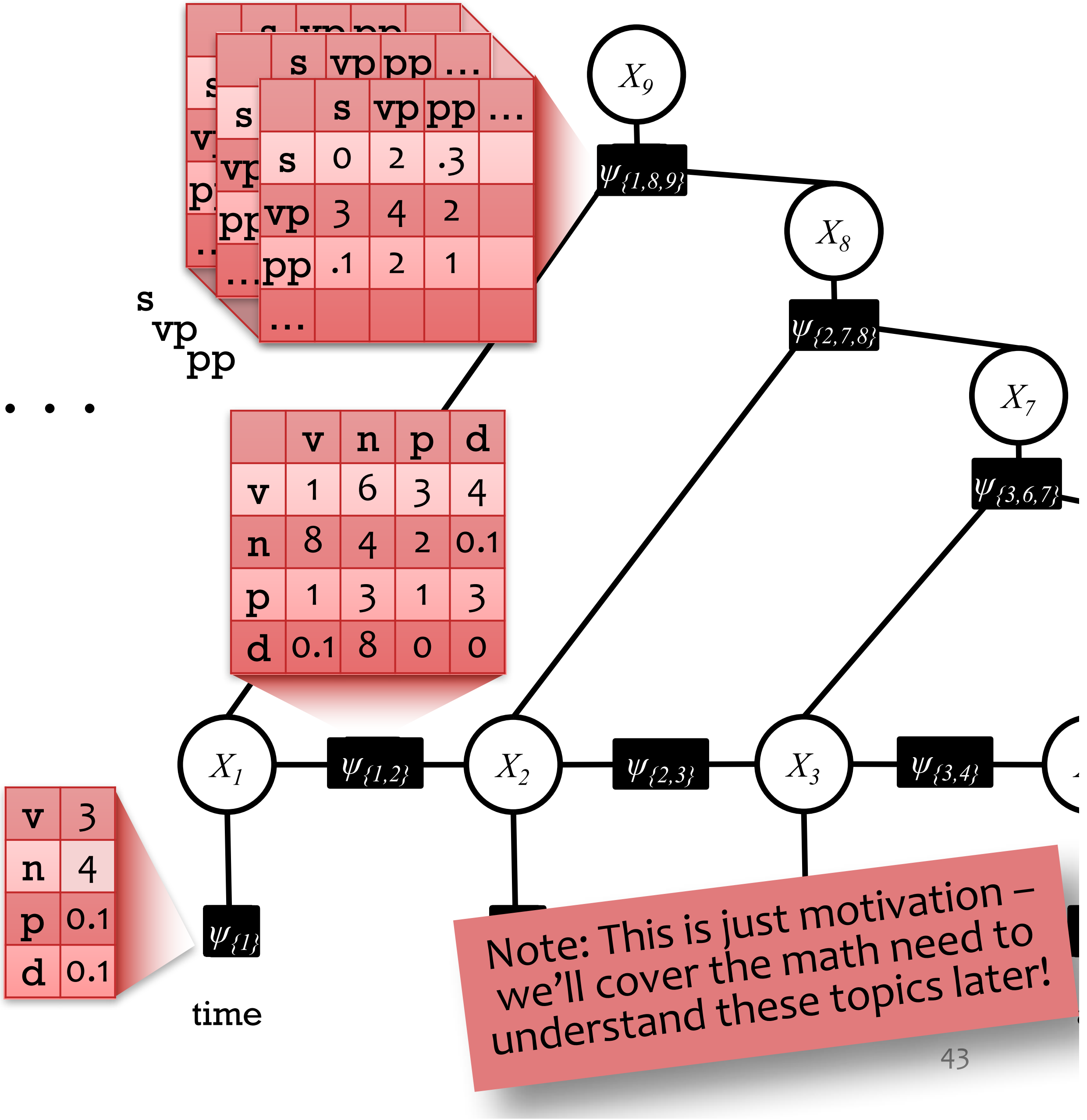


Note: This is just motivation – we'll cover the math need to understand these topics later!

Factors are Tensors

- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$



	s	vp	pp	…
s	0	2	.3	
vp	3	4	2	
pp	.1	2	1	
…				

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

v	3
n	4
p	0.1
d	0.1

Note: This is just motivation – we'll cover the math need to understand these topics later!

Inference

Given a factor graph, two common tasks …

- Compute the most likely joint assignment,
  $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}}\, p(\boldsymbol{X}=\boldsymbol{x})$
- ☆ Compute the marginal distribution of variable X_i:
  $p(X_i=x_i)$ for each value x_i

Both consider *all* joint assignments.

Both are NP-Hard in general.

So, we turn to **approximations**.

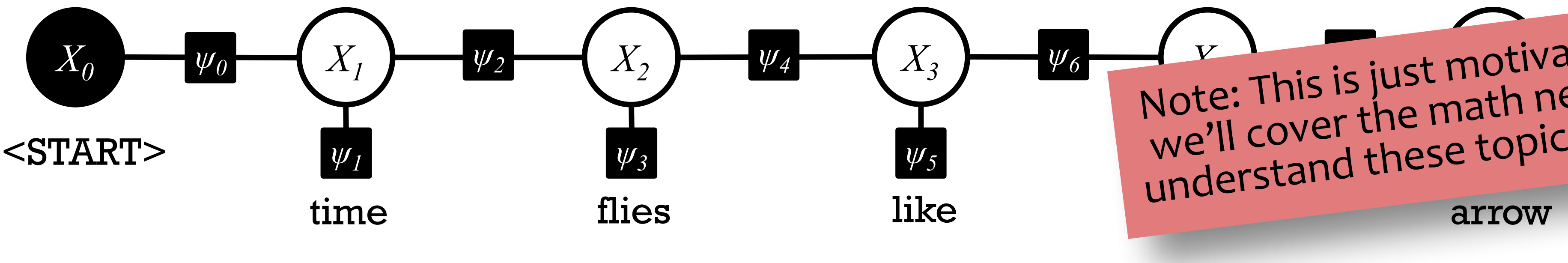
$p(X_i=x_i)$ = sum of $p(\boldsymbol{X}=\boldsymbol{x})$ over joint

Note: This is just motivation – we'll cover the math need to understand these topics later!

Marginals by Sampling on Factor Graph

Suppose we took many samples from the distribution over taggings: $p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$

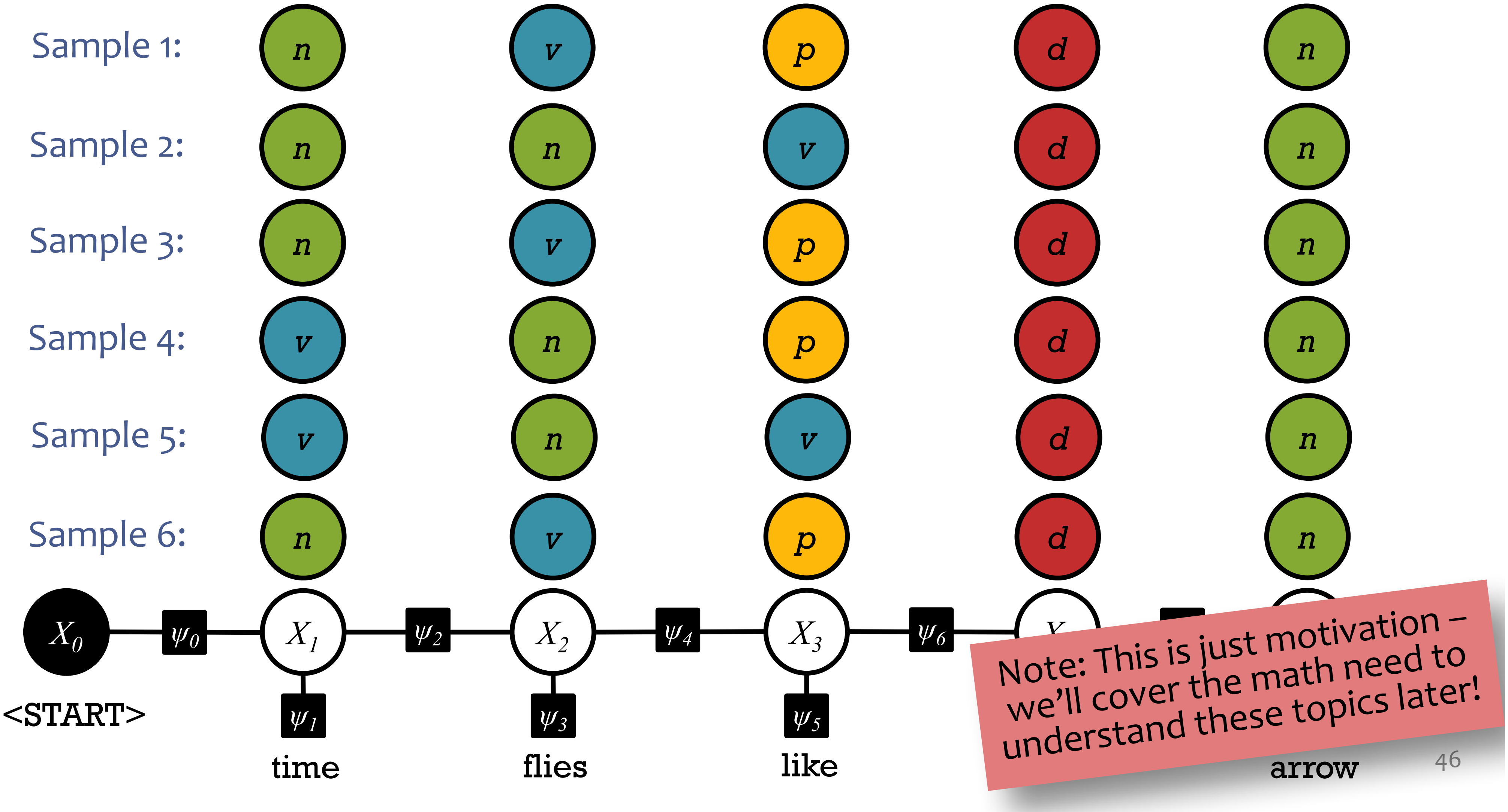
Sample 1:	*n*	*v*	*p*	*d*	*n*
Sample 2:	*n*	*n*	*v*	*d*	*n*
Sample 3:	*n*	*v*	*p*	*d*	*n*
Sample 4:	*v*	*n*	*p*	*d*	*n*
Sample 5:	*v*	*n*	*v*	*d*	*n*
Sample 6:	*n*	*v*	*p*	*d*	*n*



Note: This is just motivation – we'll cover the math need to understand these topics later!

Marginals by Sampling on Factor Graph

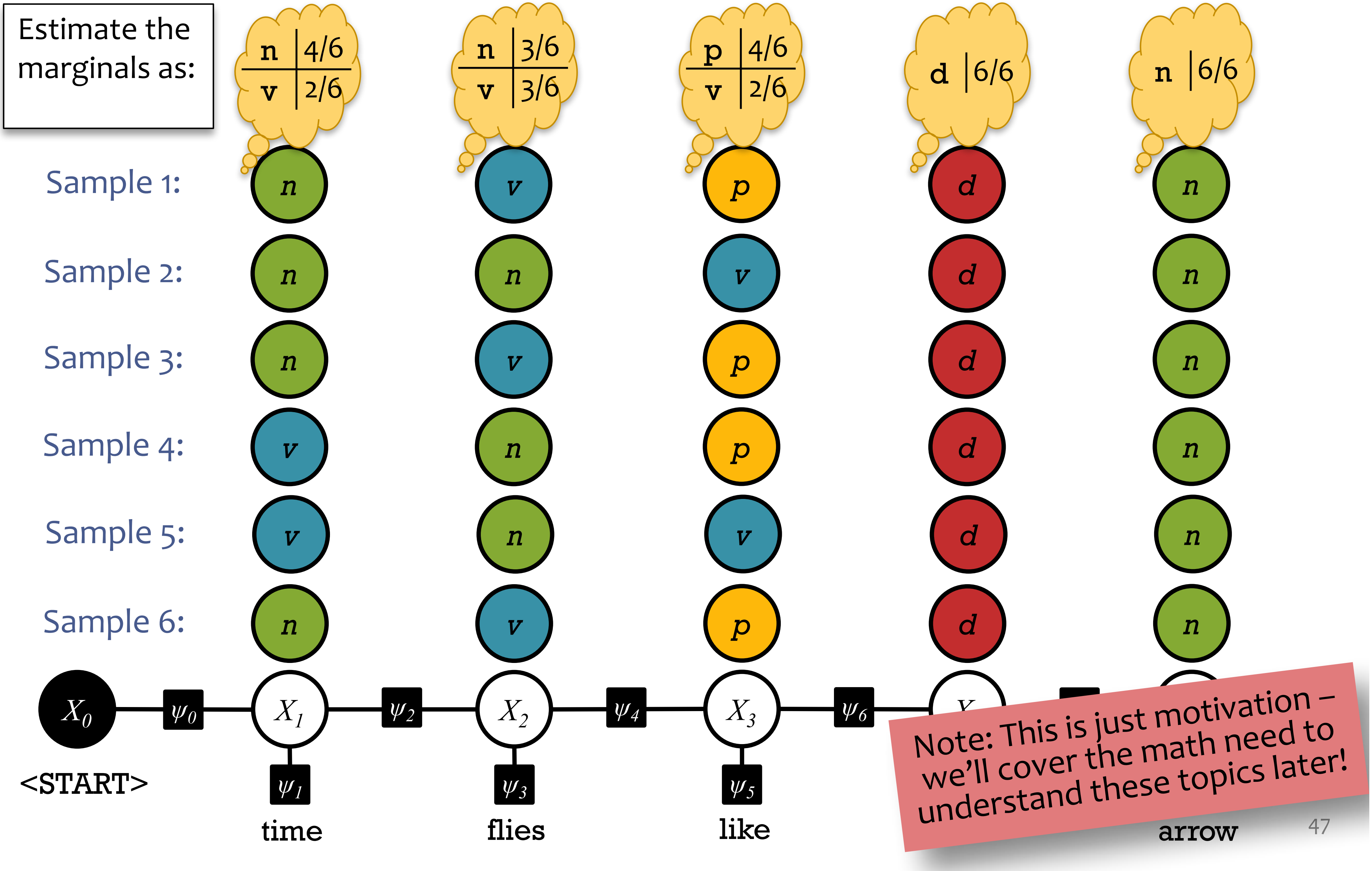
The marginal $p(X_i = x_i)$ gives the probability that variable X_i takes value x_i in a random sample



Sample 1:	*n*	*v*	*p*	*d*	*n*
Sample 2:	*n*	*n*	*v*	*d*	*n*
Sample 3:	*n*	*v*	*p*	*d*	*n*
Sample 4:	*v*	*n*	*p*	*d*	*n*
Sample 5:	*v*	*n*	*v*	*d*	*n*
Sample 6:	*n*	*v*	*p*	*d*	*n*

Note: This is just motivation – we'll cover the math need to understand these topics later!

Marginals by Sampling on Factor Graph



Note: This is just motivation – we'll cover the math need to understand these topics later!

Why Computer Science for ML?

10-607

To best understand __A__ we need __B__

A	B
Analysis of Exact Inference in Graphical Models	Computation • Computational Complexity • Recursion; Dynamic Programming • Data Structures for ML Algorithms
Implementation Design of a Deep Learning Library	Programming & Efficiency • Debugging for Machine Learning • Efficient Implementation / Profiling ML Algorithms

Finite Difference Method

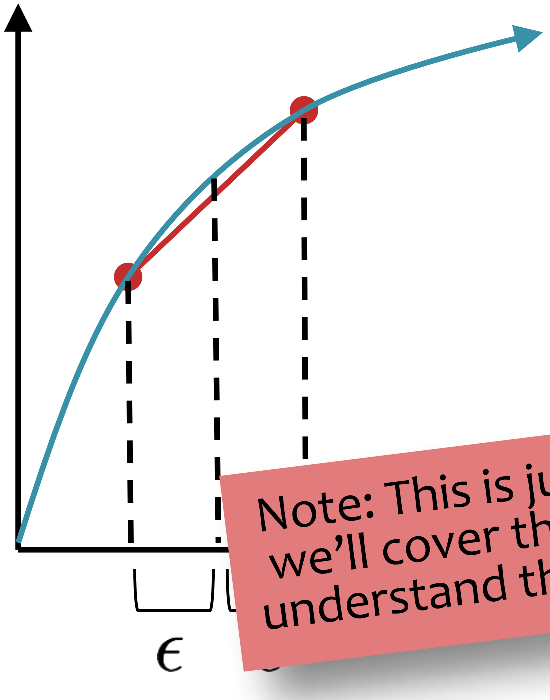
The *centered* finite difference approximation is:

$$\frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta}) \approx \frac{(J(\boldsymbol{\theta} + \epsilon \cdot \boldsymbol{d}_i) - J(\boldsymbol{\theta} - \epsilon \cdot \boldsymbol{d}_i))}{2\epsilon} \tag{1}$$

where $\boldsymbol{d}_i$ is a 1-hot vector consisting of all zeros except for the ith entry of $\boldsymbol{d}_i$, which has value 1.

Notes:

- Suffers from issues of floating point precision, in practice
- Typically only appropriate to use on small examples with an appropriately chosen epsilon



Note: This is just motivation – we'll cover the math need to understand these topics later!

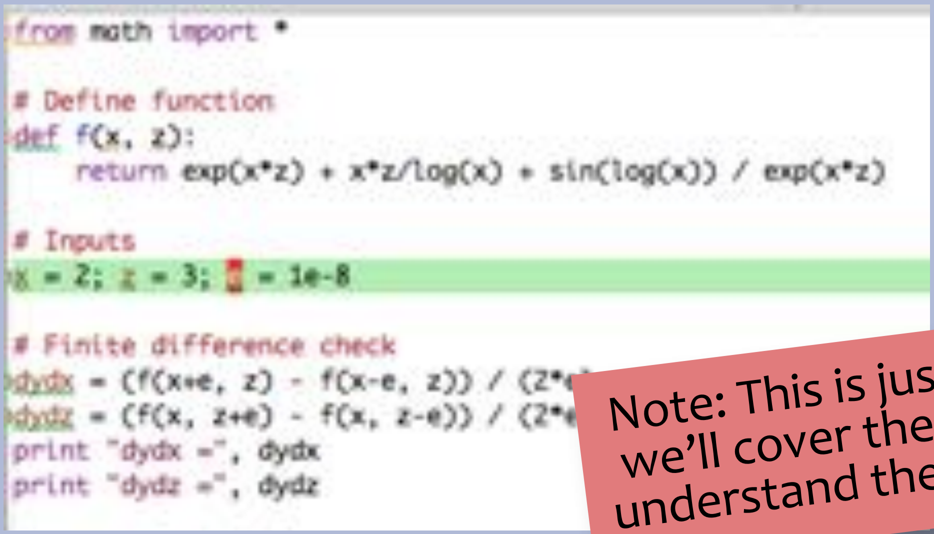
Differentiation

Chain Rule Quiz #1:

Suppose x = 2 and z = 3, what are dy/dx and dy/dz for the function below?

$$y = \exp(xz) + \frac{xz}{\log(x)} + \frac{\sin(\log(x))}{\exp(xz)}$$

Finite Difference Solution:



```
from math import *

# Define function
def f(x, z):
    return exp(x*z) + x*z/log(x) + sin(log(x)) / exp(x*z)

# Inputs
x = 2; z = 3; e = 1e-8

# Finite difference check
dydx = (f(x+e, z) - f(x-e, z)) / (2*e
dydz = (f(x, z+e) - f(x, z-e)) / (2*e
print "dydx =", dydx
print "dydz =", dydz
```

Note: This is just motivation – we'll cover the math need to understand these topics later!

Backpropagation

Automatic Differentiation – Reverse Mode (aka. Backpropagation)

Forward Computation

1. Write an **algorithm** for evaluating the function y = f(**x**). The algorithm defines a **directed acyclic graph,** where each variable is a node (i.e. the "**computation graph**")
2. Visit each node in **topological order.**
   For variable u_i with inputs $v_1, \ldots, v_N$
   a. Compute $u_i = g_i(v_1, \ldots, v_N)$
   b. Store the result at the node

Backward Computation

1. **Initialize** all partial derivatives dy/du_j to 0 and dy/dy = 1.
2. Visit each node in **reverse topological order.**
   For variable $u_i = g_i(v_1, \ldots, v_N)$
   a. We already know dy/du_i
   b. Increment dy/dv_j by $(dy/du_i)(du_i/dv_j)$
   (Choice of algorithm ensures computing (du_i/dv_j) i

Return partial derivatives dy/du_i for all variables

Note: This is just motivation – we'll cover the math need to understand these topics later!

Why Computer Science for ML?

10-607

To best understand __A__ we need __B__

A	B
Analysis of Exact Inference in Graphical Models	Computation • Computational Complexity • Recursion; Dynamic Programming • Data Structures for ML Algorithms
Implementation Design of a Deep Learning Library	Programming & Efficiency • Debugging for Machine Learning • Efficient Implementation / Profiling ML Algorithms
Optimization for Support Vector Machines (SVMs)	Optimization • Unconstrained Optimization • Preconditioning • Constrained Optimization

Support Vector Machines (SVMs)

Hard-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \ldots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

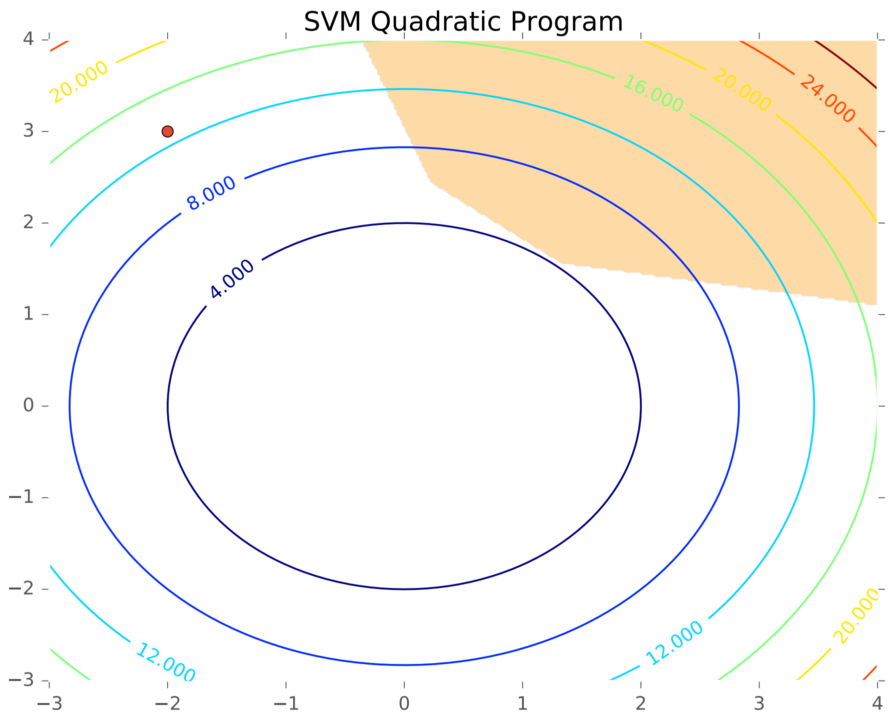
$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

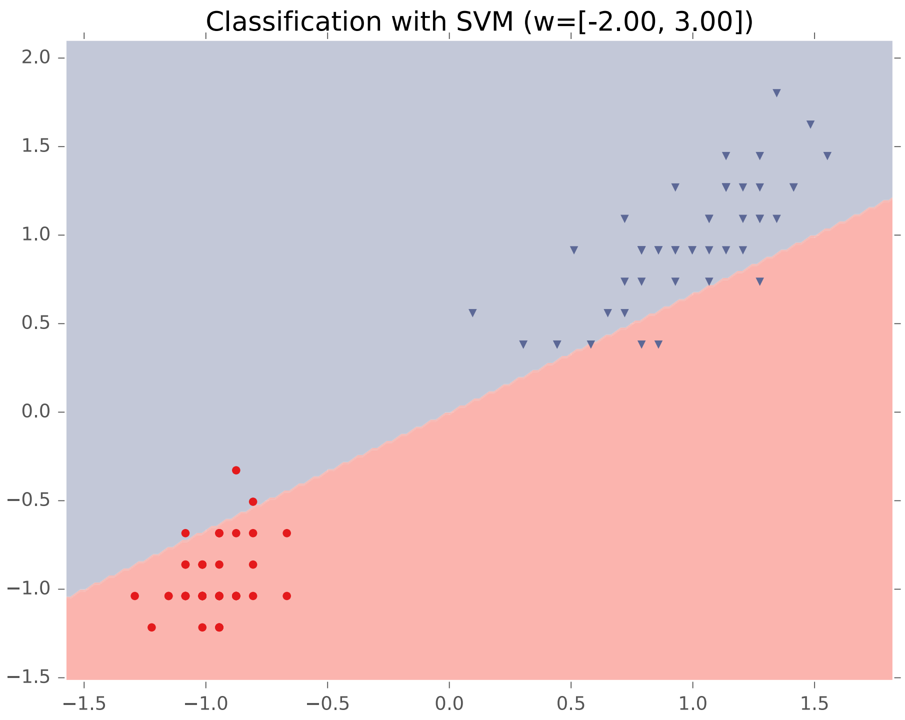
$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

- Instead of minimizing the primal, we can maximize the dual problem
- For the SVM, these two problems give the same answer (i.e. the minimum of one is the maximum of the other)
- *Definition*: **support vectors** are those which $\alpha^{(i)} \neq 0$

Note: This is just motivation – we'll cover the math need to understand these topics later!

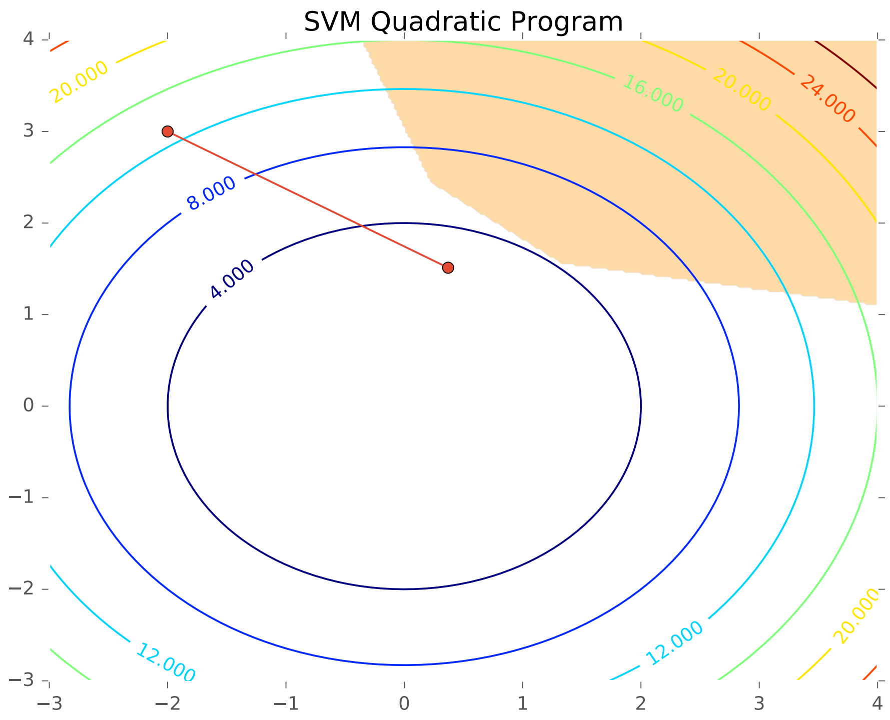
SVM QP

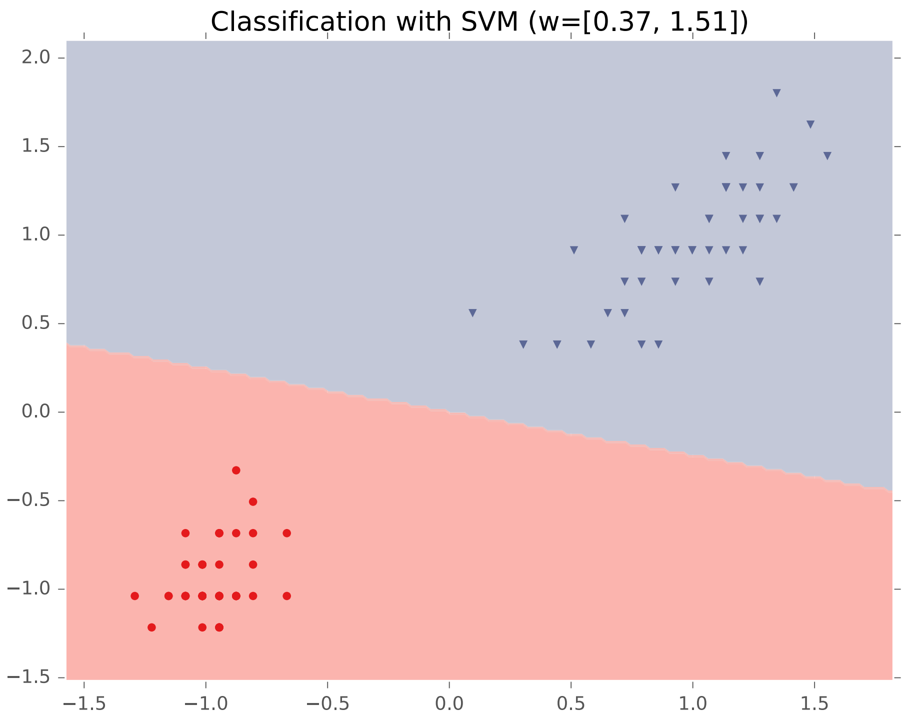

SVM Quadratic Program


Classification with SVM (w=[-2.00, 3.00])

Note: This is just motivation – we'll cover the math need to understand these topics later!

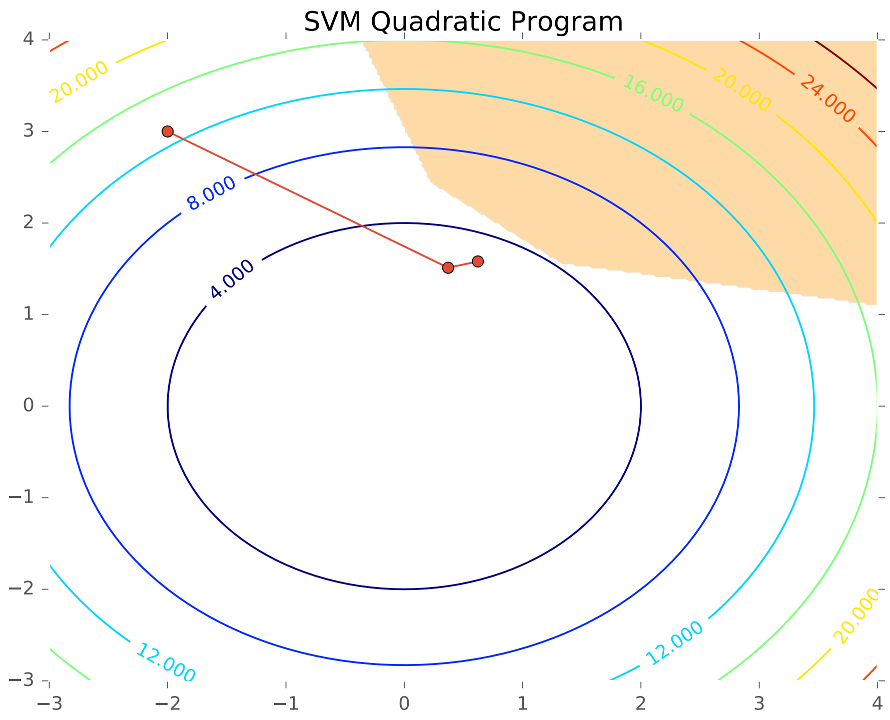
SVM QP

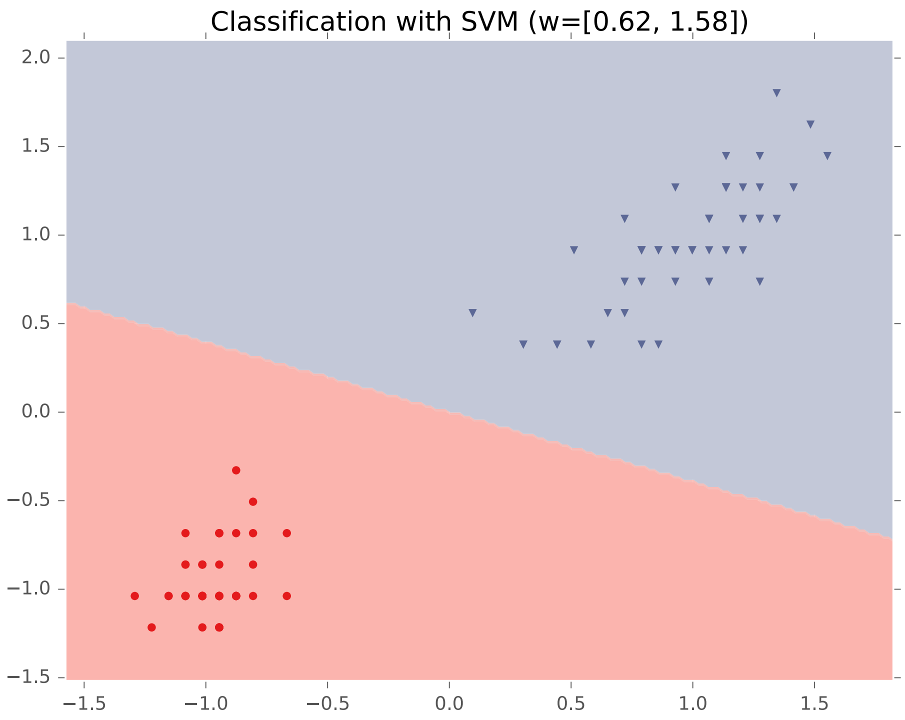




Note: This is just motivation – we'll cover the math need to understand these topics later!

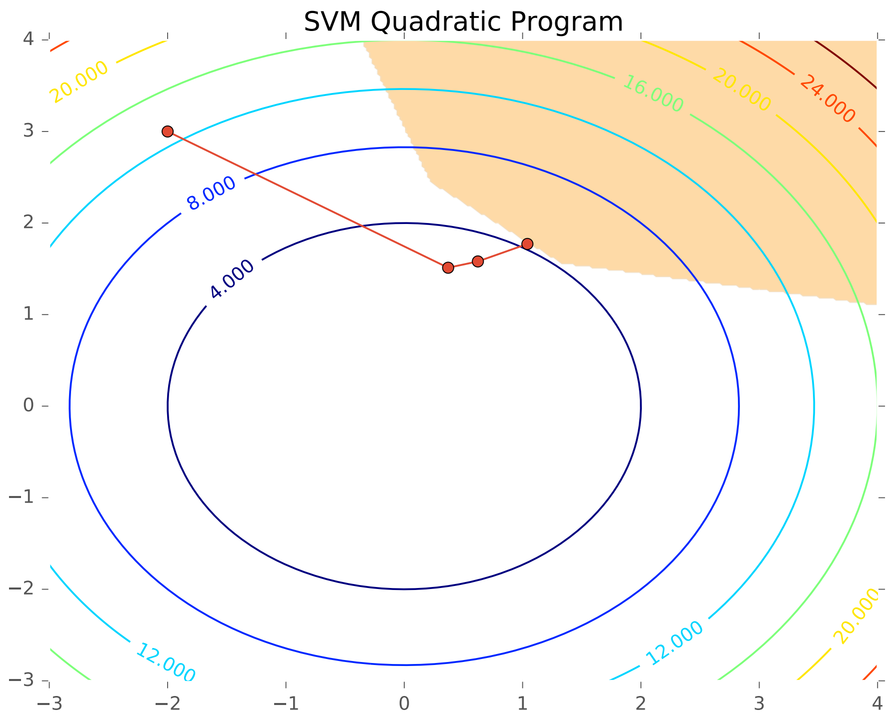
SVM QP

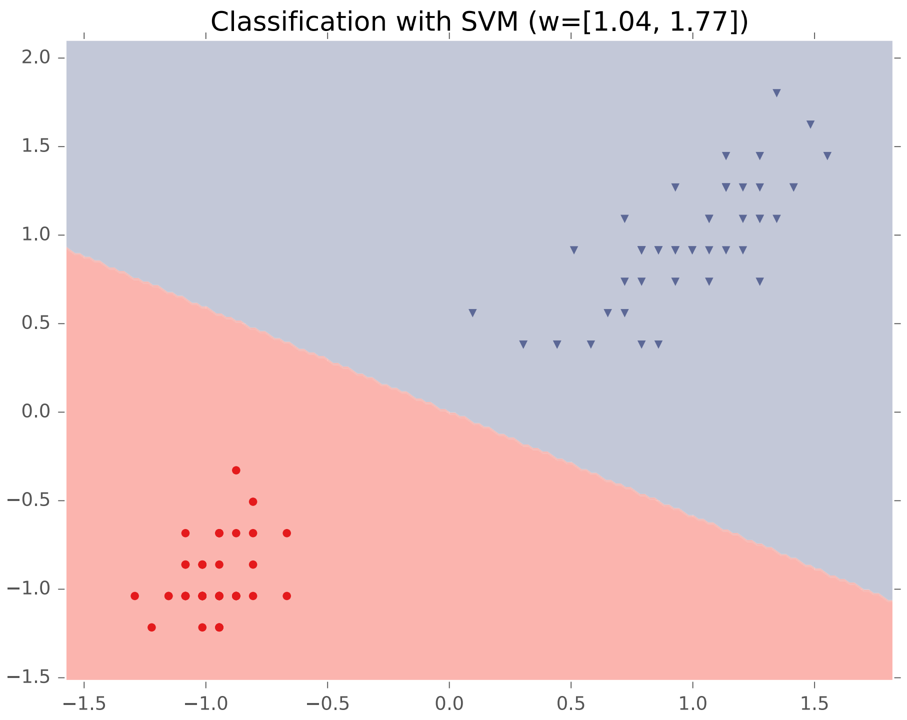




Note: This is just motivation – we'll cover the math need to understand these topics later!

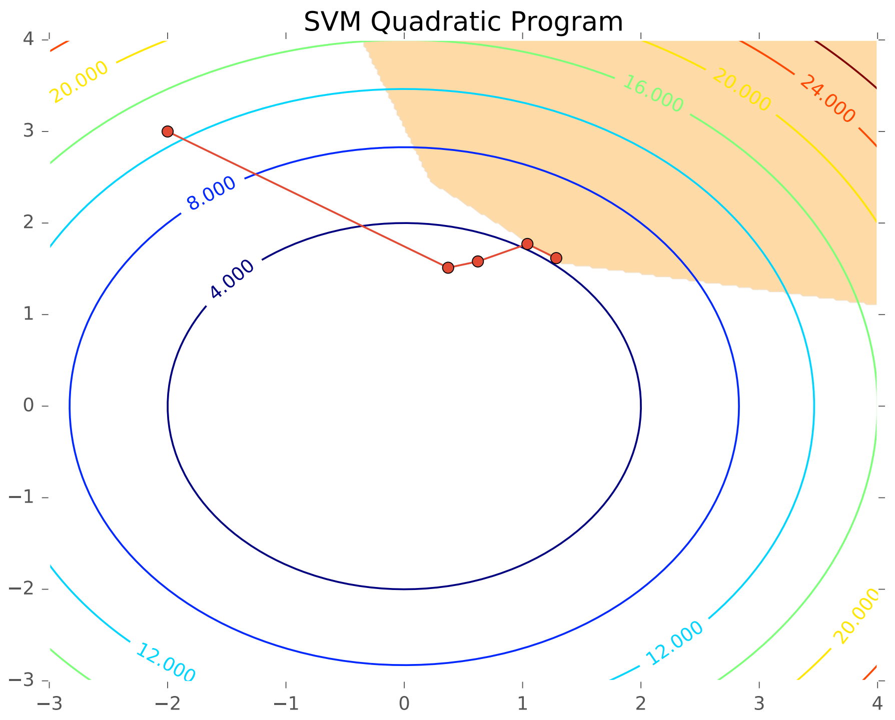
SVM QP

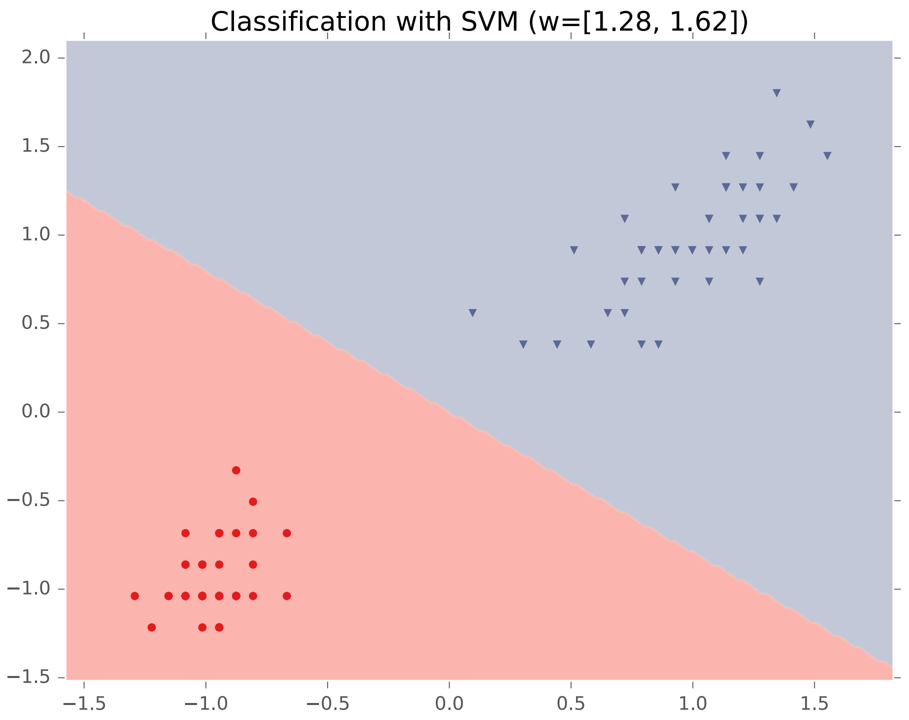

SVM Quadratic Program


Classification with SVM (w=[1.04, 1.77])

Note: This is just motivation – we'll cover the math need to understand these topics later!

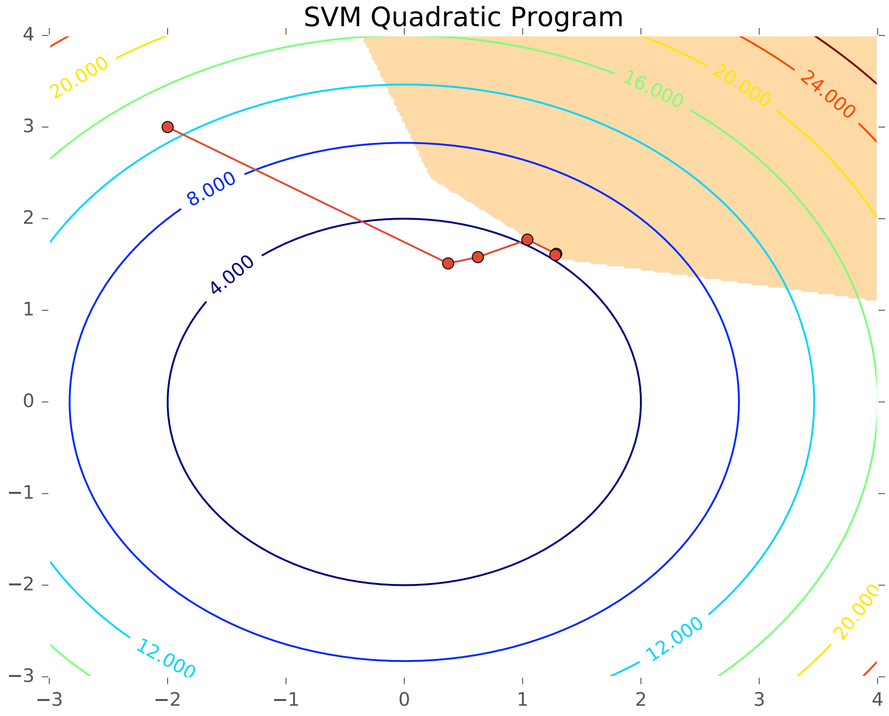
SVM QP

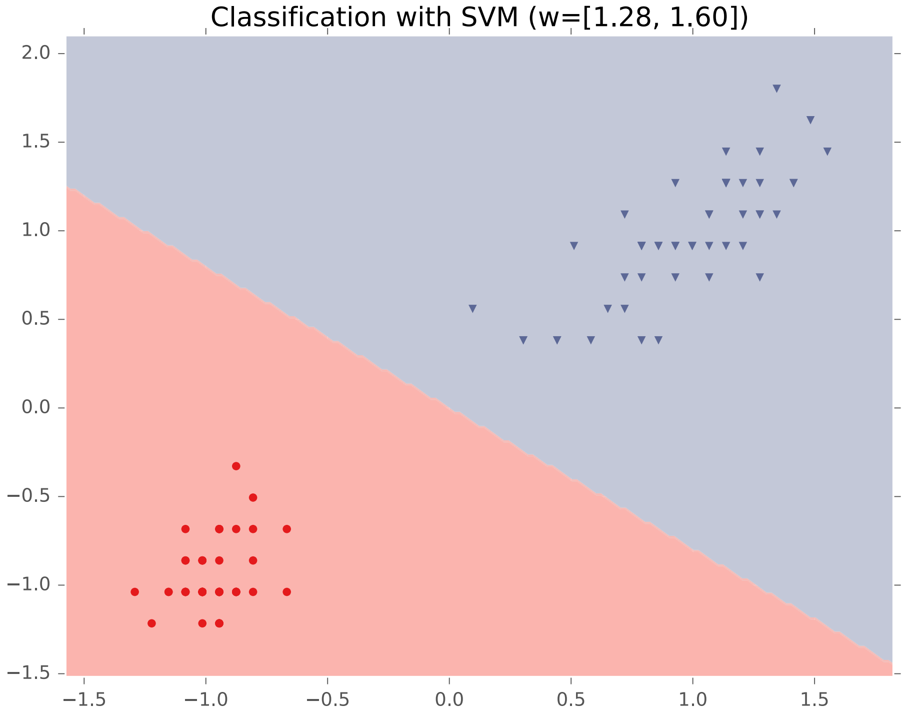




Note: This is just motivation – we'll cover the math need to understand these topics later!

SVM QP





Note: This is just motivation – we'll cover the math need to understand these topics later!

Why Computer Science for ML?

To best understand __A__ we need __B__

A	B
Analysis of Exact Inference in Graphical Models	Computation • Computational Complexity • Recursion; Dynamic Programming • Data Structures for ML Algorithms
Implementation Design of a Deep Learning Library	Programming & Efficiency • Debugging for Machine Learning • Efficient Implementation / Profiling ML Algorithms
Optimization for Support Vector Machines (SVMs)	Optimization • Unconstrained Optimization • Preconditioning • Constrained Optimization

The core content for this course is the **computer science** (Column B), but you will apply what you learn to **real problems in machine learning** (Column A)

Why Computer Science for ML?

10-607

To best understand __A__ we need __B__

A	B
Analysis of Exact Inference in Graphical Models	Computation • m Rec • Dat
Implementation Design of a Deep Learning Library	Pr a Deb • Effic
Optimization for Support Vector Machines (SVMs)	Op i Unc • Preconditioning • Constrained Optimization

Note:

You may want to take the 10-607 schedule with a grain of salt given Matt's propensity in 10-606 for swapping in new applications as the semester progressed.

The core content for this course is the **computer science** (Column B), but you will apply what you learn to **real problems in machine learning** (Column A)

SYLLABUS HIGHLIGHTS

Syllabus Highlights

The syllabus is located on the course webpage:

http://www.cs.cmu.edu/~mgormley/courses/606-607-f18

The **course policies** are **required** reading.

606/607 Syllabus Highlights

- **Grading**: 55% homework, 10% in-class quizzes, 30% final exam, 5% participation
- **Final Exam**:
  - 606: Mini-I final exam week, date TBD
  - 607: Mini-II final exam week, date TBD
- **In-Class Quizzes**: always announced ahead of time
- **Homework**: 4 assignments with written / programming portions
  - 2 grace days for the unexpected
  - Late submissions: 80% day 1, 60% day 2, 40% day 3, 20% day 4
  - No submissions accepted after 4 days w/o extension
  - Extension requests: see syllabus
- **Recitations**: Fridays, same time/place as lecture (optional, interactive sessions)
- **Readings**: required, online, recommended for after lecture
- **Technologies**: Piazza (discussion), Gradescope (homework), Canvas (gradebook only)
- **Academic Integrity**:
  - Collaboration encouraged, but must be documented
  - Solutions must always be written independently
  - No re-use of found code / past assignments
  - Severe penalties (i.e. failure)
- **Office Hours**: posted on Google Calendar on “People” page

606/607 Syllabus Highlights

- **Grading:** 55% homework, 10% in-class quizzes, 30% final exam, 5% participation
- **Final Exam:**
  - 606: Mini-I final exam week, date TBD
  - 607: Mini-II final exam week, date TBD
- **In-Class Quizzes:** always announced ahead of time
- **Homework:** 4 assignments with written / programming portions
  - 2 grace days for the unexpected
  - Late submissions: 80% day 1, 60% day 2, 40% day 3, 20% day 4
  - No submissions accepted after 4 days w/o extension
  - Extension requests: see syllabus
- **Recitations:** Fridays, same time/place as lecture (optional, interactive sessions)
- **Readings:** required, online, recommended for after lecture
- **Technologies:** Piazza (discussion), Gradescope (homework), Canvas (gradebook only)
- **Academic Integrity:**
  - Collaboration encouraged, but must be documented
  - Solutions must always be written independently
  - No re-use of found code / past assignments
  - Severe penalties (i.e. failure)
- **Office Hours:** posted on Google Calendar on “People” page

Lectures

- You should ask lots of questions
  - Interrupting (by raising a hand) to ask your question is strongly encouraged
  - Asking questions later on Piazza is also great
- When I ask a question…
  - I want you to answer
  - Even if you don’t answer, think it through as though I’m about to call on you
- Interaction improves learning (both in-class and at my office hours)

Expected Background

10-606 (Math Background 4 ML)

You should be familiar with some of the following…

- Calculus:
  - can take scalar derivatives
  - can solve scalar integrals
- Linear Algebra:
  - know basic vector operations
  - seen matrix multiplication
- Probability:
  - seen the basics: conditioning, Bayes Rule, etc.
- Programming:
  - know some Python
    OR
    have sufficient programming background to pick up the basics of Python

But we'll offer practice to make sure you can catch up on your weaker areas

10-607 (CS Background 4 ML)

You should…

- be comfortable with all the topics listed for 10-606
- ideally, have the mathematical maturity of someone who completed 10-606 **because** it will aide in understanding the motivating examples from machine learning

That said, the content of 10-607 is designed stand alone

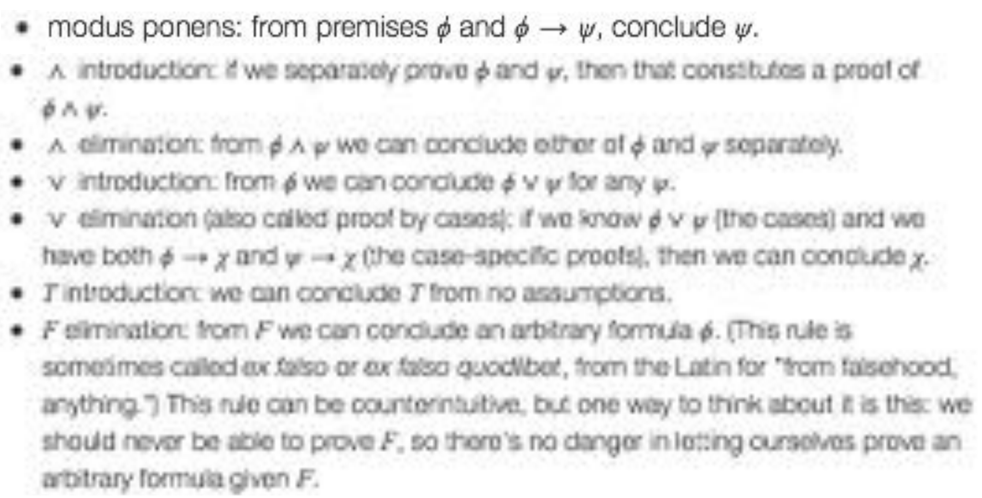
LOGIC

Propositional Logic

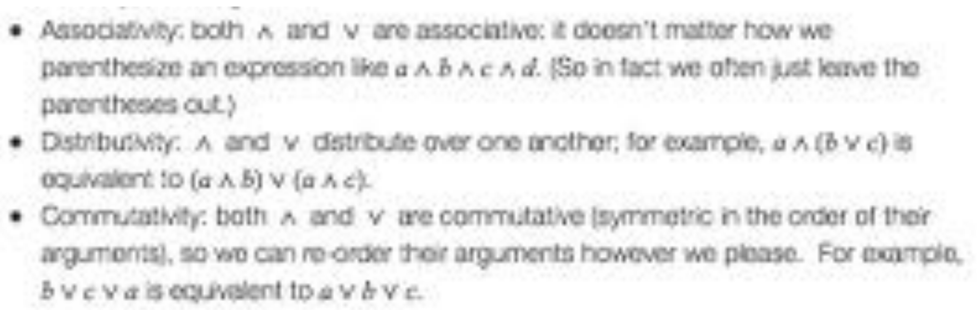
Chalkboard

- Form of arguments
- Components of propositional logic
- Two-column proofs
- *modus ponens*
- Inference rules
- Lemmas

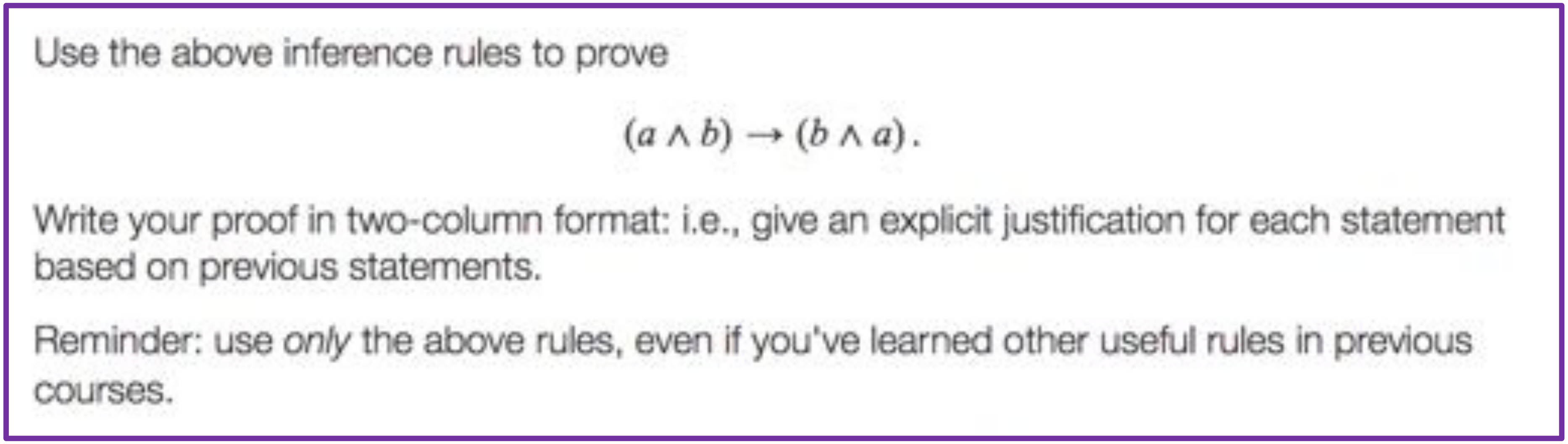
Exercise: Inference Rules



- modus ponens: from premises ϕ and $\phi \to \psi$, conclude ψ.
- $\wedge$ introduction: if we separately prove ϕ and ψ, then that constitutes a proof of $\phi \wedge \psi$.
- $\wedge$ elimination: from $\phi \wedge \psi$ we can conclude either of ϕ and ψ separately.
- $\vee$ introduction: from ϕ we can conclude $\phi \vee \psi$ for any ψ.
- $\vee$ elimination (also called proof by cases): if we know $\phi \vee \psi$ (the cases) and we have both $\phi \to \chi$ and $\psi \to \chi$ (the case-specific proofs), then we can conclude χ.
- T introduction: we can conclude T from no assumptions.
- F elimination: from F we can conclude an arbitrary formula ϕ. (This rule is sometimes called *ex falso* or *ex falso quodlibet*, from the Latin for "from falsehood, anything.") This rule can be counterintuitive, but one way to think about it is this: we should never be able to prove F, so there's no danger in letting ourselves prove an arbitrary formula given F.



- Associativity: both $\wedge$ and $\vee$ are associative: it doesn't matter how we parenthesize an expression like $a \wedge b \wedge c \wedge d$. (So in fact we often just leave the parentheses out.)
- Distributivity: $\wedge$ and $\vee$ distribute over one another; for example, $a \wedge (b \vee c)$ is equivalent to $(a \wedge b) \vee (a \wedge c)$.
- Commutativity: both $\wedge$ and $\vee$ are commutative (symmetric in the order of their arguments), so we can re-order their arguments however we please. For example, $b \vee c \vee a$ is equivalent to $a \vee b \vee c$.



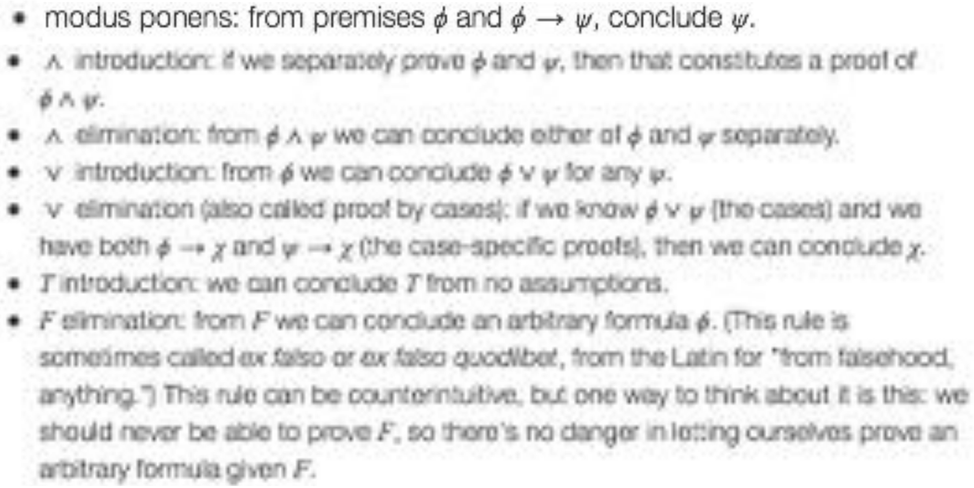
Use the above inference rules to prove

$$(a \wedge b) \to (b \wedge a).$$

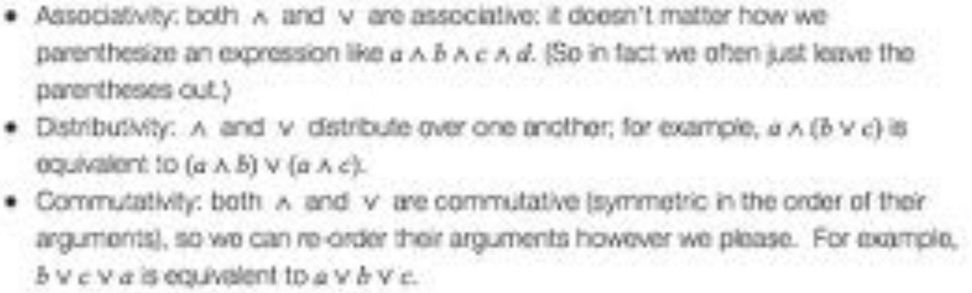
Write your proof in two-column format: i.e., give an explicit justification for each statement based on previous statements.

Reminder: use *only* the above rules, even if you've learned other useful rules in previous courses.

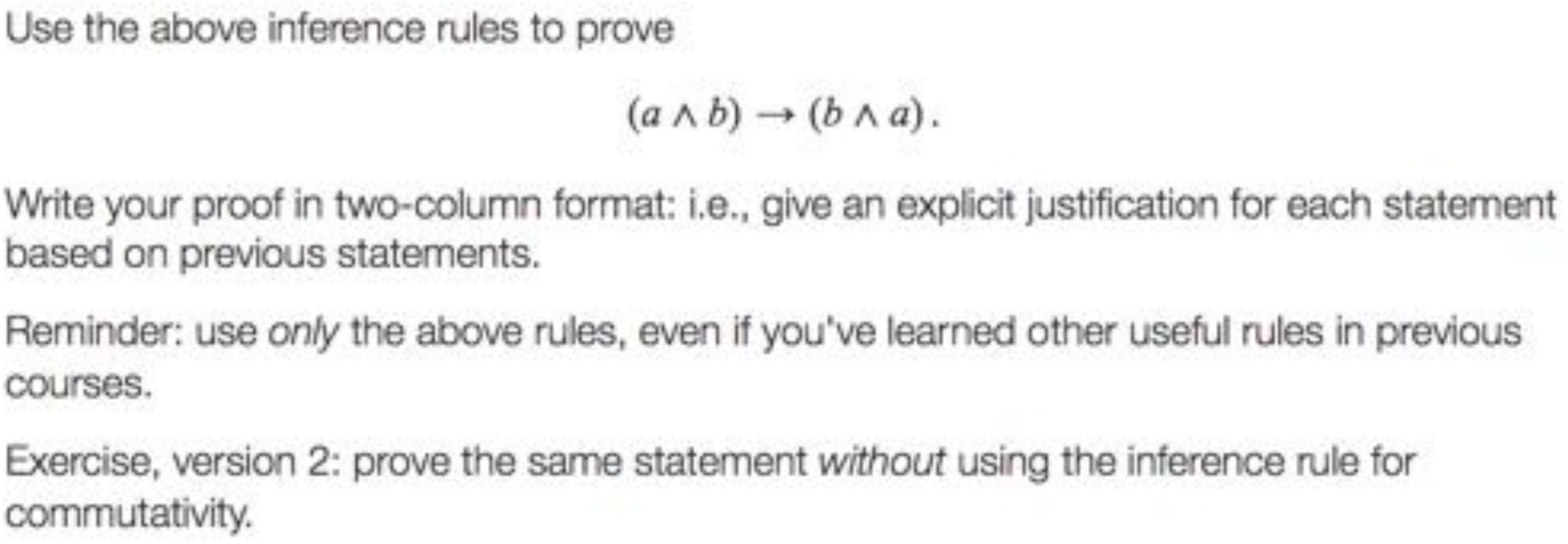
Exercise: Inference Rules



- modus ponens: from premises ϕ and $\phi \to \psi$, conclude ψ.
- $\wedge$ introduction: if we separately prove ϕ and ψ, then that constitutes a proof of $\phi \wedge \psi$.
- $\wedge$ elimination: from $\phi \wedge \psi$ we can conclude either of ϕ and ψ separately.
- $\vee$ introduction: from ϕ we can conclude $\phi \vee \psi$ for any ψ.
- $\vee$ elimination (also called proof by cases): if we know $\phi \vee \psi$ (the cases) and we have both $\phi \to \chi$ and $\psi \to \chi$ (the case-specific proofs), then we can conclude χ.
- T introduction: we can conclude T from no assumptions.
- F elimination: from F we can conclude an arbitrary formula ϕ. (This rule is sometimes called *ex falso* or *ex falso quodlibet*, from the Latin for "from falsehood, anything.") This rule can be counterintuitive, but one way to think about it is this: we should never be able to prove F, so there's no danger in letting ourselves prove an arbitrary formula given F.



- Associativity: both $\wedge$ and $\vee$ are associative: it doesn't matter how we parenthesize an expression like $a \wedge b \wedge c \wedge d$. (So in fact we often just leave the parentheses out.)
- Distributivity: $\wedge$ and $\vee$ distribute over one another; for example, $a \wedge (b \vee c)$ is equivalent to $(a \wedge b) \vee (a \wedge c)$.
- Commutativity: both $\wedge$ and $\vee$ are commutative (symmetric in the order of their arguments), so we can re-order their arguments however we please. For example, $b \vee c \vee a$ is equivalent to $a \vee b \vee c$.



Use the above inference rules to prove

$$(a \wedge b) \to (b \wedge a)\,.$$

Write your proof in two-column format: i.e., give an explicit justification for each statement based on previous statements.

Reminder: use *only* the above rules, even if you've learned other useful rules in previous courses.

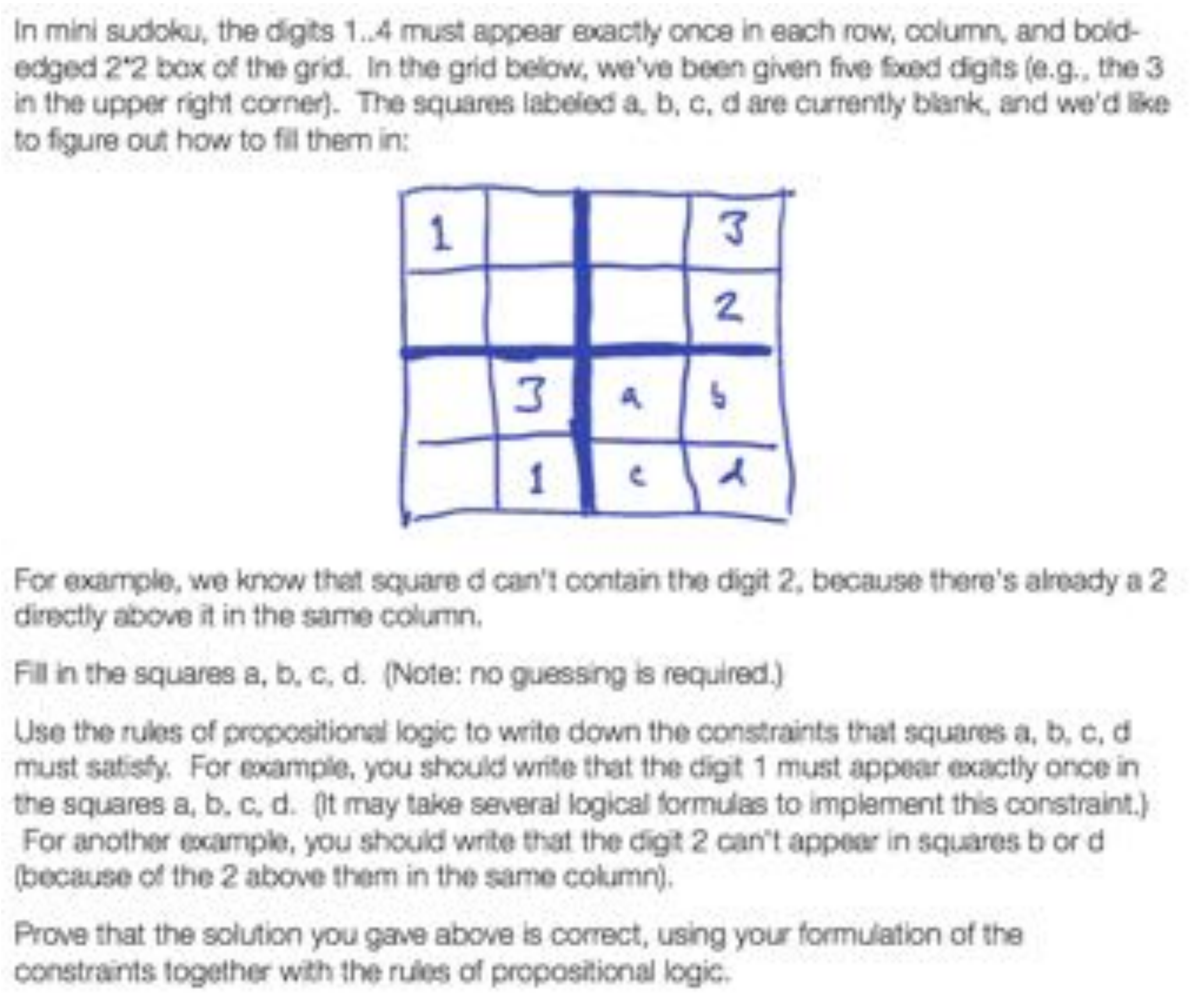
Exercise, version 2: prove the same statement *without* using the inference rule for commutativity.

Classical Logic

Chalkboard

- – Negation and constructive logic
- – Law of the extended middle
- – DeMorgan’s laws
- – Double negation elimination
- – Contraposition
- – Resolution
- – Scoping rules

Exercise: Mini-Sudoku



In mini sudoku, the digits 1..4 must appear exactly once in each row, column, and bold-edged 2*2 box of the grid. In the grid below, we've been given five fixed digits (e.g., the 3 in the upper right corner). The squares labeled a, b, c, d are currently blank, and we'd like to figure out how to fill them in:

1			3
			2
	3	a	b
	1	c	d

For example, we know that square d can't contain the digit 2, because there's already a 2 directly above it in the same column.

Fill in the squares a, b, c, d. (Note: no guessing is required.)

Use the rules of propositional logic to write down the constraints that squares a, b, c, d must satisfy. For example, you should write that the digit 1 must appear exactly once in the squares a, b, c, d. (It may take several logical formulas to implement this constraint.) For another example, you should write that the digit 2 can't appear in squares b or d (because of the 2 above them in the same column).

Prove that the solution you gave above is correct, using your formulation of the constraints together with the rules of propositional logic.

Proof Techniques

Chalkboard

- Proof by Construction
- Proof by Cases
- Proof by Contradiction
- Proof by Induction
- Proof by Contraposition