CS4442/9542b
Artificial Intelligence II
Prof. Olga Veksler

Lecture 2

Introduction to ML
Basic Linear Algebra
Matlab

Some slides on Linear Algebra are from Patrick Nichols
• Introduction to Machine Learning
• Basic Linear Algebra
• Matlab Intro
Intro: What is Machine Learning?

- Difficult to come up with explicit program for some tasks
- Digit Recognition, a classic example
  
  ![0 → 0](image1)  ![4 → 4](image2)

- Easy to collect images of digits with their correct labels

- Machine Learning Algorithm takes collected data and produces program for recognizing digits
  - done right, program will recognize correctly new images it has never seen
Intro: What is Machine Learning?

Traditional Programming

Data → Computer → Output
Program →

Machine Learning

Data → Computer → Program
Output →
Intro: What is Machine Learning?

- General definition (Tom Mitchell):
  - Based on experience $E$, improve performance on task $T$ as measured by performance measure $P$

- Digit Recognition Example
  - $T =$ recognize character in the image
  - $P =$ percentage of correctly classified images
  - $E =$ dataset of human-labeled images of characters
Different Types of Machine Learning

• **Supervised Learning**
  • given training examples with corresponding outputs
  • learn to produce correct labels for new examples

• **Unsupervised Learning**
  • given training examples only
  • discover good data representation
    • e.g. “natural” clusters
  • not covered

• **Reinforcement Learning**
  • learn to select action that maximizes payoff
  • not covered
• Classification
  • output belongs to a finite set
  • example: age $\in \{\text{baby, child, adult, elder}\}$
  • output is also called \textit{class} or \textit{label}

• Regression
  • output is continuous
  • example: age $\in [0,130]$
Supervised Machine Learning

- We are given examples with corresponding outputs
- Fish classification example \((salmon \ or \ sea \ bass)\)

\[
\begin{align*}
\mathbf{x}^1 &= \begin{bmatrix} 3.3 \\ 5.7 \end{bmatrix} & \mathbf{x}^2 &= \begin{bmatrix} 6.3 \\ 8.7 \end{bmatrix} & \mathbf{x}^3 &= \begin{bmatrix} 2.3 \\ 1.7 \end{bmatrix} & \mathbf{x}^4 &= \begin{bmatrix} 6.4 \\ 7.0 \end{bmatrix} \\
\text{salmon} & & \text{sea bass} & & \text{salmon} & & \text{sea bass} \\
\mathbf{y}^1 &= 0 & \mathbf{y}^2 &= 1 & \mathbf{y}^3 &= 0 & \mathbf{y}^4 &= 1
\end{align*}
\]

- Each example is represented in vector form
  - data may be given in vector form from the start
  - if not, for each example \(i\), extract useful features and put them in a vector \(\mathbf{x}^i\)
    - fish classification example
      - extract two features, \(fish \ length\) and \(average \ fish \ brightness\)
      - can extract as many other features
      - can also use raw pixel values as features (for images)
    - An example is often called \(feature \ vector\)
  - Each output is represented with integer \(\mathbf{y}^i\)
Supervised Machine Learning

- We are given
  1. Training examples $x^1, x^2, ..., x^n$
  2. Target output for each sample $y^1, y^2, ..., y^n$

- **Training phase**
  - estimate function $y = f(x)$ from labeled data
    - $f$ is called classifier, learning machine, prediction function, etc.

- **Testing phase** (deployment)
  - predict label $f(x)$ for a new (unseen) sample $x$
Training/Testing Phases Illustrated

**Training**

- **Training examples**
- Feature vectors
- Training
- Learned model $f$

**Testing**

- Test Image
- Feature vector
- Learned model $f$
- Label prediction
• Estimate prediction function $y = f(x)$ from labeled data

• Choose *hypothesis space* $f(x)$ belongs to
  • hypothesis space $f(x,w)$ is parameterized by vector of *weights* $w$
  • each setting of $w$ corresponds to a different hypothesis

• find $f(x,w)$ in the hypothesis space s.t. $f(x^i,w) = y^i$ “as much as possible” for training examples
  • “as much as possible” can be defined with loss function $L(f(x,w), y)$
Training Phase Example in 1D

- 2 class classification problem
  - $y^i \in \{-1, 1\}$

- Examples are one dimensional feature vectors
  - examples in class -1: $\{-2, -1, 1\}$
  - examples in class 1: $\{2, 3, 5\}$

- Hypothesis space $f(x, w) = \text{sign}(w_0 + w_1 x)$
  - $w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$
  - one member is $f(x) = \text{sign}(-1 + 2x)$, i.e. $w_0 = -1, w_1 = 2$
• 2 class classification problem
  • $y^i \in \{-1, 1\}$

• Examples are one dimensional feature vectors
  • examples in class -1: {-2, -1, 1}
  • examples in class 1: {2, 3, 5}

• Let classifier be $f(x, w) = \text{sign}(w_0 + w_1 x)$
  • another member is $f(x) = \text{sign}(-1.5 + x)$, i.e. $w_0 = -1.5, w_1 = 1$

• Often say $f(x, w)$ is a classifier, and the process of finding good $w$ is weight tuning
For 2 class problem and 2 dimensional samples

\[ f(x, w) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2) \]

Can be generalized to examples of arbitrary dimension

Classifier that makes a decision based on linear combination of features is called a **linear classifier**
Training Phase: Linear Classifier

- **bad setting of** $w$
  - classification error 38%

- **best setting of** $w$
  - classification error 4%
Training Stage: More Complex Classifier

- for example if $f(x,w)$ is a polynomial of high degree
- 0% classification error
Test Classifier on New Data

- The goal is for classifier to perform well on new data.
- Test “wiggly” classifier on new data: 25% error.
Overfitting

- Have only limited amount of data for training
- Overfitting
  - complex model often have too many parameters to fit reliably with a limited amount of training data
  - Complex model may adapt too closely to the random noise of the training data
Overfitting: Extreme Example

• 2 class problem: face and non-face images
• Memorize (i.e. store) all the “face” images
• For a new image, see if it is one of the stored faces
  • if yes, output “face” as the classification result
  • If no, output “non-face”
  • also called “rote learning”

• **problem:** new “face” images are different from stored “face” examples
  • zero error on stored data, 50% error on test (new) data
  • decision boundary is very irregular

• Rote learning is memorization without generalization
The ability to produce correct outputs on previously unseen examples is called generalization.

Big question of learning theory: how to get good generalization with a limited number of examples.

Intuitive idea: favor simpler classifiers.
- William of Occam (1284-1347): “entities are not to be multiplied without necessity”

Simpler decision boundary may not fit ideally to the training data but tends to generalize better to new data.
Training and Testing

• How to diagnose overfitting?
• Divide all labeled samples \( x^1, x^2, ... x^n \) into \textit{training} set and \textit{test} set
• Use training set (training samples) to tune classifier weights \( w \)
• Use test set (test samples) to see how well classifier with tuned weights \( w \) work on unseen examples
• Thus there are 2 main phases in classifier design
  1. training
  2. testing
Training Phase

- Find weights \( w \) s.t. \( f(x^i, w) = y^i \) “as much as possible” for training samples \( x^i \)
  - “as much as possible” needs to be defined
    - usually some penalty whenever \( f(x^i, w) \neq y^i \)
    - penalty defined with loss function \( L(f(x^i, w), y^i) \)
  - how to search for such \( w \)?
    - usually through optimization, can be quite time consuming
  - classification error on training data is called *training error*
Testing Phase

- The goal is good performance on unseen examples
- Evaluate performance of the trained classifier $f(x,w)$ on the test samples (unseen labeled samples)
- Testing on unseen labeled examples lets us approximate how well classifier will perform in practice
- If testing results are poor, may have to go back to the training phase and redesign $f(x,w)$
- Classification error on test data is called test error
- Side note
  - when we “deploy” the final classifier $f(x,w)$ in practice, this is also called testing
• Can also underfit data, i.e. too simple decision boundary
  • chosen hypothesis space is not expressive enough
• No linear decision boundary can well separate the samples
• Training error is too high
  • test error is, of course, also high
Underfitting $\rightarrow$ Overfitting

**underfitting**
- high training error
- high test error

**“just right”**
- low training error
- low test error

**overfitting**
- low training error
- high test error
How Overfitting affects Prediction

Error

Model Complexity

underfitting  ideal range  overfitting

training data  test data
Fixing Underfitting/Overfitting

- Underfitting
  - add more features
  - use more complex $f(x,w)$

- Overfitting
  - remove features
  - collect more training data
  - use less complex $f(x,w)$
Sketch of Supervised Machine Learning

- Chose a hypothesis space $f(x, w)$
  - $w$ are tunable weights
  - $x$ is the input sample
  - tune $w$ so that $f(x, w)$ gives the correct label for training samples $x$

- Which hypothesis space $f(x, w)$ to choose?
  - has to be expressive enough to model our problem well, i.e. to avoid *underfitting*
  - yet not too complicated to avoid *overfitting*
Classification System Design Overview

- Collect and label data by hand
  - salmon  sea bass  salmon  salmon  sea bass  sea bass
- Preprocess data (i.e. segmenting fish from background)
- Extract possibly discriminating features
  - length, lightness, width, number of fins, etc.
- Classifier design
  - Choose model for classifier
  - Train classifier on training data
- Test classifier on test data

we mostly look at these steps in the course
Basic Linear Algebra

• Basic Concepts in Linear Algebra
  • vectors and matrices
  • products and norms
Why Linear Algebra?

• For each example (e.g. a fish image), we extract a set of features (e.g. length, width, color)
• This set of features is represented as a *feature vector* 
  • [length, width, color]
• All collected examples will be represented as collection of (feature) vectors

\[
\begin{bmatrix}
[l_1, w_1, c_1] & \text{example 1} \\
[l_2, w_2, c_2] & \text{example 2} \\
[l_3, w_3, c_3] & \text{example 3}
\end{bmatrix}
\]

• Often use linear classifiers since they are simple and computationally tractable
What is a Matrix?

- A matrix is a set of elements, organized into rows and columns.
Basic Matrix Operations

- addition, subtraction, multiplication by a scalar

\[
\begin{bmatrix}
a & b \\
c & d \\
\end{bmatrix} + \begin{bmatrix}
e & f \\
g & h \\
\end{bmatrix} = \begin{bmatrix}
a+e & b+f \\
c+g & d+h \\
\end{bmatrix} \quad \text{add elements}
\]

\[
\begin{bmatrix}
a & b \\
c & d \\
\end{bmatrix} - \begin{bmatrix}
e & f \\
g & h \\
\end{bmatrix} = \begin{bmatrix}
a-e & b-f \\
c-g & d-h \\
\end{bmatrix} \quad \text{subtract elements}
\]

\[
\alpha \cdot \begin{bmatrix}
a & b \\
c & d \\
\end{bmatrix} = \begin{bmatrix}
\alpha \cdot a & \alpha \cdot b \\
\alpha \cdot c & \alpha \cdot d \\
\end{bmatrix} \quad \text{multiply every entry}
\]
Matrix Transpose

- **n** by **m** matrix $A$ and its **m** by **n** transpose $A^T$

$A = \begin{bmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{bmatrix}$

$A^T = \begin{bmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{12} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{nm} & x_{2m} & \cdots & x_{nm}
\end{bmatrix}$
• Vector: N x 1 matrix

\[ v = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]

• *dot product* and *magnitude* defined on vectors only

vector addition

vector subtraction
More on Vectors

• n-dimensional row vector \( x = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix} \)

• Transpose of row vector is column vector \( x^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \)

• **Vector** product (or **inner** or **dot** product)

\[
\langle x, y \rangle = x \cdot y = x^T y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n = \sum_{i=1}^{n} x_i y_i
\]
More on Vectors

- **Euclidian norm or length** \( \|x\| = \sqrt{\langle x, x \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2} \)

- If \( \|x\| = 1 \) we say \( x \) is *normalized* or *unit* length

- angle \( q \) between vectors \( x \) and \( y \) : \( \cos \theta = \frac{x^T y}{\|x\| \|y\|} \)

- inner product captures direction relationship

\[
\begin{align*}
\cos \theta &= 0 & \cos \theta &= 1 & \cos \theta &= -1 \\
\Rightarrow x^T y &= 0 & x^T y &= \|x\| \|y\| > 0 & x^T y &= -\|x\| \|y\| < 0 \\
x \perp y & & \text{adjacent} & \text{opposite}
\end{align*}
\]
More on Vectors

- Vectors $x$ and $y$ are orthonormal if they are orthogonal and $\|x\| = \|y\| = 1$

- Euclidian distance between vectors $x$ and $y$

\[
\|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\]
Matrix Product

\[ AB = \begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \cdots & a_{1d} \\
    a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nd} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    a_{d1} & a_{d2} & a_{d3} & \cdots & a_{dm}
\end{bmatrix} \begin{bmatrix}
    b_{11} & \cdots & b_{1m} \\
    b_{21} & \cdots & b_{2m} \\
    \vdots & \ddots & \vdots \\
    b_{d1} & \cdots & b_{dm}
\end{bmatrix} = \begin{bmatrix}
    c_{ij}
\end{bmatrix}
\]

- # of columns of A = # of rows of B
- even if defined, in general \( AB \neq BA \)
MATLAB
• Starting matlab
  • xterm -fn 12X24
  • matlab
  • matlab -nodisplay
• Basic Navigation
  • quit
  • more
  • help general
• Scalars, variables, basic arithmetic
  • Clear
  • + - * / ^
  • help arith
• Relational operators
  • ==,&,|,~,xor
  • help relop
• Lists, vectors, matrices
  • A=[2 3;4 5]
  • A' 
• Matrix and vector operations
  • find(A>3), colon operator
  • * / ^ .* ./ .^ 
  • eye(n),norm(A),det(A),eig(A)
  • max,min,std
  • help matfun
• Elementary functions
  • help elfun
• Data types
  • double
  • Char
• Programming in Matlab
  • .m files
  • scripts
  • function y=square(x)
  • help lang
• Flow control
  • if i== 1 else end, if else if end
  • for i=1:0.5:2   ...  end
  • while i == 1 ... end
  • Return
  • help lang
• Graphics
  • help graphics
  • help graph3d
• File I/O
  • load, save
  • fopen, fclose, fprintf, fscanf