An Introduction to Statistical Learning

What this is

- **the following are notes from Ch1 from ISLRv2**
- **Chapter 1: Introduction**
- **pp 1-6**
- ****/Spring_2026/DATA780/Unit1/ISLRv2_corrected_June_2023.pdf**

An Overview of Statistical Learning

Statistical learning is a collection of **methods** used to **understand,** discover **patterns,** and make **predictions** in data and is widely used in business, medicine, science [sic], and public policy applications.`

What is Statistical Learning?

Supervised Learning	• used to **predict** or **estimate** the **output** (Y) using **inputs** (X) • Common tasks include predicting **numerical** and **categorical** values and classifications
Unsupervised Learning	• only **inputs** are observed • there is **no output** variable • used to **discover structure** or **relationships** in the data • common tasks include **clustering** and **dimension reduction**

Regression Example: Wage Data

Goal	• **predict** a man's **wage** using information such as age, education level, and calendar year
Key Observations	• wages **increase with age** up to age 60, **then decreases** • wages **increase over time** • higher **education** is associated with **higher wages** • there is **substantial variability** in wages
Insights	• no single variable predicts wage well on its own • combining **multiple variables improves prediction** • relationships can be non-linear (especially with age)

Classification Example: Stock Market Data

Goal	• **predict trends** in stock market
Data Used	• daily percentage changes in the S&P 500 • past 5 days of returns

Key Considerations	• the **output** is **categorical**, not numerical • past returns show very weak predictive power
Results	• prediction accuracy of ~60% using statistical learning methods
Unsupervised Learning Example: Gene Expression	
Goal	• identify groups of similar cancer cell lines using gene expression data
Data Used	• 64 cancer cell lines • 6,830 gene measurements per cell line • no output variable
Approach	• Reduce thousands of variables to a small number (dimension reduction) • visualize the data in two dimensions • look for natural groupings/clusters
Result	• cell lines form visible clusters • clusters tend to correspond to actual cancer types • cancer type information was not used to form the clusters • evidence provided that unsupervised learning can uncover real structure
Supervised Learning: A Brief History	
Early Developments	• 1800s: least squares >> linear regression • 1936: linear discriminant analysis • 1940s: logistic regression • 1970s: generalized linear models
Major Advancements	• early methods were restricted by computational limits • 1980s onward computing power enabled non-linear methods
Modern Era	• statistical learning is a distinct field • methods have become widely available through tools such as R • expanded far beyond statistics and computer science

NumPy: Broadcasting

What this is

- the following are notes from the NumPy v2.4 manual
- **https://numpy.org/doc/stable/user/basics.broadcasting.html**

An Overview of NumPy Broadcasting

What Broadcasting is	• a rule NumPy uses to let you do **arithmetic** on **arrays** with **different shapes**
What Broadcasting does	• when NumPy sees two arrays that do **not match shape** exactly it attempts to **expand** the **smaller array**
Result	• NumPy **avoids loops** and unnecessary data copying, making array math **fast** and **efficient**

Why Broadcasting Matters

NumPy without Broadcasting

- normally NumPy does element-by-element operations which traditionally only work with array with the same shape

```
a = np.array([1, 2, 3])
b = np.array([2, 2, 2])
a * b          # Works
```

NumPy with Broadcasting

- let's NumPy **relax** the **constraints**
- facilitates **scalar** and **non-conformable** array arithmetic

Scalar Multiplication Example

Problem

- a problem requires **multiplying** a 3 x 1 array by a **scalar**

```
a = np.array([1, 2, 3])
b = 2
a * b
```

Solution

- NumPy conceptually "**stretches**" 'b' to **match** the shape of a
- this is achieved **without duplicating** the scalar in **memory**
- essentially makes the operation:

```
[1 * 2, 2 * 2, 3 * 2]
```

How Broadcasting Works

Rules

- compare shapes from the **trailing** (rightmost) **dimension**
- two dimensions are **compatible** if
  - they are **equal**
  - one of them **equals** ‘**1**‘
- if **neither** condition is **true**
  - NumPy raises ‘**ValueError**’

What it does

- If **one array** has **fewer dimensions**, NumPy **treats** missing dimension sizes **as ‘1’**
- If a **dimension** is **‘1’** in one array but **larger** in the **other**, NumPy **expands** that dimension

Resulting Array

- the resulting shape has the **maximum** number of **dimensions** from the **input arrays**
- each **axis** is the **size** of the **larger** of the two
- if shapes **cannot** be **aligned** NumPy raises an **error**

Examples

- NumPy **expands** the dimensions of ‘**B**’ so it is compatible with ‘A’

```
Array A shape: (8, 1, 6, 1)
Array B shape:     (7, 1, 5)
Result shape: (8, 7, 6, 5)
```

- when operation is with scalar and array ‘**b**’ is **broadcasted** into ‘a’

```
a = np.array([1, 2, 3])
b = 2
a * b              # Works by broadcasting b
```

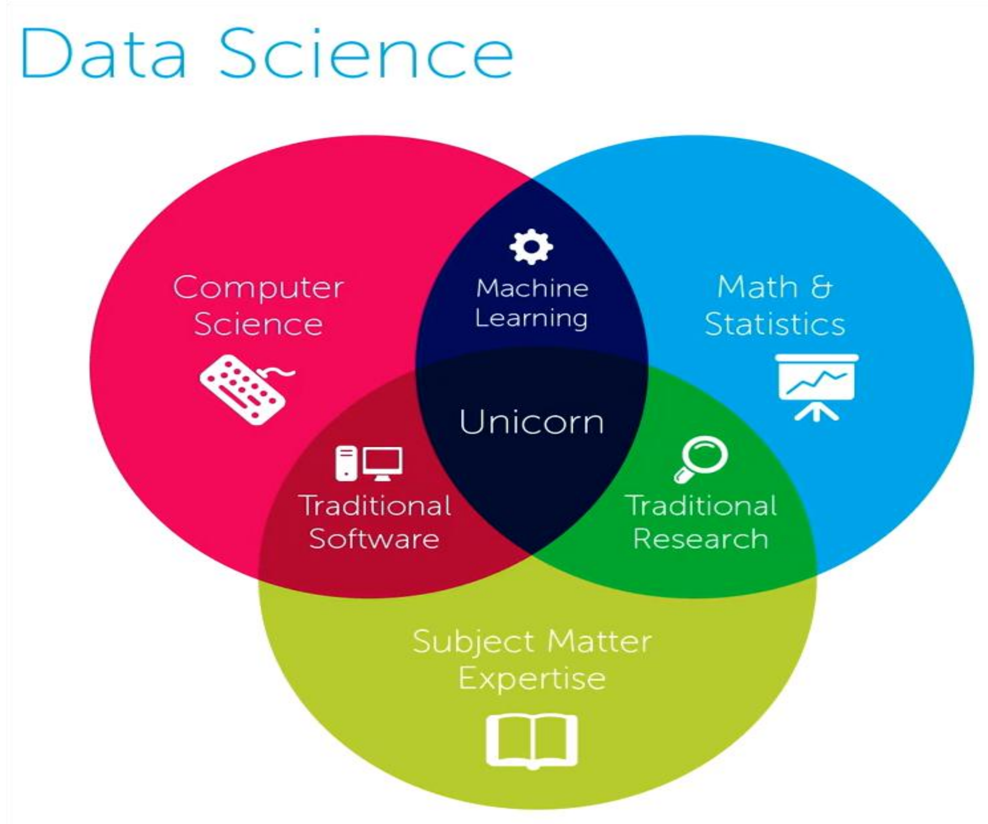
- when operation is with 1-D and 2-D arrays

```
a = np.array([[0, 0, 0],
              [10,10,10],
              [20,20,20],
              [30,30,30]])

b = np.array([1,2,3])

a + b   # b stretches across each row of a
```

	• a set of arrays are **broadcast-able** if all can be expanded to a **common shape** that **meets** the **rules** • `Shapes: (5,1), (1,6), (6,), ()  → all can broadcast to (5,6)` • scalar shape '()' acts like a shape of **ones** • the **1-D array** '(6,) is treated like **(1, 6)** • all **arrays** are conceptually **expanded** to (**5, 6**) • this is all achieved with **minimal memory** use by NumPy
When Broadcasting Fails	• when two **arrays** do **not** satisfy the **rules** and broadcasting is **not possible** NumPy shows a broadcasting **error** • `Array shapes: (4,3) and (4,)` `→ Error: cannot broadcast because trailing dimensions don't match`
Broadcasting Summary	• let's NumPy **combine** arrays of **different** shapes avoiding loops and memory bloat • **compares** shapes from **trailing** dimensions • dimensions must either **match** or be '**1**' • if conditions **not met** NumPy raises an **error**

Async Materials

Machine Learning



Machine Learning Terminology

Inputs to Models

- **features** – the individual **measurable properties** in a model
- **covariates –** input variables that are **statistically related** to the output
- **dimensions** – **number** of input variables or **axes** in a feature space
- **parameters –** learnable **variables needed** to compute the model's **output**

Contents of a Dataset

- **instances** – a single **unit of observation** in a dataset
- **samples** – a **single** observed **data point**
- **examples** – a **training** or **testing** data point

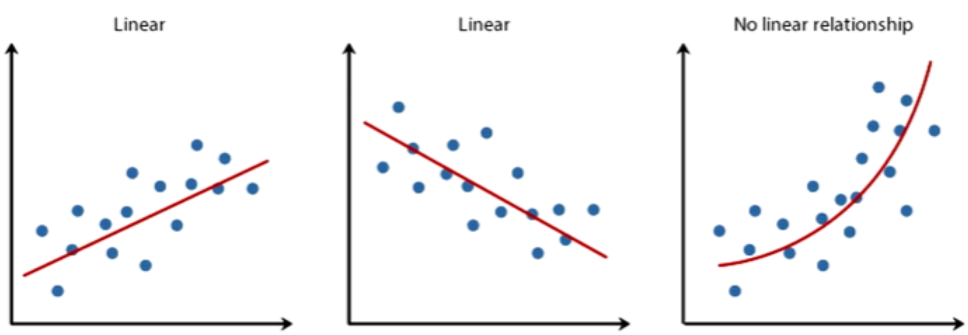
Quick Reference

Concept Type	Term	Emphasis
inputs	**feature**	**measurable** input variable
	covariate	**input** related to **outcome**
	dimension	**number** of input **axes**
dataset unit	**instance**	one **entity** or **case**
	sample	**observed** data point
	example	labeled **training/test** case

-

Training	• **inference** – using understood relationships to draw conclusions
	• **prediction** – using data, machine learning, and statistical modeling to forecast future outcomes
	• **classification label** – predefined category or class assigned to a data point correlating to output in supervised machine learning
	• **types of learning**
	  ○ **supervised** – mapping input to output
	  ○ **unsupervised** – learn data characteristics
	  ○ **reinforcement** – learn how to interact with an environment through sequential 'stacked' learning

Types of Machine Learning	

Supervised Learning	• **inputs** > **outputs**
	  ○ {example1, output1}
	  ○ {example2, output2}
	• speech > text
	• **regression**
	• 

Unsupervised Learning	• does **not** make use of **prespecified**/annotated **examples**
	• {example1, example2, example3…}
	• $\{x_i\}^n_{i=1}$
	• **clustering**
	  ○ attempting to **discover** the salient **groups** of the data
	• **dimensionality reduction**
	  ○ **identifies** the degrees of freedom or **core descriptors** of the data
	• **generative models**
	  ○ given training data, generate new samples from same distribution

Reinforcement Learning	• interact with an environment and assess the environment state to then output actions that change that state

	• predictions build on one another to maximize ‘rewards’ through iterative application of the assessment, action, change state, assess, …
Keys to Success with Machine Learning	
Three Pillars of Machine Learning	• big **datasets** • **fast** processing • **innovative** methods
Typical ML Model Lifecycle	• **select** loss/**model-type** for data task • **optimize** model with training **data** • **evaluate** on held-out data to **validate** choices • use with unseen **future** data
Three Pillars to YOUR Success in Machine Learning	• **statistics** and **mathematics** • computer **engineering** • data **analytics** acumen
Philosophy: Intelligence Can Grow	

<table>
<tr><th colspan="2">Mathematics Review</th></tr>
<tr><td>

Differentiation

</td><td>

taking derivatives of functions

$$\frac{\mathrm{d}}{\mathrm{d}x} f(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Basic Facts

- *Constant rule*: if *f*(*x*) is constant, then
  $f'(x) = 0.$
- *Sum rule*:
  $(\alpha f + \beta g)' = \alpha f' + \beta g'$ for all functions *f* and *g* and all real numbers α and β.
- *Product rule*:
  $(fg)' = f'g + fg'$ for all functions *f* and *g*. As a special case, this rule includes the fact $(\alpha f)' = \alpha f'$ whenever α is a constant, because $\alpha' f = 0 \cdot f = 0$ by the constant rule.
- *Quotient rule*:
  $\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$ for all functions *f* and *g* at all inputs where $g \neq 0$.
- *Chain rule* for composite functions: If $f(x) = h(g(x))$, then
  $f'(x) = h'(g(x)) \cdot g'(x).$

Derivative

</td></tr>
<tr><td>

Gradients

</td><td>

derivatives that take in multiple variables and produce a real value output

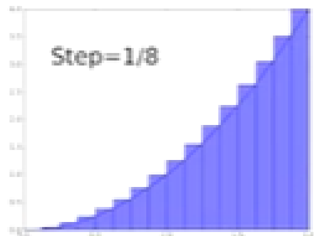
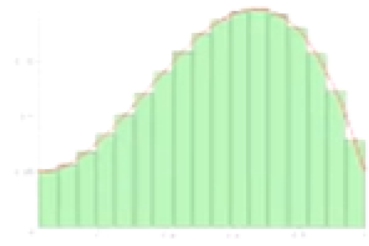
Gradients

$$f : \mathbb{R}^n \mapsto \mathbb{R}$$

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

</td></tr>
</table>

	Example: Gradient $$f(x_1, x_2, x_3) = x_1 x_2^2 + \log(x_3)$$ $$\nabla f(a) = \begin{bmatrix} a_2^2 \\ 2a_1 a_2 \\ \frac{1}{a_3} \end{bmatrix}$$
Derivatives as Approximators	**Derivatives as Approximators** $$f(x+h) \approx f(x) + \nabla f(x)^T h$$

Integration	**Integration as a Limit of Sums**   $$\sum_{i=1}^{n} f(t_i)\,\Delta_i$$ **Integration in Multiple Dimensions** $$\int_A f(x,y)\boxed{\mathrm{d}(x,y)}$$ $$\int_A f(x)\mathrm{d}x,\ x \in \mathbb{R}^n$$  Double integral as volume under a surface $z = 10 - (\frac{x^2 - y^2}{8})$. The rectangular region at the bottom of the body is the *domain* of integration, while the **surface** is the graph of the two-variable function to be integrated. Multiple Integral
Antiderivative	**integration and differentiation are inverse operations** **Antiderivative** $$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = g(x), \text{ then } \int g(x)\mathrm{d}x = f(x) + C$$

Live Session	
Introduction	**05 Jan 2026**
Instructor	Rei Sanchez-Arias
Email	reisanar@unc.edu
Website	https://www.reisanar.com/
Office Hours	Mondays 12:00 pm to 1:00 pm
What to expect	**DATA780**
Meeting Time	Monday 6:00 pm to 7:30 pm

Final Project	project deliverables:
	• project writeup: ~8 pages NeruIPS format
	• open-source repository with executable code for methods developed
	• the github repo can be private or **public**
	• spotlight presentation (last live session)
	  ○ 6 to 7 minutes for presentation
	  ○ 2 minutes for Q&A
	  ○ prepared slide deck
	  ○ optional live demo
	• final project can be combined for DATA780 and DATA740 (if it applies to both)
	• your project may innovate some new LM methodology, or make an improvement to an existing methodology

Final Project Proposal	• state the task, goals • what methods do you plan to use? • what datasets will you consider? • evaluations metrics you plan to use
Assessments	Quizzes will be weekly
Syllabus	https://digitalcampus.instructure.com/courses/55373/assignments/syllabus
Live Session Summary	Meeting Notes found at the top of the Modules page on canvas https://digitalcampus.instructure.com/courses/55373/pages/meeting-notes?module_item_id=9295727

What to expect	DATA780

Resources (free pdf downloads available)	• Mathematics for Machine Learning
	  ○ https://mml-book.github.io/book/mml-book.pdf
	• Data-Driven Science and Engineering
	  ○ https://databookuw.com/
	• Introduction to Statistical Learning
	  ○ https://www.statlearning.com/

Blogs/Websites	**3Blue1Brown Videos (by Grant Sanderson)**
	• Linear Algebra Series
	  ○ https://www.3blue1brown.com/topics/linear-algebra
	• Calculus Series
	  ○ https://www.3blue1brown.com/topics/calculus

TODO	
High Priority	• review linear algebra terminology • finish CW-Data780-unit_01.ipynb

Low Priority	• review assignments
	  ○ Homework 1
	  ○ Final Project