Machine Learning Basics

Kevin Duh

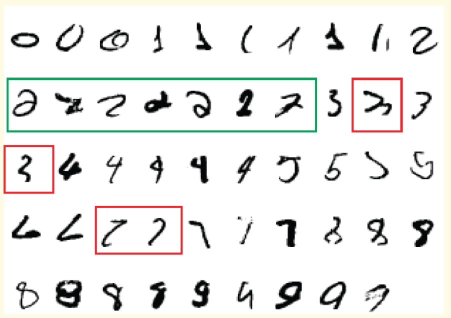
Today's Topics

1. Machine Learning basics
   - Why Machine Learning is needed?
   - Main Concepts: Generalization, Model Expressiveness, Overfitting
   - Formal Notation
   - Experiment Design

Write a Program* to Recognize the Digit 2

This is hard to do manually!

```
bool recognizeDigitAs2(int** imagePixels){...}
```

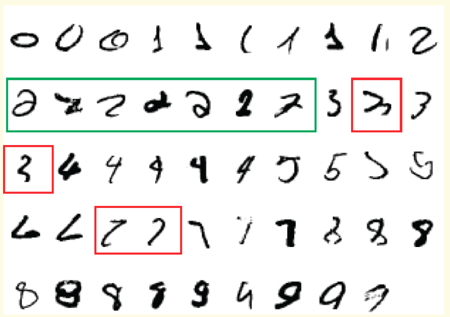


*example from Hinton's Coursera course

Write a Program* to Recognize the Digit 2

This is hard to do manually!

```
bool recognizeDigitAs2(int** imagePixels){...}
```

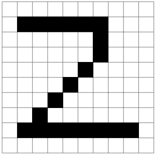
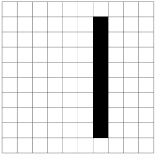


Machine Learning solution:

1. Assume you have a database (training data) of 2's and non-2's.
2. Automatically "learn" this function from data

*example from Hinton's Coursera course

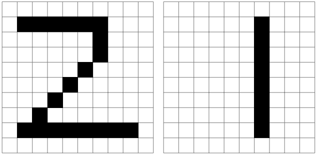
A Machine Learning Solution

Training data are represented as pixel matrices:  

Classifier is parameterized by weight matrix of same dimension.

A Machine Learning Solution

Training data are represented as pixel matrices:



Classifier is parameterized by weight matrix of same dimension.

Training procedure:

1. When observe "2", add 1 to corresponding matrix elements
2. When observe "non-2", subtract 1 to corresponding matrix elements

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

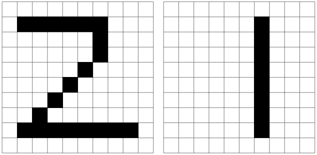
→

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

→

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

A Machine Learning Solution

Training data are represented as pixel matrices: 

Classifier is parameterized by weight matrix of same dimension.

Training procedure:

1. When observe "2", add 1 to corresponding matrix elements
2. When observe "non-2", subtract 1 to corresponding matrix elements

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\rightarrow$

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

$\rightarrow$

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

Test procedure: given new image, take sum of element-wise product.
If positive, predict "2"; else predict "non-2".

Today's Topics

1. Machine Learning basics
   - Why Machine Learning is needed?
   - Main Concepts: Generalization, Model Expressiveness, Overfitting
   - Formal Notation
   - Experiment Design

Generalization $\neq$ Memorization

Key Issue in Machine Learning: Training data is limited

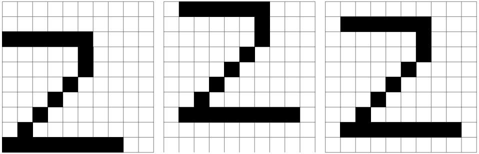
- If the classifier just memorizes the training data, it may perform poorly on new data
- "Generalization" is ability to extend accurate predictions to new data

Generalization $\neq$ Memorization

Key Issue in Machine Learning: Training data is limited

- If the classifier just memorizes the training data, it may perform poorly on new data
- "Generalization" is ability to extend accurate predictions to new data

E.g. consider shifted image:



Will this classifier generalize?

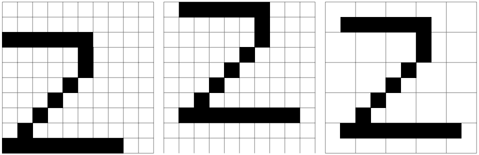
0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

Generalization ≠ Memorization

One potential way to increase generalization ability:

- Discretize weight matrix with larger grids (fewer weights to train)

E.g. consider shifted image:



Now will this classifier generalize?

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

$\rightarrow$

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	-1	0	0	0
0	0	0	0	1	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0	0
0	0	1	0	0	0	-1	0	0	0
0	1	1	1	1	1	-1	1	1	0
0	0	0	0	0	0	0	0	0	0

$\rightarrow$

1	1	1	0	0
0	0	0	0	0
0	0	1	-1	0
0	1	0	-1	0
1	1	1	0	1

Model Expressiveness and Overfitting

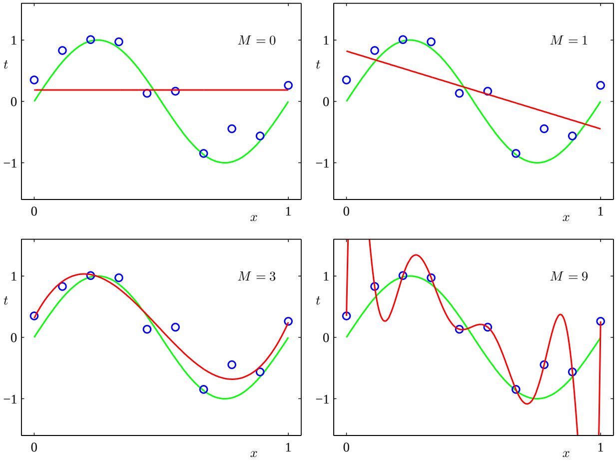
- A model with more weight parameters may fit training data better
- But since training data is limited, expressive model stand the risk of overfitting to peculiarities of the data.

$$\begin{array}{ccc} \text{Less Expressive Model} & \Longleftrightarrow & \text{More Expressive Model} \\ \text{(fewer weights)} & & \text{(more weights)} \end{array}$$

$$\text{Underfit training data} \Longleftrightarrow \text{Overfit training data}$$

Model Expressiveness and Overfitting

Fitting the training data (blue points: x_n)
with a polynomial model: $f(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$
under squared error objective $\frac{1}{2}\sum_n (f(x_n) - t_n)^2$



from PRML Chapter 1 [?]

Today's Topics

1. Machine Learning basics
   - Why Machine Learning is needed?
   - Main Concepts: Generalization, Model Expressiveness, Overfitting
   - Formal Notation
   - Experiment Design

Basic Problem Setup in Machine Learning

- Training Data: a set of $(x^{(m)}, y^{(m)})_{m=\{1,2,..M\}}$ pairs, where input $x^{(m)} \in R^d$ and output $y^{(m)} = \{0, 1\}$
  - e.g. x=vectorized image pixels, y=2 or non-2
- Goal: Learn function $f : x \rightarrow y$ to predicts correctly on new inputs x.

Basic Problem Setup in Machine Learning

- Training Data: a set of $(x^{(m)}, y^{(m)})_{m=\{1,2,..M\}}$ pairs, where input $x^{(m)} \in R^d$ and output $y^{(m)} = \{0, 1\}$
    - e.g. x=vectorized image pixels, y=2 or non-2
- Goal: Learn function $f : x \rightarrow y$ to predicts correctly on new inputs x.
    - Step 1: Choose a function model family:
        - ★ e.g. logistic regression, support vector machines, neural networks

Basic Problem Setup in Machine Learning

- Training Data: a set of $(x^{(m)}, y^{(m)})_{m=\{1,2,..M\}}$ pairs, where input $x^{(m)} \in R^d$ and output $y^{(m)} = \{0, 1\}$
  - e.g. x=vectorized image pixels, y=2 or non-2
- Goal: Learn function $f : x \to y$ to predicts correctly on new inputs x.
  - Step 1: Choose a function model family:
    - e.g. logistic regression, support vector machines, neural networks
  - Step 2: Optimize parameters w on the Training Data
    - e.g. minimize loss function $\min_w \sum_{m=1}^{M} (f_w(x^{(m)}) - y^{(m)})^2$

Today's Topics

1. Machine Learning basics
   - Why Machine Learning is needed?
   - Main Concepts: Generalization, Model Expressiveness, Overfitting
   - Formal Notation
   - Experiment Design

Experiment Design

- Training Data: used to learn function $f()$
- Development / Validation Data: used to evaluate how good $f()$ is, to make high-level model selection decisions, e.g.
  - Which machine learning model to deploy?
  - Tradeoff hyperparameter between regularizer and likelihood?
- Test Data: used to *really* evaluate how good $f()$ is

Experiment Design

- Training Data: used to learn function $f()$
- Development / Validation Data: used to evaluate how good $f()$ is, to make high-level model selection decisions, e.g.
  - Which machine learning model to deploy?
  - Tradeoff hyperparameter between regularizer and likelihood?
- Test Data: used to *really* evaluate how good $f()$ is
- IMPORTANT:
  - Don't train on test data

Experiment Design

- Training Data: used to learn function $f()$
- Development / Validation Data: used to evaluate how good $f()$ is, to make high-level model selection decisions, e.g.
  - Which machine learning model to deploy?
  - Tradeoff hyperparameter between regularizer and likelihood?
- Test Data: used to *really* evaluate how good $f()$ is
- IMPORTANT:
  - Don't train on test data
  - Don't do model selection on test data

Experiment Design

- Training Data: used to learn function $f()$
- Development / Validation Data: used to evaluate how good $f()$ is, to make high-level model selection decisions, e.g.
  - Which machine learning model to deploy?
  - Tradeoff hyperparameter between regularizer and likelihood?
- Test Data: used to *really* evaluate how good $f()$ is
- IMPORTANT:
  - Don't train on test data
  - Don't do model selection on test data
  - Be careful in your experiment design, so that you aren't fooled by over-optimism when deploying a model in real life!

Bias-Variance Tradeoff

- When deploying a model $f()$, we're really interested in the expected error/loss on unseen data
- But we only have estimates based on dev loss

Bias-Variance Tradeoff

- When deploying a model $f()$, we're really interested in the expected error/loss on unseen data
- But we only have estimates based on dev loss
- This estimated loss can be viewed as bias + variance
  - bias: errors from simplifying assumptions in the model $f()$.
  - variance: A different sample of the training set may have resulted in a very different $f()$.
  - Imagine throwing many darts: bias = closeness to bullseye, variance = dispersion of darts

Bias-Variance Tradeoff

- When deploying a model $f()$, we're really interested in the expected error/loss on unseen data
- But we only have estimates based on dev loss
- This estimated loss can be viewed as bias + variance
  - bias: errors from simplifying assumptions in the model $f()$.
  - variance: A different sample of the training set may have resulted in a very different $f()$.
  - Imagine throwing many darts: bias = closeness to bullseye, variance = dispersion of darts
- Generally, complex functions fit the data better, so have low bias but more susceptible to high variance

K-Fold Cross-Validation

- Suppose we can't afford to hold out a dev set out of our training set.

K-Fold Cross-Validation

- Suppose we can't afford to hold out a dev set out of our training set.
- Divide data into K parts, e.g. K=5
  - Fold 1, 2, 3, 4 as training, Fold 5 as dev
  - Fold 2, 3, 4, 5 as training, Fold 1 as dev
  - Fold 3, 4, 5, 1 as training, Fold 2 as dev
  - Fold 4, 5, 1, 2 as training, Fold 3 as dev
  - Fold 5, 1, 2, 3 as training, Fold 4 as dev

K-Fold Cross-Validation

- Suppose we can't afford to hold out a dev set out of our training set.
- Divide data into K parts, e.g. K=5
  - Fold 1, 2, 3, 4 as training, Fold 5 as dev
  - Fold 2, 3, 4, 5 as training, Fold 1 as dev
  - Fold 3, 4, 5, 1 as training, Fold 2 as dev
  - Fold 4, 5, 1, 2 as training, Fold 3 as dev
  - Fold 5, 1, 2, 3 as training, Fold 4 as dev
- Run the same algorithm on all K cases, compute average dev loss.

K-Fold Cross-Validation

- Suppose we can't afford to hold out a dev set out of our training set.
- Divide data into K parts, e.g. K=5
  - Fold 1, 2, 3, 4 as training, Fold 5 as dev
  - Fold 2, 3, 4, 5 as training, Fold 1 as dev
  - Fold 3, 4, 5, 1 as training, Fold 2 as dev
  - Fold 4, 5, 1, 2 as training, Fold 3 as dev
  - Fold 5, 1, 2, 3 as training, Fold 4 as dev
- Run the same algorithm on all K cases, compute average dev loss.
- Select model based on avg. dev loss, then evaluate on test.

K-Fold Cross-Validation

- Suppose we can't afford to hold out a dev set out of our training set.
- Divide data into K parts, e.g. K=5
  - Fold 1, 2, 3, 4 as training, Fold 5 as dev
  - Fold 2, 3, 4, 5 as training, Fold 1 as dev
  - Fold 3, 4, 5, 1 as training, Fold 2 as dev
  - Fold 4, 5, 1, 2 as training, Fold 3 as dev
  - Fold 5, 1, 2, 3 as training, Fold 4 as dev
- Run the same algorithm on all K cases, compute average dev loss.
- Select model based on avg. dev loss, then evaluate on test.
- How to pick K? Consider computation and Bias-Variance tradeoff

No Free Lunch Theorem

- No machine learning method is best for all datasets
- You must learn to choose an appropriate model family and optimization algorithm for your task
- Don't trust anyone who advertises a machine learning method that always wins.