Resources

- Lecture slides are available online.
  - Part1: [link](#)
  - Part2: [link](#)
  - Part3: [link](#)

- Python programming
  - A quick overview of python: [html](#) or [jupyter notebook](#)
  - Google’s python class: [link](#)
  - Pytorch tutorial: [link](#)

- This event is supported by National Science Foundation [NSF](#).
AI VS ML VS Deep Learning

Fig. 1. Image source: NVIDIA blog
Q: What data science methods do you use at work?

Fig. 2. 2017 Kaggle Data Science Survey
ML in Practice

Q: What tools are used at work?

![Bar chart showing the most used tools in data science as of 2017, with Python at 76.3%, R at 59.2%, SQL at 53.6%, Jupyter notebooks at 40.3%, TensorFlow at 28.4%, Amazon Web services at 23.5%, Unix shell/awk at 23.3%, Tableau at 20.4%, C/C++ at 19.2%, NoSQL at 19.2%, MATLAB/Octave at 18.4%, Java at 18.3%, Hadoop/Hive/Pig at 17.3%, Spark/MLib at 17.1%, and Microsoft Excel Data Mining at 13.7%.]

Fig. 3. 2017 Kaggle Data Science Survey
Q: What language would you recommend new data scientists learn first?

Fig. 4. 2017 Kaggle Data Science Survey

- Python: 63.1%
- R: 24.0%
- SQL: 3.5%
- C/C++/C#: 2.8%
- Matlab: 2.2%
- Java: 1.3%
- Scala: 0.9%
- Other: 0.8%
- SAS: 0.8%
- Julia: 0.3%
- Stata: 0.3%
- Haskell: 0.2%
- F#: 0.0%

10,998 responses
Background
1 Background
   ▪ Linear Algebra
   ▪ Probability
   ▪ Calculus
   ▪ ML Overview

2 Linear Regression
   ▪ Motivation
   ▪ Implementation
   ▪ Diagnostics

3 Python Programming
   ▪ Basics
   ▪ Numpy
Notation

- $\mathbb{N}$: set of natural numbers
- $\mathbb{R}$: set of real numbers
- *scalar*: an italicized character, e.g., $a, A_{ij} \in \mathbb{R}$
- *vector*: a lowercase letter in boldface, e.g., $a \in \mathbb{R}^n$
- *matrix*: an uppercase letter in boldface, e.g., $A \in \mathbb{R}^{m \times n}$
- *tensor*: an uppercase letter in sans serif font, e.g., $A \in \mathbb{R}^{m_1 \times m_2 \times m_3}$
Linear Algebra: Vectors

- \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \)

- A vector space \( \mathcal{V} \): a set equipped with two operations: addition and scalar multiplication

\[
(x, y) \in \mathcal{V} \times \mathcal{V} \Rightarrow x + y \in \mathcal{V} \\
(\alpha, x) \in \mathbb{R} \times \mathcal{V} \Rightarrow \alpha x \in \mathcal{V}
\]

- Properties
  - \( \exists \) zero vector \( \mathbf{0} \in \mathcal{V} \) s.t. \( \mathbf{u} + \mathbf{0} = \mathbf{u} \)
  - For \( \forall \mathbf{u} \in \mathcal{V} \), \( \exists -\mathbf{u} \) s.t. \( \mathbf{u} + (-\mathbf{c}) = \mathbf{0} \)
  - \( c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v} \)
Linear Algebra: Matrices

With $m, n \in \mathbb{N}$, a matrix is a rectangular array of numbers $a_{ij}, i = 1, \ldots, m$, $j = 1, \ldots, n$ arranged in rows and columns:

$$A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}$$

- Row vectors and column vectors

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad x' = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$a \in \mathbb{R}^6 \quad a' \in \mathbb{R}^6$$
Image as a matrix: Why using matrices?

Fig. 5. An image is just an array of numbers.
RGB Color model: How about a colored image?

- RGB Color model
  - Express a color as a tuple \((R, G, B)\) of 3 values
  - \(R, G, B \in [0, 255]\)
  - An RGB image has 3 *channels*.

- Other color models
  - CMY, HSV, XYZ
  - ETC
Image as a tensor
Image Representation

\[ x \in \mathbb{R}^{C \times H \times W} \]

\[ x[0] \in \mathbb{R}^{H \times W} \]

\[ x[1] \in \mathbb{R}^{H \times W} \]

\[ x[2] \in \mathbb{R}^{H \times W} \]

\[ x \in \mathbb{R}^{C \cdot H \cdot W} \]
Vector Operations

- **Vector addition**: for \( u, v \in \mathcal{V} \),

\[
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{bmatrix} + \begin{bmatrix}
v_1 \\
v_2 \\
\vdots \\
v_n
\end{bmatrix} = \begin{bmatrix}
u_1 + v_1 \\
u_2 + v_2 \\
\vdots \\
u_n + v_n
\end{bmatrix}
\]

- **Scalar multiplication**: for \( \alpha \in \mathbb{R} \) and \( u \in \mathbb{R}^n \),

\[
\begin{bmatrix}
\alpha u_1 \\
\alpha u_2 \\
\vdots \\
\alpha u_n
\end{bmatrix} = \begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_n
\end{bmatrix}
\]

- **Linear combination**: let \( \alpha, \beta \in \mathbb{R} \) and \( u, v \in \mathcal{V} \).

\[
\begin{bmatrix}
\alpha u_1 \\
\alpha u_2 \\
\vdots \\
\alpha u_n
\end{bmatrix} + \begin{bmatrix}
\beta v_1 \\
\beta v_2 \\
\vdots \\
\beta v_n
\end{bmatrix} = \begin{bmatrix}
\alpha u_1 + \beta v_1 \\
\alpha u_2 + \beta v_2 \\
\vdots \\
\alpha u_n + \beta v_n
\end{bmatrix}
\]
Geometric View

\[ x - y \quad x \quad x + y \]

\[ -y \quad \frac{1}{3}x \quad y \]

\[ 2x \]
Linear Combination

Let $u = (1, 0)^T$ and $v = (1, 2)^T$.

- $a = 1u + 1v$
- $b = -1u + 1v$
- $c = 0u + 2v$
System of Equations

Does the following system of equations have a solution?

\[
\begin{align*}
x - y &= 1 \\
2x + y &= 8 \\
5x - y &= 13
\end{align*}
\]

- Using vector notation, we can rewrite

\[
\begin{bmatrix} 1x \\ 2x \\ 5x \end{bmatrix} + \begin{bmatrix} -1y \\ 1y \\ -1y \end{bmatrix} = x \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix} + y \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ 13 \end{bmatrix}.
\]

- Can \( w \) be expressed as a linear combination of \( u \) and \( v \)?
Transpose

- **Matrix transpose**: for $A \in \mathbb{R}^{m \times n}$, the transpose of $A$, denoted by $A^T$, is an $n \times m$ matrix with
  $$ (A^T)_{ij} = A_{ji}. $$

- **Example**
  $$ A = \begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 5 & 9 \end{bmatrix} \quad A^T = \begin{bmatrix} 2 & 1 & 5 \\ 1 & -1 & 9 \end{bmatrix} $$

- $(A^T)^T = A$
- $A = A^T$, then $A$ is **symmetric**.

- **Vector transpose**
  $$ x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad x^T = [x_1 \ x_2 \ \cdots \ x_n] $$
Vector Norm

- **Norm**: $\| \cdot \| : \mathcal{V} \rightarrow \mathbb{R}_+$, a function that takes a vector as input and returns a positive real number.

- **$L_2$ norm**: for a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$,

  $$\| \mathbf{x} \|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

- **Decomposition**:

  $$\mathbf{x} = \| \mathbf{x} \| \cdot \frac{\mathbf{x}}{\| \mathbf{x} \|}$$

  - magnitude
  - direction

- $\| a \mathbf{x} \| = |a| \| \mathbf{x} \|$.

- $\| \mathbf{x} + \mathbf{y} \| \leq \| \mathbf{x} \| + \| \mathbf{y} \|$ (triangle inequality).

- $\| \mathbf{x} \| = 0 \implies \mathbf{x} = \mathbf{0}$.
Inner product

Given two (column) vectors \( x, y \in \mathbb{R}^n \), the inner product (or dot product) between two vectors is the matrix-matrix product:

\[
\langle x, y \rangle = x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n
\]

\[
= \sum_{i=1}^{n} x_i y_i
\]

- inner product as a function
  \[
  \langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}
  \]

- Properties
  - \( \langle x, y \rangle = \langle y, x \rangle \) (commutative)
  - \( \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle \) (linearity in first argument)
  - for \( \forall x \), \( \langle x, x \rangle = x^T x \geq 0 \), \( \langle x, x \rangle = 0 \implies x = 0 \) (why?)
Matrix Addition

Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times n}$.

$$A + B = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

- Commutative: $A + B = B + A$
- Associative: $(A + B) + C = A + (B + C)$
- $(A + B)^\top = A^\top + B^\top$
Matrix Multiplication

• If $A \in \mathbb{R}^{m \times p}$ and $B \in \mathbb{R}^{p \times n}$, then $AB \in \mathbb{R}^{m \times n}$

\[
\begin{bmatrix}
2 & 1 & 3 \\
-1 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & -1 \\
0 & 2 \\
-2 & 3 \\
\end{bmatrix} =
\begin{bmatrix}
-4 & 9 \\
-3 & 4 \\
\end{bmatrix}
\]

▶ Rule

\[
C_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} = a_{i1} b_{1j} + \cdots + a_{ip} b_{pj}
\]

▶ Properties

• $IA = AI = A$
• $(AB)C = A(BC)$
• $AB \neq BA$ (not commutative)
• $A(B + C) = AB + AC$, $(A + B)C = AC + BC$
• $(AB)^\top = B^\top A^\top$
System of linear equations

- A basic problem in linear algebra: given $b \in \mathbb{R}^m$, find $x$ such that $Ax = b$,

where $A \in \mathbb{R}^{m \times n}$ with column vectors $a_1, \ldots, a_n \in \mathbb{R}^m$.

\[
Ax = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} =
\begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_m
\end{bmatrix}
\]

\[
x = x_1 \begin{bmatrix}
a_{11} \\
a_{21} \\
\vdots \\
a_{m1}
\end{bmatrix} + x_2 \begin{bmatrix}
a_{12} \\
a_{22} \\
\vdots \\
a_{m2}
\end{bmatrix} + \cdots + x_n \begin{bmatrix}
a_{1n} \\
a_{2n} \\
\vdots \\
a_{mn}
\end{bmatrix}
\]

\[
= x_1 a_1 + x_2 a_2 + \cdots + x_n a_n
\]
1 Background
- Linear Algebra
- Probability
- Calculus
- ML Overview

2 Linear Regression
- Motivation
- Implementation
- Diagnostics

3 Python Programming
- Basics
- Numpy
What is Probability?

Definitions
1. mathematical model to explain real-life phenomena
2. mathematical language for quantifying uncertainty
3. numerical measure of the likelihood of some events
4. proportion of observations in which a given outcome happens, given infinite observations
5. strength of belief that a certain event will happen
6. a function that takes an “event” as input and returns a number in [0, 1]

Probability is a mathematical tool for modeling uncertainty!
Formal Definition

**Probability Space**

A *probability space* is a triple of \((\Omega, \mathcal{F}, P)\).

- **Sample space** \(\Omega\): set of all possible outcomes
  - flipping a coin: \(\Omega = \{H, T\}\)
  - flipping a coin twice: \(\Omega = \{HH, HT, TH, TT\}\)
  - rolling a die: \(\Omega = \{1, 2, 3, 4, 5, 6\}\)

- **Event space** \(\mathcal{F}\): a set of event
  - An event is a subset of \(\Omega\).
  - A possible event \(E = \{\text{roll is even}\}\)
  - A event: at least one head \(E = \{HH, HT, TH\}\)

- **Probability measure** \(P : \mathcal{F} \to [0, 1]\)
  - given an event, \(P\) returns a number between 0 and 1
  - \(P(H) = \frac{1}{2}, P[\text{die = 1}] = \frac{1}{6}\)
Axioms of Probability (a.k.a Kolmogorov axioms)

- $\mathbb{P}(E) \geq 0$ for $\forall E \in \mathcal{F}$
- $\mathbb{P}(\Omega) = 1$
- $E_1, E_2, \ldots \in \mathcal{F}$ are pairwise disjoint, then $\mathbb{P}(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mathbb{P}(E_i)$.

Consequences

- $\mathbb{P}\left(E^c\right) = 1 - \mathbb{P}(E)$, $\mathbb{P}(\emptyset) = 0$
- If $E \subset F$, $\mathbb{P}(E) \leq \mathbb{P}(F)$.
- $\mathbb{P}(E \cup F) = \mathbb{P}(E) + \mathbb{P}(F) - \mathbb{P}(E \cap F)$
Naive example

Are people in GA generous in rating restaurants?

- Not clear what is meant by being generous
- favorable review = stars $\geq 4$
- unfavorable review = stars $< 4$

What is $P[\text{an individual gives a favorable review}]$?
Probability Estimation

- You collected 10,000 yelp reviews posted by individuals living in X
  - \( X = \{\text{Atlanta, Savannah, Athens}\} \)
  - favorable review = stars \( \geq 4 \),
  - unfavorable review = stars < 4

- Assume each review corresponds to a distinct individual.

<table>
<thead>
<tr>
<th>Review</th>
<th>Atlanta</th>
<th>Savannah</th>
<th>Athens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Favorable</td>
<td>1K</td>
<td>2K</td>
<td>2K</td>
</tr>
<tr>
<td>Unfavorable</td>
<td>2K</td>
<td>2K</td>
<td>1K</td>
</tr>
</tbody>
</table>

- You pick a random person.
- What is \( P[\text{Favorable review}] \)?
Conditional Probability

What if *additional information* is available?

- You pick a random person.
- You found that the person is from Athens.
- Should you change the probability given the extra information?

**Conditional probability**

If $\mathbb{P}(B) > 0$, then the probability of $A$ given $B$ is

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

- By definition, we have

$$\mathbb{P}(A \cap B) = \mathbb{P}(A \mid B) \mathbb{P}(B) = \mathbb{P}(B \mid A) \mathbb{P}(A)$$  \hspace{1cm} (1)
Conditional Probability

Fig. 6. $\mathbb{P}(A)$

Fig. 7. $\mathbb{P}(A \mid B)$
Independence

Consider two events $A$ and $B$.

- Does knowing $B$ occurred tell us whether $A$ happens?
- We say $A$ and $B$ are independent if $\Pr(A \mid B) = \Pr(A)$.
- We write $A \perp \perp B$ to when $A$ and $B$ are independent.
- This means $\Pr(A \cap B) = \Pr(A) \Pr(B)$.
- $B$ bears no information about $A$.

Suppose $A$ and $B$ are two events.

- if $A \perp \perp B$, are $A$ and $B$ disjoint (i.e., $A \cap B = \emptyset$)?
- if $A$ and $B$ are disjoint, are $A$ and $B$ independent?
Law of Total Probability

**Partition**

The events $E_1, E_2, \ldots, E_n$ are partitions of $\Omega$ if
- $E_i \cap E_j = \emptyset$ for $i \neq j$ (mutually exclusive) and
- $E_1 \cup \ldots \cup E_n = \Omega$ (collectively exhaustive).

Let $B_1, B_2, \ldots, B_n$ be a partition of $\Omega$. Then

$$P(A) = \sum_{i=1}^{n} P(A \mid B_i) P(B_i).$$

**Implication**

- translate the conditional probabilities $P(A \mid B_i)$ into unconditional probability $P(A)$
Let $A$ and $B$ be two events. $A$ and $B$ are conditionally independent given $C$ iff

$$P(A, B \mid C) = P(A \mid C) P(B \mid C).$$

- Equivalently, $P(A \mid B, C) = P(A \mid C)$.
- This means that $B$ doesn’t tell us anything about $A$ if we know that $C$ occurred.
Suppose you have a dataset about diabetes test results.

- a diagnostic test is known to be 95% accurate.
- If a person is a diabetic, the test will detect it with probability 0.95.
- If a person is not a diabetic, the test will report that the subject is not a diabetic with probability 0.99.
- It is known that about 1% of the population have diabetes.

Given a person chosen at random from the population, what is the probability that the person will be reported positive?
Bayes Theorem

- A way to relate $\mathbb{P}(A \mid B)$ to $\mathbb{P}(B \mid A)$?

\[
\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A, B)}{\mathbb{P}(B)} \quad \text{(by definition)}
\]
\[
= \frac{\mathbb{P}(B \mid A) \mathbb{P}(A)}{\mathbb{P}(B)} \quad \text{(using the property (1))}
\]

- Let $A_1, A_2, \ldots A_n$ are a partition of $\Omega$.

\[
\mathbb{P}(A_k \mid B) = \frac{\mathbb{P}(A_k) \mathbb{P}(B \mid A_k)}{\mathbb{P}(A_k) \mathbb{P}(B \mid A_1) + \mathbb{P}(A_2) \mathbb{P}(B \mid A_2) + \ldots + \mathbb{P}(A_n) \mathbb{P}(B \mid A_n)}
\]

- When is it useful?
  - partitioning the sample space is convenient.
  - prediction: conditional probability $\mathbb{P}(A_k \mid B)$ is enough for classification.
Random Variable

Definition

If \((\Omega, \mathcal{F}, P)\) is a probability space, a \textit{random variable} is a real-valued function \(X : \Omega \rightarrow \mathbb{R}\).

Example

- Let \(X\) be the sum of two dice.
  - \(X \in \{2, 3, 4, 5, \ldots, 12\}\)
  - \(P(X = 2) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}\)
  - \(P(X = 3) = \frac{2}{36}, P(X = 4) = \frac{3}{36}\), and so on.

- Let \(X\) be the number of heads in 10 coin flips.
  - \(X \in \{0, 1, \ldots, 10\}\)
Discrete VS Continuous

- $X$ takes values in some countable set $\Omega$.

**Example**
- The number on a die $X$
- # of heads in coin flips
- $f_X(x) = P(X = x)$
  - probability mass function
- $\sum_{\omega \in \Omega} f_X(\omega) = 1$

- $X$ can take on an infinite number of values between any two given values.

**Example**
- Precipitation in Jan.
- # of heads in coin flips
  - $P(a \leq X \leq b) = \int_a^b f_X(x) \, dx$
  - Probability density function
  - $P[X = x] = 0$
  - $\int_{-\infty}^{\infty} f_X(x) \, dx = 1$
Why using Random Variables?

Who is going to win the super bowl this year?

- $X \in \{\text{Patriots, Cowboys, Falcons, Steelers, \ldots}\}$
  - $P[X = \text{Patriots}]$
  - $P[X = \text{Cowboys}]$
  - \ldots
Common mistakes

Let $X$ be a random variable and $f_X(x)$ be its probability density function.

- $f_X(x)$ is not a probability, i.e.,
  \[ f_X(x) \neq \mathbb{P}(X = x) \]

- $\int_{-\infty}^{\infty} f_X(x) \, dx = 1$ but $f_X(x)$ can be greater than 1.

Example: Uniform distribution

- $X \sim U(0, 0.5)$
The cumulative distribution function (c.d.f.) of a random variable $X$ is $F_X(x) := \mathbb{P}(X \leq x)$.

$$f_X(x) = \begin{cases} 
\sum_{y \leq x : y \in \Omega} f_X(y) & \text{if } X \text{ is discrete} \\
\int_{-\infty}^{x} f_X(y) \, dy & \text{if } X \text{ is continuous}
\end{cases}$$

- $\frac{\partial F_X(x)}{\partial x} = f_X(x)$ (fundamental theorem of calculus)
Expectation

\[
\mathbb{E}[X] = \sum_{x \in \Omega} x f_X(x) \quad \text{(discrete)}
\]

\[
\mathbb{E}[X] = \int_{-\infty}^{\infty} x f_X(x) \, dx \quad \text{(continuous)}
\]

- linearity of expectation

\[
\mathbb{E}[aX + bY] = a \mathbb{E}[X] + b \mathbb{E}[Y]
\]

- proof?
Variance

\[
\text{Var}(X) = \mathbb{E}[(X - \mu)^2] \\
= \mathbb{E}[X^2 - 2\mu X + \mu^2] \\
= \mathbb{E}[X^2] - 2\mu \mathbb{E}[X] + \mu^2 \\
= \mathbb{E}[X]^2 - \mu^2 = \mathbb{E}[X^2] - (\mathbb{E}[X])^2,
\]

where \( \mu = \mathbb{E}[X] \).

- Variance measures dispersion around the mean.
- Variance is not linear.

\[
\text{Var}(aX + b) = a^2 \text{Var}(X)
\]
Independence of RVs

- Recall that events $A$ and $B$ are independent if $\mathbb{P}(A, B) = \mathbb{P}(A) \mathbb{P}(B)$.
- Two RVs $X$ and $Y$ are independent if

$$\mathbb{P}(X \leq x, Y \leq y) = \mathbb{P}(X \leq x) \mathbb{P}(Y \leq y).$$

- Equivalently, $F_{XY}(x, y) = F_X(x)F_Y(y)$

- If $X$ and $Y$ are independent,
  - their joint distribution factorizes into the product of marginals:

$$f_{XY}(x, y) = f_X(x)f_Y(y)$$

- $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$
Terminology

**Statistical Learning**

- Data is a set of **observations** (or examples).
  - $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^p$ and $y \in \mathcal{Y}$
  - an observation is described by a set of **attributes** or **features**.
  - $x_i$ is called feature vector.

- Feature vector: the set of attributes
  - $x_i = (x_i(1), x_i(2), \ldots, x_i(p))$

- **Labels** $\mathcal{Y}$: Categories assigned to observations
  - $\mathcal{Y}$ is discrete in classification
  - $\mathcal{Y}$ is continuous in regression

- Data can be viewed as a **matrix**.
  - each row: an observation
  - each column: a feature
1 Background
- Linear Algebra
- Probability
- Calculus
- ML Overview

2 Linear Regression
- Motivation
- Implementation
- Diagnostics

3 Python Programming
- Basics
- Numpy
A function \( f : \mathcal{X} \rightarrow \mathcal{Y} \) is a quantity that relates two quantities to each other.

- \( x \in \mathcal{X} \): \textit{domain} of \( f \)
- \( f(x) \in \mathcal{Y} \): \textit{image/codomain} of \( f \)
- Notation

\[
\begin{align*}
  f : \mathbb{R}^n & \rightarrow \mathbb{R} \\
  f : x & \mapsto f(x)
\end{align*}
\]
Consider a function \( f : \mathbb{R} \to \mathbb{R} \). The first order derivative of \( f \) at \( x \) is defined as:

\[
 f'(x) := \frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.
\]

- The derivative \( f'(x) \) tells us how much \( f \) changes when we introduce a small change in \( x \).
- Differentiation rules
  - Product rule: \((f(x) \cdot g(x))' = f'(x)g(x) + f(x)g'(x)\)
  - Sum rule: \((f(x) + g(x))' = f'(x) + g'(x)\)
  - Chain rule: \((g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)\)
Derivative: an example

Recall

\[ f'(x) = \frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}. \]

• Let \( f(x) = x^2 \). Compute \( f'(x) \).

\[
\frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} = \lim_{h \to 0} \frac{x^2 + 2hx + h^2 - x^2}{h} = \lim_{h \to 0} \frac{2hx + h^2}{h} = \lim_{h \to 0} 2x + h.
\]
Example: Chain Rule

**Example** Let $h(x) = (2x - 1)^2$.

- This means $x \mapsto (2x - 1) \mapsto (2x - 1)^2$.
- If we set $f(x) = 2x - 1$, $g(x) = x^2$, then $h(x) = g(f(x)) = (x - 1)^2$.
- By the chain rule, $h'(x) = 2(x - 1) \cdot 2 = 4(x - 1)$. 
What if $f$ is a function of multiple variables $x_1, x_2, \ldots$?

- Consider $f : \mathbb{R}^P \to \mathbb{R}$.
- For example, $f(x) = \|x\|^2 = x^T x$, where $x = (x_1, x_2)$.

For a function $f : \mathbb{R}^P \to \mathbb{R}$ of $p$ variables $x_1, \ldots, x_p$, the partial derivatives of $f$ are defined as

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_p)}{h}$$

$$\frac{\partial f}{\partial x_2} = \lim_{h \to 0} \frac{f(x_1, x_2 + h, \ldots, x_p)}{h}$$

$$\vdots$$

$$\frac{\partial f}{\partial x_p} = \lim_{h \to 0} \frac{f(x_1, x_2, \ldots, x_p + h)}{h}$$
The gradient of $f$ at $x$ is a collection of partial derivatives

$$\nabla_x f = \frac{df}{dx} = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_p} \right]^T \in \mathbb{R}^{p \times 1}.$$

- Notice that $\nabla f$ is a column vector (in denominator layout).
- $y \in \mathbb{R}$, $x \in \mathbb{R}^p$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{p \times 1}$$

- $y \in \mathbb{R}^p$, $x \in \mathbb{R}$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_p}{\partial x} \end{bmatrix} \in \mathbb{R}^{p \times 1}$$
Example: Gradient

Let \( f(x) = \|x\|^2 = \langle x, x \rangle = x_1^2 + x_2^2 \).

- What is the gradient of \( f \) at \( x = (x_1, x_2) \)?
  
  ▶ Notice that \( f(x) \) is a **scalar**.

\[
\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} \in \mathbb{R}^{2 \times 1}
\]

- Chain rule: assume \( x_1(t) \) and \( x_2(t) \) are functions of \( t \).

\[
\frac{\partial f}{\partial t} = \frac{\partial x}{\partial t} \cdot \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \frac{\partial x_1}{\partial t} \frac{\partial f}{\partial x_1} + \frac{\partial x_2}{\partial t} \frac{\partial f}{\partial x_2}
\]

right to left
Example: Gradient

Let \( f(x) = Ax, \ A \in \mathbb{R}^{m \times n}, \ x \in \mathbb{R}^n, \ f(x) \in \mathbb{R}^m. \)

1. Determine the dimension of \( \frac{\partial f}{\partial x} \in \mathbb{R}^{n \times m} \)

2. Determine the partial derivative of \( f_i \) w.r.t. \( x_j \), i.e., \( \frac{\partial f_i}{\partial x_j} \)

\[
f_i(x) = \sum_{j=1}^{n} A_{ij} x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij}
\]

3. Use the definition

\[
\frac{\partial f}{\partial x} = \begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_1}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n}
\end{pmatrix} = \begin{pmatrix}
A_{11} & \cdots & A_{1n} \\
\vdots & \ddots & \vdots \\
A_{m1} & \cdots & A_{mn}
\end{pmatrix} = A^T \in \mathbb{R}^{n \times m}
\]
Example: Gradient of an $L_2$ loss

Consider a regression task in which you solve

$$
\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^{p+1}} L(\mathbf{w}) := \frac{1}{2} \| \mathbf{y} - \mathbf{X} \mathbf{w} \|^2.
$$

- Dataset $D = (\mathbf{X}, \mathbf{y})$, $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$, $\mathbf{y} \in \mathbb{R}^n$
- Suppose you want to use GD to train your model.

$$
\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}^{(t)}; D)
$$

- Before we compute $\nabla L(\mathbf{w}^{(t)})$, we can determine the dimensionality of the gradient:

$$
\frac{\partial L}{\partial \mathbf{w}} \in \mathbb{R}^{(p+1) \times 1}.
$$
Example: Gradient of an $L_2$ loss

For one example

- $e = y - \mathbf{w}^\top \mathbf{x} \in \mathbb{R}$
  
  $\quad = y - (w_0 + w_1 x_1 + \cdots w_p x_p)$

- Chain rule
  
  \[
  \frac{\partial L}{\partial \mathbf{w}} = \frac{\partial e}{\partial \mathbf{w}} \cdot \frac{\partial L}{\partial e}
  \]

  \[
  \begin{bmatrix}
  \frac{\partial e}{\partial w_0} \\
  \frac{\partial e}{\partial w_1} \\
  \vdots \\
  \frac{\partial e}{\partial w_p}
  \end{bmatrix}
  \cdot 
  \begin{bmatrix}
  1 \\
  x_1 \\
  \vdots \\
  x_p
  \end{bmatrix}
  \cdot
  \begin{bmatrix}
  y - \mathbf{w}^\top \mathbf{x}
  \end{bmatrix}
  \]

  \[
  L(\mathbf{w}) = \frac{1}{2} e^2
  \]
Example: Gradient of an $L_2$ loss

- $e = y - Xw \in \mathbb{R}^n$
  
  $e_i = y_i - x_i^\top w$
  
  $= y_i - (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip}) \in \mathbb{R}$

- Chain rule
  
  $\frac{\partial L}{\partial w} = \frac{\partial e}{\partial w} \cdot \frac{\partial L}{\partial e}$

  $\in \mathbb{R}^{(p+1) \times n}$

  $\in \mathbb{R}^{n \times 1}$

  $= \begin{bmatrix}
  \frac{\partial e_1}{\partial w_0} & \cdots & \frac{\partial e_n}{\partial w_0} \\
  \vdots & \ddots & \vdots \\
  \frac{\partial e_1}{\partial w_p} & \cdots & \frac{\partial e_n}{\partial w_p}
  \end{bmatrix} \cdot e$

  $= - \begin{bmatrix}
  1 & \cdots & 1 \\
  x_{11} & \cdots & x_{n1} \\
  \vdots & \ddots & \vdots \\
  x_{1p} & \cdots & x_{np}
  \end{bmatrix} \cdot (y - Xw) = -X^\top (y - Xw)$

- $L(w) = \frac{1}{2} \|e\|^2$
1 Background
   - Linear Algebra
   - Probability
   - Calculus
   - ML Overview

2 Linear Regression
   - Motivation
   - Implementation
   - Diagnostics

3 Python Programming
   - Basics
   - Numpy
• Consider learning a function $f$.
  
  - **input**: image $x$ of either a cat or dog
  - **output**: $\{\text{CAT}(0), \text{DOG}(1)\}$

  $$f(x) = \begin{cases} 
  0 & \text{if } x \text{ is a cat}, \\
  1 & \text{if } x \text{ is a dog}. 
  \end{cases}$$

• The function $f$ is called a **classifier**.
A Classifier

What should the function $f$ model?

- We could *hardcode* our knowledge into $f$?
  - For example, if $x$ has pointy ears, then do something...
  - If $x$ has whisker, do something...

What is the anticipated *problem* of this approach?
ML learns from data (examples)!

- To build a classifier, we need a set of examples.

\[ D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \]

- Mathematically, we can model \( D \) using a pair \((X, Y)\) of random variables.
  
  Why do we use a random variable?

<table>
<thead>
<tr>
<th>Computer Science</th>
<th>Statistics</th>
<th>Other names</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>Example</td>
<td>Observation</td>
</tr>
<tr>
<td>( Y )</td>
<td>Label</td>
<td>Response</td>
</tr>
<tr>
<td>( D )</td>
<td>Dataset</td>
<td>Sample</td>
</tr>
<tr>
<td>( Y = ? )</td>
<td>Learning</td>
<td>Estimation</td>
</tr>
</tbody>
</table>
Assumptions on Data

• We assume \((X, Y)^{iid} \sim P\).
• IID means
  1. Independently and
  2. IDentically distributed.
• Why do we make this assumption?
What to learn?

- A **probabilistic** approach
  - \[ f(x) = \mathbb{E}[Y|X = x] = \mathbb{P}[Y = 1 | X = x] \]
  - Recall that this is a **conditional** probability.
ML Pipeline

\( f = \phi_2 \left( W^{(2)} \phi_1 \left( W^{(1)} X \right) \right) \)

- \( f \) takes \( x \) as input and returns a prediction \( y' \).
Features
ML Models

• Goal: learn a function $f$ that returns $P[Y|X = x]$
• Still not sure how to construct $f$.
• What should be the functional form of $f$?
  ▶ linear/non-linear? e.g., $f(x) = ax + b$ for some $a, b \in \mathbb{R}$.
  ▶ polynomial/exponential $f(x) = ax^2 + bx + c$ for $a, b, c \in \mathbb{R}$
ML Models

We make an *assumption* about our data!

- We could restrict $f$ to functions of form:
  \[ \mathcal{H} = \{ f : f(x) = ax + b, a \in \mathbb{R}, b \in \mathbb{R} \} . \]

- $\theta = (a, b)$

- $\theta$ uniquely indexes functions in $\mathcal{H}$.

- Finding the best function $f$ reduces to finding the best $\theta$.

- $\mathcal{H}$: hypothesis class (or model class)

- $\theta$: *parameter* of your model
Example

Suppose you’re given a set of numbers $D = \{x_1, x_2, \ldots, x_n\}$.

- Your goal is to learn the probability $\mathbb{P}[X > 7]$.
- How can we estimate $\mathbb{P}[X > 7]$?
- The histogram looks like a bell curve, so maybe we can assume my data comes from a Gaussian distribution.
Example: Model class

That is, we are restricting ourselves to a function class

\[ \mathcal{H} = \{ \mathcal{N}(\mu, 2) : \mu \in \mathbb{R} \} . \]

- What is my model?
- What is my model parameter?
- Now our learning goal is to choose the best \( \theta \)

\[ \arg\min_{\theta} D(\mathcal{N}(\theta, 2), \text{observed histogram}) \]
ML Pipeline

Given an image $x$, my model $f$ returns $\hat{y} = f(x)$, prediction.

In the (training) dataset, we have the ground truth $(x_i, y_i)$. 

$$f = \phi_2 \left( W^{(2)} \phi_1 \left( W^{(1)} X \right) \right)$$
Loss function

We should let our model $f$ know whether its prediction is correct or not.

• We define

$$\ell(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y}, \\ 1 & \text{if } y \neq \hat{y}. \end{cases}$$

• Formally, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
  
  ▶ $\ell(\cdot, \cdot)$ tells us how much $\hat{y}$ differs from $y$.
  
  ▶ $\ell$ is called a loss function.
  
  ▶ a.k.a. penalty function or score function
Training

Now we formally state the training process.

- \((X, Y) \sim \mathcal{P}\)
- \(D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}\), i.i.d. draws from \(\mathcal{P}\)
- \(\mathcal{H} = \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}\)
- \(\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}\)

The training in ML is to find the optimal parameter \(\theta^*\) such that

\[
\theta^* = \arg\min_{\theta \in \Theta} L(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{y} = f_\theta(x_i)).
\]

- Suppose we use 0/1 loss. What does \(L(\theta)\) represent?
Linear Regression
1 Background
  - Linear Algebra
  - Probability
  - Calculus
  - ML Overview

2 Linear Regression
  - Motivation
  - Implementation
  - Diagnostics

3 Python Programming
  - Basics
  - Numpy
- Relationship between children’s heights and the heights of their parents?

  ▶ collected data from 928 children
  ▶ prediction on a child’s height?

<table>
<thead>
<tr>
<th>Height of the midparent in inches</th>
<th>Height of the adult child</th>
<th>Total no. of adult children</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;61.7</td>
<td>62.2</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>&gt;73.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>71.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69.5</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>68.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>67.5</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>66.5</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>65.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>64.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&lt;64.0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 8. Image of Francis Galton
Galton’s Experiment

- Given a fixed $X$, $Y$ is concentrated around its mean.
- $X$ and $Y$ seems to have a linear relationship.
Galton Regression

![Galton Regression Diagram]

Jaewoo Lee
Problem Setup

We observe a sample $D = \{(X_1, Y_1), \ldots, (X_N, Y_N)\} \sim \mathcal{P}$, where

- $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, supervised
- $X_i \in \mathcal{X} \subseteq \mathbb{R}^p$, feature or covariate (or explanatory variable)
- $Y_i \in \mathbb{R}$, response variable

- Build a function $f$ that predicts $Y$ given $X$
Simple Regression

In *simple regression*, we model

\[ \hat{Y} = f(X) = \mathbb{E}[Y \mid X] = \beta_0 + \beta_1 X, \]

where

- \( \beta_0 = y\text{-intercept} \) (value of \( y \) when \( x = 0 \))
  - \( \mathbb{E}[Y \mid X = 0] = \beta_0 + \beta_1 \cdot 0 = \beta_0 \)

- \( \beta_1 = \text{slope} \)
  - change in expected response per 1 unit increase in \( X_i \)

- A regression is a description of \( Y \) as a function of \( X \).
  - \( X \): independent variable, explanatory variable, predictor, feature
  - \( Y \): dependent variable, response variable, target
Simple Regression: fitting a line to data

- Model: linear lines that can be expressed as
  \[ \mathcal{M} = \{ f \mid Y = f(X) = \beta_0 + \beta_1 X \}. \]

- \( \theta = (\beta_0, \beta_1) \): model parameter
- \( \theta \) uniquely identifies a line.
- Fitting a line reduces to finding the best \( \theta \).
Method of Least Squares

How to find “good” estimators of $\beta_0$ and $\beta_1$?

- Error of my model
  - Given a point $(X_i, Y_i)$
  - Model parameter $(\beta_0, \beta_1)$

\[ Y_i - (\beta_0 + \beta_1 X_i) \]

\[ y = \beta_0 + \beta_1 x \]
Method of Least Squares

- Since we have $n$ observations in our data,

$$\ell(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

▶ $L_2$ loss or MSE (Mean Squared Error)

<table>
<thead>
<tr>
<th>Long (X)</th>
<th>JanTemp (Y)</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.5</td>
<td>44</td>
<td>$(44 - (\beta_0 + \beta_1 \times 88.5))^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86.8</td>
<td>38</td>
<td>$(38 - (\beta_0 + \beta_1 \times 86.8))^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>112.5</td>
<td>35</td>
<td>$(35 - (\beta_0 + \beta_1 \times 112.5))^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35.4</td>
<td>92.8</td>
<td>$(92.8 - (\beta_0 + \beta_1 \times 35.4))^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.7</td>
<td>105.3</td>
<td>$(105.3 - (\beta_0 + \beta_1 \times 40.7))^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Method of Least Squares

- Minimizing $\hat{\text{MSE}}(b_0, b_1)$ yields

\[
\beta_1 = \frac{\sum X_i Y_i - \left( \frac{\sum X_i}{n} \right) \left( \sum Y_i \right)}{\sum X_i^2 - \left( \frac{\sum X_i}{n} \right)^2} = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2}
\]

\[
\beta_0 = \frac{1}{n} \left( \sum Y_i - \beta_1 \sum X_i \right) = \bar{Y} - \beta_1 \bar{X}
\]
Multiple Regression

Optimal way of doing regression is to find $f(X) = E[Y | X]$.

- Impossible without knowing $(X, Y) \sim P$
- Assume $Y$ is a linear function of features $X = (X_1, \ldots, X_p)$.

\[
f(x) = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = \sum_{i=1}^{p} \beta_i x_i = \beta^T x
\]

where

- $\beta_1, \ldots, \beta_p$ are parameters (or coefficients).
Bias Term

\[ y = \beta x \]

\[ y = \beta_0 + \beta^\top x \]

Notational trick

\[ f(x) = \beta_0 + \beta^\top x \]

- \( \beta_0 \) is called bias (or intercept),
- Modified data \( x' = (1, x) \)
- \( y(x') = \beta^\top x', \) where \( \beta \in \mathbb{R}^{p+1} \).
Multiple Linear Regression

- Regression function $f : \mathcal{X} \to \mathcal{Y}$ is a linear combination of inputs.

$$\hat{y} = f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$
Linear Regression

\[
y = X \beta + \epsilon
\]

- $y$: response variable
- $X$: data matrix, each row corresponds to an example.
- $\beta$: coefficients (or weights)
- $\epsilon$: noise in the measurement
**Residual Sum of Squares**

\[
\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2
\]

\[
= \sum_{i=1}^{N} \left( y_i - \left( \sum_{j=1}^{p} \beta_j x_{ij} + \beta_0 \right) \right)^2
\]

\[
= \sum_{i=1}^{N} (y_i - \beta^T x_i)^2
\]

\[
= \|y - X\beta\|^2
\]

Fig. 10. Image from Hastie et al. ESL
Optimization

Linear Regression

\[
\text{minimize}_{\beta \in \mathbb{R}^{p+1}} \| y - X\beta \|^2
\]

where

\[
X = \begin{pmatrix}
1 & x_{11} & x_{12} & \cdots & x_{1p} \\
1 & x_{21} & x_{22} & \cdots & x_{2p} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_{N1} & x_{N2} & \cdots & x_{Np}
\end{pmatrix}
\]

and

\[
y = \begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix}
\]

Residual

\[
\| y - X\beta \|^2 = \left\| \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \right\|^2
\]
How to solve?

Assumptions

• \( \mathbf{X} \) has a full-rank (nonsingular)
• Gaussian distributed noise \( \epsilon \sim \mathcal{N}(0, \sigma^2) \)
• constant variance \( \sigma^2 \)

The negative log likelihood of \( \beta \) is

\[
NLL(\beta) = \frac{1}{2} \beta^\top \mathbf{X}^\top \mathbf{X} \beta - \beta^\top (\mathbf{X}^\top \mathbf{y}) .
\]

Taking derivatives and setting equal to zero yields

\[
(\mathbf{X}^\top \mathbf{X}) \beta - \mathbf{X}^\top \mathbf{y} = 0
\]

\[
(\mathbf{X}^\top \mathbf{X}) \beta = \mathbf{X}^\top \mathbf{y}
\]

\[
\beta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\]
1 Background
   - Linear Algebra
   - Probability
   - Calculus
   - ML Overview

2 Linear Regression
   - Motivation
   - Implementation
   - Diagnostics

3 Python Programming
   - Basics
   - Numpy
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

model = smf.ols(formula='mpg ~ drat + wt', data=mtcars).fit()
model.summary()
```
Statsmodels

```python
# plot the data
plt.plot(mtcars['drat'], mtcars['mpg'], 'ko', mfc='none')

# values for x-axis
b0, b1 = model.params
x = np.linspace(2, 5, 100)
y = b0 + b1 * x
plt.plot(x, y, '-', color='C1')
plt.xlabel('drat')
plt.ylabel('mpg')
```
Scikit-Learn

Refer to the accompanied jupyter notebook
1 Background
- Linear Algebra
- Probability
- Calculus
- ML Overview

2 Linear Regression
- Motivation
- Implementation
- Diagnostics

3 Python Programming
- Basics
- Numpy
Variance Score

The proportion of the total explained variation in y

\[ R^2 = \frac{\text{Var}(\hat{\mathbf{X}}\hat{\mathbf{\beta}})}{\text{Var}(y)} = \frac{\sum_{i=1}^{N} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \]

• It tells you how much variability in the dataset is explained by your model.
• Caveat: \( R^2 \) can only increase as we add more variables to our model.
Residual Plot

(a) Lat

(b) Long
Fig. 12. Residual Plots
Overfitting

Fig. 13. An example dataset

- blue: $\beta_0 + \beta_1 x$
- green: $\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{10} x^{10}$

Fig. 14. An example of overfitting

Jaewoo Lee
Python Programming
1 Background
- Linear Algebra
- Probability
- Calculus
- ML Overview

2 Linear Regression
- Motivation
- Implementation
- Diagnostics

3 Python Programming
- Basics
- Numpy
Getting Started

How to run a Python program?

1. script mode
   - type your code into a file and name it as *filename.py*
   - run `python filename.py`

2. interactive mode
   - command line shell
   - enter each statement and receive an immediate feedback

3. Jupyter notebook
   - code cell
   - markdown cell
   - raw cell
Windows users (Optional)

- Recommend to install **Git-bash**!
  - This allows you to use bash on windows systems.
Windows users (Optional)

- If your system doesn’t recognize the command you typed,
  - check your **PATH environment variable**
  - check your installation
Linux users (Optional)

- Adding a directory to $PATH
- You can add this line to your .bashrc file to execute it whenever you open a new shell.
Python Development Environment

1. Integrated Development Environment (IDE)
   - Spyder
   - PyCharm
   - NetBeans

2. Text Editor + plug-ins
   - Atom
   - Sublime
   - Notepad++
   - Emacs
Hello World!

- “Hello World!” in **Java**

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- “Hello World!” in **C**

```c
#include <stdio.h>

int main(int argc, char **argv) {
    printf("Hello World!\n");
}
```

- “Hello World!” in **Python**

```python
print("Hello World!")
```
Java VS Python

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class MyJavaProgram {

    public static void main(String[] args) {
        BufferedReader objReader = null;
        try {
            String strCurrentLine;
            objReader = new BufferedReader(new FileReader("C:\mydata.txt"));

            while ((strCurrentLine = objReader.readLine()) != null) {
                System.out.println(strCurrentLine);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (objReader != null)
                    objReader.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```
Java VS Python

```python
with open('C:\mydata.txt', 'r') as f:
    print(f.read())
```
Formatting Python code

- In Python, the scope of code is determined by indentation!
  - Don’t mix spaces and tabs

- In other languages,
  - some form of begin-end symbol (e.g., { and })

```python
import numpy as np

def odd_sum(n):
    return np.sum(np.arange(1, n+1, 2))

sum = 0

for i in range(1, 11):
    if i % 2 == 0:
        continue
    sum += i

print("sum = ", sum)

mysum = odd_sum(10)
print("mysum = ", mysum)
```
```python
def add_two(a, b):
    
    This function adds two given numbers.

    Parameters
    ------------
    a: integer, operand1
    b: integer, operand2

    Returns:
    ------------
    result: integer, results of a+b

    # this is a single line comment
    result = a + b

    return result
```

- *Document* your input and output parameters in a comment!
Variable

- Variables are *dynamically typed!*
  - can bind a name to objects of different types

- Type checking
  - Static: types are checked before run-time
  - Dynamics: types are checked on the fly, during execution

```python
1  x = 2
2  type(x)  # <class 'int'>
3  x = 2.0  
4  type(x)  # <class 'float'>
5  x = "my string"
6  x = 20
7
```
String

\[ s = "\text{H e l l o}\"^{\text{(from index 0 to 4)}} \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Slicing

- list[start : stop : step]
  - start: start index for slicing (inclusive)
  - stop: end index, slicing stops at stop - 1 (exclusive)
  - step: increment between each index for slicing

- length: \text{len(s)}

- Examples
  - s[1:]
  - s[1:-1]
  - s[1:4:2]
  - s[-1::-2]
String manipulation

```python
my_string = "Hello World!"

'll' in my_string  # True  
my_string.find('ll')  # 2
my_string.replace('ello', 'i')
my_string.replace('!', '')
my_string.isalpha()  # False
my_string.lower()
my_string.upper()
my_string.strip()  # "Hello World!"
my_string.split()  # ['Hello', 'World!']
','.join(my_string)
```
my_list = []
my_list.append('a')  # append an element to the end of a list
my_list.append('b')  # my_list = ['a', 'b']
my_list.append(3)
my_list.append([1, 2, 3, 4, 5])  # a list can contain another list as an element
my_list.append([])
my_list.pop(0)  # pops the element at index 0

# membership check
'a' in my_list  # True
1 in my_list  # False

- What is my_list[3][2]?
- What is len(my_list)?
Functions

- When some parts of your code repeat,
  - copy and paste?
  - defining them as a function allow you to avoid repetition.

```python
def avg_or_sum(nums, is_avg=True):
    v_sum = sum(nums)  # built-in function sum()
    cnt = len(nums)    # len() function returns the number of element in a list

    if is_avg:
        v_sum /= cnt

    return v_sum, cnt

s, c = avg_or_sum(range(1, 11))
print("s={}",format(s, c))
```

- return is optional
  - can return multiple values!

- arguments
  - positional
  - keyword
Scope of Variables

```python
# 1
x = 1

# 2
def myfunc(y):
    x = y
    print("x={}, y={}".format(x, y))

# 3
myfunc(3)

# 4
print("x={}".format(x))

# 5
x = 3

# 6
if condition:
    x = 1
else:
    x = 2

# 7
print("x={}", x)
```

Symbol tables

- Local
- Global
- Built-in

- If/For statement does not introduce a new scope.
Functions

- **Default** values for parameters
  - when there exists a common setting
  - possible to call a function with fewer argument than it is defined

```python
def myfunc(a, b, c=3, d='...'):
    print("a={}, b={}, c={}, d={}".format(a, b, c, d))

myfunc(1, 2)
myfunc(a=1, b=2)
# positional argument follows keyword argument
myfunc(a=1, 2, c=3)
```
def myfunc(*vals, norm=True):  # argument after *vals needs to be keyword-only arguments
    mysum = 0

    for v in vals:
        mysum += v

    if norm:
        mysum /= len(vals)

    return mysum

print(myfunc(1, 2, 3))
print(myfunc(1, 2, 3, 4, 5, norm=False))
mylist = [1, 2, 3, 4, 5]
print(myfunc(*mylist), norm=False)
Built-in Functions

See here for a complete list of built-in functions.

- **range(start, stop[, step])**
  1. `range(i, 11, 2)`  # specified all three parameters
  2. `range(i, 11)`    # specified start and stop
  3. `range(10)`       # specified only stop

- **len(iterable)**
  1. `len(range(i, 11, 2))`
  2. `len("hello world")`

- **min(iterable)**
- **max(iterable)**
  1. `max(np.arange(i, 11, 2))`
  2. `min(-1, 3, 45, 0, -7, 10)`
Built-in Functions

- **open(file, mode=’r’)**
  - file.read(): read the whole file
  - file.readline(): read a single line
  - file.write(content): write content to the file

- **zip(*iterables)**
for loops

- Loops execute a block of code multiple times

```
for value in iterable:
    # execute this
    # execute that
```

- loop over any *iterable* Python object (e.g., lists, arrays, and strings)

```
a_list = [1, 2, 3]
a_dict = {'key1': 'val1', 'key2': 'val2'}

for x in a_list:
    print(x)

for k in a_dict:
    print('key=' + k + ' value=' + a_dict[k])

for c in "CSCI 3360 is cool!":
    print(c, end='')
```
range

- `range(stop)`
- `range(start, stop, [step])`

```python
1  range(10)  # generates integers from 0 to 9
2  range(1, 11)  # generates integers from 1 to 10
3  range(1, 11, 2)  # generates 1, 3, 5, 7, 9
```
Control Statement

- if statements
  - execute a block of code only when a certain condition is true

```python
# cnt = 0
# for val in [8, 2, 7, 11, 5, 7, 7]:
#     if val > 7:
#         print("x", end='')
#     else:
#         if val == 7:
#             cnt += 1
#         else:
#             print("o", end='')
# print("\tcnt={}".format(cnt))
```

```python
# cnt = 0
# for val in [8, 2, 7, 11, 5, 7, 7]:
#     if val > 7:
#         print("x", end='')
#     elif val == 7:
#         cnt += 1
#     else:
#         print("o", end='')
# print("\tcnt={}".format(cnt))
```
Importing Modules

- module: a file containing Python definitions and statements.
- Suppose you defined the following two function and stored into “mylib.py”

```python
import numpy as np

def sum_to_n(n):
    mysum = sum(range(1, n+1))
    return mysum

def prod_to_n(n):
    return np.prod(np.arange(1, n+1))
```

- Later you can reuse these functions:

```python
import mylib
mylib.sum_to_n(10)
mylib.prod_to_n(10)
```

```python
from mylib import sum_to_n, prod_to_n
sum_to_n(10)
prod_to_n(10)
```
Packages

- `__init__.py` is an empty file (only needed for python 2)
- `import mylib.calculus.func1 as func1`
- `from mylib.linalg.matrix import prod`

Two ways to make this work
- put the package under: PYTHON_DIR\Lib\site-packages
- add the path to your package to `PYTHONPATH` environment variable
Q: Write a program that draws a rectangle with the given width and height.

```python
def draw_rectangle(width, height, title=None):
    
    # Draw a rectangle with the given width and height

    Parameters
    ---------------
    width: integer, width of the rectangle to draw
    height: integer, height of the rectangle to draw
    title: title to display at the center, None by default

    pass
```
Q: Write a program that will print out all the prime numbers smaller than 1000.

- (hint): prime numbers are only divisible by one and themselves
- (hint): as soon as you have found that the given number is divisible by another number, you immediately know it’s not a prime number
1 Background
   - Linear Algebra
   - Probability
   - Calculus
   - ML Overview

2 Linear Regression
   - Motivation
   - Implementation
   - Diagnostics

3 Python Programming
   - Basics
   - Numpy
Numpy Package

What is numpy?

- short for Numerical Python
- efficient storage and data operations (especially for large arrays)
- well suited for scientific computation

```python
# conventional alias
import numpy as np

# make sure you use the latest version
cpy.__version__
```

Fig. 16. Numpy array
Basic attributes

```python
X = np.arange(9).reshape(3, 3)
print X.dtype
print X.ndim
print X.shape
print X.size
print len(X)  # is this the same with X.size?
```

- **ndim**: number of axes
- **shape**: tuple of integers representing # of dimensions along each axis
- **dtype**: data type of a tensor

**Conversion to List**

```python
x.tolist()  # same with list(x)
```
np.arange(0.1, 1.0, 0.1)
# array([ 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])

np.zeros(10)
# array([ 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

np.ones(10)
# array([ 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

np.empty(10)
# array([ 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

x = np.array([1, 2, 3, 4, 5])
>>> x.dtype
# dtype('int32')

y = np.zeros_like(x)

np.linspace(0, 1, 20)
array([ 0.  , 0.05263158, 0.10526316, 0.15789474, 0.21052632, 0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421, 0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211, 0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.   ])
Random Number Generation

```python
np.random.randn(2, 5)
array([[-1.17418172, 0.42727156, 0.05420085, 0.1515822 , 0.5032227 ],
       [-0.4873585 , 1.64119856, 0.13500736, -0.97912331, 0.2859684 ]])

np.random.normal(loc=2, scale=2, size=(2, 5))
array([[ 0.75083689, 2.48187302, 4.06580453, 1.88676289, 2.64741859],
       [ 1.4172726 , 5.84980115, 0.58042385, 3.31189573, 3.24031452]])

np.random.random((2, 5))
array([[ 0.35295465, 0.1684032 , 0.03598568, 0.06234497, 0.372518 ],
       [ 0.32309613, 0.40686758, 0.19170374, 0.89235971, 0.05863298]])
```
Array Operations

Conventional Way

```python
def square(x):
    a = range(x)
    b = []
    for v in a:
        b.append(v*v)
    return b

In [11]: %timeit square(100)
100000 loops, best of 3: 11.7 µs per loop
```
Array Operations

Numpy’s Way

def square_np(x):
    return np.square(np.arange(x))

In [12]: %timeit square_np(100)
# The slowest run took 27.16 times longer than the fastest. This could
# mean that an intermediate result is being cached.
# 1000000 loops, best of 3: 1.27 µs per loop
Array Operations

- Arithmetic operators are applied *elementwise*. 
Array Operations

Fig. 17. Broadcasting in numpy (Image from https://mathematica.stackexchange.com/questions/99171/how-to-implement-the-general-array-broadcasting-method-from-numpy)
Broadcasting

Rules

1. If the number of dimensions between two arrays doesn’t match, numpy tries to *pad* the one with smaller dimensionality with ones from the left side.

2. If the shape of two arrays does not match in any dimension, the dimension with 1 element is *stretched* to match the other array’s shape.

3. If the size differs in all dimensions and none of them is 1, numpy throws an *error*!

```python
A = np.array([[1, 2, 3], [4, 5, 6]])
x = np.array([1, 2, 3])
A + x
```
Quiz: what is the result?

```python
A = np.array([[1, 2, 3], [4, 5, 6]])
x = np.array([1, 2])
A + x  # ??
```
Vector Operations

```python
x = np.array([1, 2, 3])
y = np.array([2, 3, 4])

In [26]: np.dot(x, y)
Out[26]: 20

X = np.array([[1, 2], [3, 4]])
Y = np.array([[2, 3], [4, 5]])

In [33]: X * Y
Out[33]:
array([[  2,   6],
       [12,  20]])

# matrix multiplication
In [34]: np.dot(X, Y)
Out[34]:
array([[10, 13],
       [22, 29]])

# not commutative
In [35]: np.dot(Y, X)
Out[35]:
array([[11, 16],
       [19, 28]])
```
Descriptive Statistics

```python
x = np.arange(10)

In [38]: x.sum()
Out[38]: 45

In [39]: x.cumsum()
Out[39]: array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45])

In [40]: x.min()
Out[40]: 0

In [41]: x.max()
Out[41]: 9

y = np.arange(9).reshape((3, 3))

# array([[0, 1, 2],
#        [3, 4, 5],
#        [6, 7, 8]])

In [44]: y.max()
Out[44]: 8

# columnwise max
In [46]: np.max(y, axis=1)
Out[46]: array([2, 5, 8])
```

File Input and Output

- Save an array to a binary file in `.npy` format
  - `np.save(file, array)`
  - `np.load(filename)`

- Save/load a numpy array into/from a text file

```python
1 np.loadtxt('mydata.csv', delimiter=',
2 np.savetxt('mydata.csv', X, delimiter=',')
```
Data Representation for NN

Scalars (0D tensors)

- \texttt{ndim attribute} = 0

```python
import numpy as np

x = np.array(12)
print(x.ndim)
```
Data Representation for NN

Vectors (1D tensors)

- $n$-dimensional vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, \cdots, x_n)^\top$

$$
\mathbf{x} =
\begin{pmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{pmatrix}
$$

Initialization

```python
x = np.array([1, 2, 3], dtype=int)
x = np.zeros(3) # 1d array filled with zeros
x = np.ones(3)
x = np.empty(3) # created but not initialized
y = np.zeros_like(x)
print "is x.shape==y.shape? ", x.shape==y.shape
x = np.arange(0.1, 1, 0.1) # initialized with evenly spaced numbers
x = np.linspace(0.1, 0.9, 9)
```
Data Representation for NN

Matrices (2D tensors)

- $m \times n$ matrix $X \in \mathbb{R}^{m \times n}$

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}, \quad X^\top = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{m1} \\ x_{12} & x_{22} & \cdots & x_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{mn} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

```python
# 2d array is a list of list
X = np.array([[1, 2, 3], [4, 5, 6]])
print(X.ndim)
X.T
```

```
Out[0]: array([[1, 4],
            [2, 5],
            [3, 6]])
```
3D Tensors

```python
X = np.arange(2*3*4).reshape(2, 3, 4)
print('rank=', X.dims)
X
```

```
Out[0]: array([[[ 0,  1,  2,  3],
              [ 4,  5,  6,  7],
              [ 8,  9, 10, 11]],
             [[12, 13, 14, 15],
              [16, 17, 18, 19],
              [20, 21, 22, 23]]])
```
If you need a help

```python
1 help(np.array)
```

```
Out[0]: Help on built-in function array in module numpy.core.multiarray:

    array(...)  
    array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)  

Create an array.

Parameters
----------
object : array_like
    An array, any object exposing the array interface, an object whose
    __array__ method returns an array, or any (nested) sequence.
dtype : data-type, optional
    The desired data-type for the array. If not given, then the type will
```
What data types are available?

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>np.int64</td>
<td>signed 64-bit integers</td>
</tr>
<tr>
<td>np.float32</td>
<td>double-precision floating point</td>
</tr>
<tr>
<td>np.bool</td>
<td>boolean type {True, False}</td>
</tr>
<tr>
<td>np.object</td>
<td>python object</td>
</tr>
<tr>
<td>np.string_</td>
<td>fixed-length string</td>
</tr>
<tr>
<td>np.unicode_</td>
<td>fixed-length unicode</td>
</tr>
</tbody>
</table>

Table 1. numpy data types
Example: MNIST

```python
from keras.datasets import mnist
(train_X, train_Y), (test_X, test_Y) = mnist.load_data()
print("train data.shape={0}".format(train_X.shape))
```

Out[0]: `train data.shape=(60000, 28, 28)`

Some examples of data tensors

- Numerical data: 2D tensors of shape (samples, features)
- Time series data: 3D tensors of shape (samples, timesteps, features)
- Image data: 4D tensors of shape
  - (samples, height, width, channels)
- Video: 5D tensors of shape
  (samples, frames, height, width, channels)
Example: 4D Tensors

```python
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)  # (50000, 32, 32, 3)
print(x_train.shape[0], 'train samples')  # 50000
print(x_test.shape[0], 'test samples')  # 10000

fig = plt.figure(figsize=(8, 3))

for i in range(10):
    ax = fig.add_subplot(2, 5, i+1)
    plt.imshow(x_train[i])  # display an image

plt.show()
```

Fig. 18. A 4D Image data
Slicing

- syntax: `X[start:stop:step]`
- Indices could be negative numbers!
```python
a = np.array([[1, 2, 3], [4, 5, 6]])
b = a.ravel()
print b
b.shape

# reshape
b = b.reshape(3, 2)
print b
b.shape
```
Slices are references to memory in original array.
• Avoid using loops whenever possible.
• Perform *vectorized* operations.
Vector and Matrix

- \( \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n} \)

- Two ways to express matrices in python
  - numpy array (np.ndarray)
  - matrix object (np.matrix)

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
\begin{pmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{pmatrix}
= 
\begin{pmatrix}
a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\
a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22}
\end{pmatrix}
\]

```python
# matrix multiplication ?
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
print A*B

A = np.matrix(A)
B = np.matrix(B)

print A*B
print np.dot(A, B)
```

- Test and verify vector operations using a small toy examples!
import time

n = 100
w = np.random.randn(n)
x = np.random.randn(n)

rep = 100000  # number of iterations
# do not do this!
start_time = time.clock()

for i in range(rep):
    result = 0
    for wi, xi in zip(w, x):
        result += wi * xi
    print time.clock() - start_time

# dot product using numpy
start_time = time.clock()
for i in range(rep):
    result = np.dot(w, x)

print time.clock() - start_time
Reading from a file

```python
import numpy as np

X = []

for line in open('data.csv'):
    row = line.split(',')
    sample = map(float, row)
    X.append(sample)

X = np.array(X)
print X
```

- Read the input file line by line
- Alternative

```python
np.loadtxt('mydata.csv', delimiter=',')
np.savetxt('mydata.csv', X, delimiter=',')
```