Machine Learning Overview and Applications

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Computational Learning Lab (CLLab)
Department of Computer Science & Information Engineering
National Taiwan University
(國立台灣大學資訊工程系計算學習實驗室)

materials mostly taken from my “Learning from Data” book, my “Machine Learning Foundations” free online course, and works from NTU CLLab and NTU KDDCup teams
What is Machine Learning
The Learning Problem
What is Machine Learning

From Learning to Machine Learning

Learning: acquiring skill with experience accumulated from observations

Observations → Learning → Skill

Machine learning: acquiring skill with experience accumulated/computed from data

Data → ML → Skill

What is skill?
The Learning Problem

What is Machine Learning

A More Concrete Definition

skill
⇔ improve some performance measure (e.g. prediction accuracy)

machine learning: improving some performance measure with experience computed from data

An Application in Computational Finance

stock data → ML → more investment gain

Why use machine learning?
Yet Another Application: Tree Recognition

- ‘define’ trees and hand-program: difficult
- learn from data (observations) and recognize: a 3-year-old can do so
- ‘ML-based tree recognition system’ can be easier to build than hand-programmed system

ML: an alternative route to build complicated systems
The Machine Learning Route

ML: an **alternative route** to build complicated systems

Some Use Scenarios

- when human cannot program the system manually
  — **navigating on Mars**

- when human cannot ‘define the solution’ easily
  — **speech/visual recognition**

- when needing rapid decisions that humans cannot do
  — **high-frequency trading**

- when needing to be user-oriented in a massive scale
  — **consumer-targeted marketing**

Give a **computer** a fish, you feed it for a day; teach it how to fish, you feed it for a lifetime. :-)

Hsuan-Tien Lin (NTU CSIE)
**Key Essence of Machine Learning**

**machine learning**: improving some **performance measure** with experience **computed** from **data**

1. **exists** some ‘underlying pattern’ to be learned
   —so ‘performance measure’ can be improved
2. **but no** programmable (easy) **definition**
   —so ‘ML’ is needed
3. **somehow** there is **data** about the pattern
   —so **ML** has some ‘inputs’ to learn from

**key essence**: help decide whether to use **ML**
Snapshot Applications of Machine Learning
Daily Needs: Food, Clothing, Housing, Transportation

1. **Food** (Sadilek et al., 2013)
   - **data**: Twitter data (words + location)
   - **skill**: tell food poisoning likeliness of restaurant properly

2. **Clothing** (Abu-Mostafa, 2012)
   - **data**: sales figures + client surveys
   - **skill**: give good fashion recommendations to clients

3. **Housing** (Tsanas and Xifara, 2012)
   - **data**: characteristics of buildings and their energy load
   - **skill**: predict energy load of other buildings closely

4. **Transportation** (Stallkamp et al., 2012)
   - **data**: some traffic sign images and meanings
   - **skill**: recognize traffic signs accurately

**ML is everywhere!**
Education

- **data**: students’ records on quizzes on a Math tutoring system
- **skill**: predict whether a student can give a correct answer to another quiz question

### A Possible ML Solution

answer correctly \( \approx [ \text{recent strength of student} > \text{difficulty of question} ] \)

- give ML 9 million records from 3000 students
- ML determines *(reverse-engineers)* strength and difficulty automatically

Key part of the **world-champion** system from National Taiwan Univ. in KDDCup 2010
Entertainment: Recommender System (1/2)

- **data**: how many users have rated some movies
- **skill**: predict how a user would rate an unrated movie

**A Hot Problem**
- competition held by Netflix in 2006
  - 100,480,507 ratings that 480,189 users gave to 17,770 movies
  - 10% improvement = 1 million dollar prize
- similar competition (movies $\rightarrow$ songs) held by Yahoo! in KDDCup 2011
  - 252,800,275 ratings that 1,000,990 users gave to 624,961 songs

How can machines learn our preferences?
The Learning Problem
Snapshot Applications of Machine Learning
Entertainment: Recommender System (2/2)

A Possible ML Solution

- pattern: 
  rating ← viewer/movie factors

- learning:
  known rating
  → learned factors
  → unknown rating prediction

Key part of the world-champion (again!) system from National Taiwan Univ. in KDDCup 2011

Hsuan-Tien Lin (NTU CSIE)
Machine Learning Overview and Applications
Components of Machine Learning
Components of Learning: Metaphor Using Credit Approval

Applicant Information

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

unknown pattern to be learned: ‘approve credit card good for bank?’
Formalize the Learning Problem

Basic Notations

- **input**: $x \in \mathcal{X}$ (customer application)
- **output**: $y \in \mathcal{Y}$ (good/bad after approving credit card)
- **unknown pattern to be learned** $\Leftrightarrow$ **target function**: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (ideal credit approval formula)
- **data** $\Leftrightarrow$ **training examples**: $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ (historical records in bank)
- **hypothesis** $\Leftrightarrow$ **skill** with hopefully **good performance**: $g : \mathcal{X} \rightarrow \mathcal{Y}$ (‘learned’ formula to be used)
Learning Flow for Credit Approval

unknown target function
\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]
(ideal credit approval formula)

training examples
\[ \mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N) \]
(historical records in bank)

learning algorithm \[ \mathcal{A} \]

final hypothesis
\[ g \approx f \]
(‘learned’ formula to be used)

- target \( f \) **unknown**
  (i.e. no programmable definition)
- hypothesis \( g \) hopefully \( \approx f \)
  but possibly **different** from \( f \)
  (perfection ‘impossible’ when \( f \) unknown)

What does \( g \) look like?
The Learning Model

The Learning Problem

Components of Machine Learning

training examples $\mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N)$

(historical records in bank)

learning algorithm $\mathcal{A}$

final hypothesis $g \approx f$

('learned' formula to be used)

hypothesis set $\mathcal{H}$

(set of candidate formula)

- assume $g \in \mathcal{H} = \{h_k\}$, i.e. approving if
  - $h_1$: annual salary > NTD 800,000
  - $h_2$: debt > NTD 100,000 (really?)
  - $h_3$: year in job $\leq 2$ (really?)

- hypothesis set $\mathcal{H}$:
  - can contain good or bad hypotheses
  - up to $\mathcal{A}$ to pick the 'best' one as $g$

learning model $= \mathcal{A}$ and $\mathcal{H}$
Practical Definition of Machine Learning

unknown target function

\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]

(ideal credit approval formula)

training examples

\[ \mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N) \]

(historical records in bank)

learning algorithm

\[ \mathcal{A} \]

(final hypothesis

\[ g \approx f \]

(‘learned’ formula to be used)

hypothesis set

\[ \mathcal{H} \]

(set of candidate formula)

machine learning:

use data to compute hypothesis \( g \) that approximates target \( f \)
Learning with Different Output Space $\mathcal{Y}$
Credit Approval Problem Revisited

**The Learning Problem**

Learning with Different Output Space $\mathcal{Y}$

**unknown target function**

$f : \mathcal{X} \rightarrow \mathcal{Y}$

(ideal credit approval formula)

**training examples**

$\mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N)$

(historical records in bank)

**hypothesis set**

$\mathcal{H}$

(set of candidate formula)

**$\mathcal{Y} = \{-1, +1\}$: binary classification**

**age** 23 years

**gender** female

**annual salary** NTD 1,000,000

**year in residence** 1 year

**year in job** 0.5 year

**current debt** 200,000

**credit?** \{no$(-1)$, yes$(+1)$\}

**learning algorithm** $\mathcal{A}$

$g \approx f$

('learned' formula to be used)

**final hypothesis**

Hsuan-Tien Lin (NTU CSIE)
More Binary Classification Problems

- credit approve/disapprove
- email spam/non-spam
- patient sick/not sick
- ad profitable/not profitable
- answer correct/incorrect (KDDCup 2010)

Core and important problem with many tools as building block of other tools
Multiclass Classification: Coin Recognition Problem

- classify US coins (1c, 5c, 10c, 25c) by (size, mass)
- $\mathcal{Y} = \{1c, 5c, 10c, 25c\}$, or
  $\mathcal{Y} = \{1, 2, \cdots, K\}$ (abstractly)
- binary classification: special case with $K = 2$

Other Multiclass Classification Problems

- written digits $\Rightarrow$ 0, 1, $\cdots$, 9
- pictures $\Rightarrow$ apple, orange, strawberry
- emails $\Rightarrow$ spam, primary, social, promotion, update (Google)

many applications in practice, especially for ‘recognition’
Regression: Patient Recovery Prediction Problem

- binary classification: patient features $\Rightarrow$ sick or not
- multiclass classification: patient features $\Rightarrow$ which type of cancer
- regression: patient features $\Rightarrow$ **how many days before recovery**
- $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [\text{lower}, \text{upper}] \subset \mathbb{R}$ (bounded regression)
  —**deeply studied in statistics**

Other Regression Problems

- company data $\Rightarrow$ stock price
- climate data $\Rightarrow$ temperature

also core and important with many ‘statistical’ tools as **building block of other tools**
Mini Summary

Learning with Different Output Space $\mathcal{Y}$

- **binary classification**: $\mathcal{Y} = \{-1, +1\}$
- **multiclass classification**: $\mathcal{Y} = \{1, 2, \cdots, K\}$
- **regression**: $\mathcal{Y} = \mathbb{R}$
- ... and a lot more!!

core tools: binary classification and regression

unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$

training examples $\mathcal{D}: (x_1, y_1), \cdots, (x_N, y_N)$

learning algorithm $A$

final hypothesis $g \approx f$

hypothesis set $\mathcal{H}$
Learning with Different Data Label $y_n$
Supervised: Coin Recognition Revisited

unknown target function
\( f : \mathcal{X} \rightarrow \mathcal{Y} \)

training examples
\( \mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N) \)

learning algorithm
\( \mathcal{A} \)

final hypothesis
\( g \approx f \)

supervised learning:
\( \text{every } x_n \text{ comes with corresponding } y_n \)
Unsupervised: Coin Recognition without $y_n$

supervised multiclass classification

unsupervised multiclass classification $\iff$ 'clustering'

Other Clustering Problems

- articles $\Rightarrow$ topics
- consumer profiles $\Rightarrow$ consumer groups

clustering: a challenging but useful problem
Unsupervised: Coin Recognition without $y_n$

supervised multiclass classification

unsupervised multiclass classification $\iff$ 'clustering'

Other Clustering Problems
- articles $\Rightarrow$ topics
- consumer profiles $\Rightarrow$ consumer groups

clustering: a challenging but useful problem
Unsupervised: Learning without $y_n$

Other Unsupervised Learning Problems

- **clustering**: $\{x_n\} \Rightarrow \text{cluster}(x)$
  $(\approx \text{‘unsupervised multiclass classification’})$
  —i.e. articles $\Rightarrow$ topics

- **density estimation**: $\{x_n\} \Rightarrow \text{density}(x)$
  $(\approx \text{‘unsupervised bounded regression’})$
  —i.e. traffic reports with location $\Rightarrow$ dangerous areas

- **outlier detection**: $\{x_n\} \Rightarrow \text{unusual}(x)$
  $(\approx \text{extreme ‘unsupervised binary classification’})$
  —i.e. Internet logs $\Rightarrow$ intrusion alert

- ... and a lot more!!

unsupervised learning: diverse, with possibly very different performance goals
Semi-supervised: Coin Recognition with Some $y_n$

Other Semi-supervised Learning Problems
- face images with a few labeled $\Rightarrow$ face identifier (Facebook)
- medicine data with a few labeled $\Rightarrow$ medicine effect predictor

**semi-supervised learning**: leverage unlabeled data to avoid ‘expensive’ labeling
Reinforcement Learning

a ‘very different’ but natural way of learning

Teach Your Dog: Say ‘Sit Down’

*The dog pees on the ground.*

**BAD DOG. THAT’S A VERY WRONG ACTION.**

- cannot easily show the dog that \( y_n = \) sit when \( x_n = \) ‘sit down’
- but can ‘punish’ to say \( \tilde{y}_n = \) pee is wrong

Other Reinforcement Learning Problems Using \((x, \tilde{y}, \text{goodness})\)

- (customer, ad choice, ad click earning) \(\Rightarrow\) ad system
- (cards, strategy, winning amount) \(\Rightarrow\) black jack agent

reinforcement: learn with ‘partial/implicit information’ (often sequentially)
Reinforcement Learning

a ‘very different’ but natural way of learning

Teach Your Dog: Say ‘Sit Down’

*The dog sits down.*

Good Dog. Let me give you some cookies.

- still cannot show $y_n = \text{sit}$ when $x_n = \text{‘sit down’}$
- but can ‘reward’ to say $\tilde{y}_n = \text{sit is good}$

Other Reinforcement Learning Problems Using $(x, \tilde{y}, \text{goodness})$

- (customer, ad choice, ad click earning) $\Rightarrow$ ad system
- (cards, strategy, winning amount) $\Rightarrow$ black jack agent

reinforcement: learn with ‘partial/implicit information’ (often sequentially)
The Learning Problem

Learning with Different Data Label $y_n$

**Mini Summary**

Learning with Different Data Label $y_n$

- **supervised**: all $y_n$
- **unsupervised**: no $y_n$
- **semi-supervised**: some $y_n$
- **reinforcement**: implicit $y_n$ by goodness($\tilde{y}_n$)
- ... and more!!

---

**Diagram**

- **unknown target function** $f: \mathcal{X} \rightarrow \mathcal{Y}$
- **training examples** $\mathcal{D}: (x_1, y_1), \cdots, (x_N, y_N)$
- **core tool**: supervised learning
- **final hypothesis** $g \approx f$

- **learning algorithm** $A$
- **hypothesis set** $\mathcal{H}$
The Learning Problem

Learning with Different Protocol $f \Rightarrow (x_n, y_n)$
The Learning Problem

Learning with Different Protocol \( f \mapsto (x_n, y_n) \)

Batch Learning: Coin Recognition Revisited

unknown target function
\( f: \mathcal{X} \rightarrow \mathcal{Y} \)

training examples
\( \mathcal{D}: (x_1, y_1), \cdots, (x_N, y_N) \)

learning algorithm \( \mathcal{A} \)

final hypothesis
\( g \approx f \)

hypothesis set \( \mathcal{H} \)

**batch** supervised multiclass classification: learn from **all known** data
More Batch Learning Problems

- batch of (email, spam?) $\Rightarrow$ spam filter
- batch of (patient, cancer) $\Rightarrow$ cancer classifier
- batch of patient data $\Rightarrow$ group of patients

batch learning: a very common protocol
Online: Spam Filter that ‘Improves’

- batch spam filter:
  learn with known (email, spam?) pairs, and predict with fixed $g$
- **online** spam filter, which **sequentially**:
  1. observe an email $x_t$
  2. predict spam status with current $g_t(x_t)$
  3. receive ‘desired label’ $y_t$ from user, and then update $g_t$ with $(x_t, y_t)$

Connection to What We Have Learned

- PLA can be easily adapted to online protocol (how?)
- reinforcement learning is often done online (why?)

online: hypothesis ‘improves’ through receiving data instances **sequentially**
Active Learning: Learning by ‘Asking’

Protocol $\leftrightarrow$ Learning Philosophy
- batch: ‘duck feeding’
- online: ‘passive sequential’
- active: ‘question asking’ (sequentially) —query the $y_n$ of the chosen $x_n$

unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$

training examples $\mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N)$

learning algorithm $\mathcal{A}$

final hypothesis $g \approx f$

hypothesis set $\mathcal{H}$

active: improve hypothesis with fewer labels (hopefully) by asking questions **strategically**

Hsuan-Tien Lin (NTU CSIE)
The Learning Problem

Learning with Different Protocol $f \Rightarrow (x_n, y_n)$

Mini Summary

Learning with Different Protocol $f \Rightarrow (x_n, y_n)$

- **batch**: all known data
- **online**: sequential (passive) data
- **active**: strategically-observed data
- ... and more!!

unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$

training examples $\mathcal{D}: (x_1, y_1), \ldots, (x_N, y_N)$

learning algorithm $\mathcal{A}$

final hypothesis $g \approx f$

hypothesis set $\mathcal{H}$

core protocol: batch
Learning with Different Input Space $\mathcal{X}$
Credit Approval Problem Revisited

unknown target function
\( f : \mathcal{X} \rightarrow \mathcal{Y} \)

(ideal credit approval formula)

training examples
\( \mathcal{D}: (x_1, y_1), \cdots, (x_N, y_N) \)

(historical records in bank)

learning algorithm
\( \mathcal{A} \)

final hypothesis
\( g \approx f \)

('learned' formula to be used)

hypothesis set
\( \mathcal{H} \)

(set of candidate formula)

concrete features: each dimension of \( \mathcal{X} \subseteq \mathbb{R}^d \) represents 'sophisticated physical meaning'

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>
More on Concrete Features

- **(size, mass)** for coin classification
- **customer info** for credit approval
- **patient info** for cancer diagnosis
- Often including ‘human intelligence’ on the learning task

Concrete features: the ‘easy’ ones for ML
Raw Features: Digit Recognition Problem (1/2)

- digit recognition problem: features $\Rightarrow$ meaning of digit
- a typical supervised multiclass classification problem
Raw Features: Digit Recognition Problem (2/2)

by Concrete Features:
- Digit images: X
- 16 by 16 gray images:
  - x = (symmetry, density)

by Raw Features:
- 16 by 16 gray image:
  - x ≡ (0, 0, 0.9, 0.6, ...) ∈ R^{256}
- ‘simple’ physical meaning;
  - thus more difficult for ML than concrete features

Other Problems with Raw Features:
- image pixels, speech signal, etc.

raw features: often need human or machines to convert to concrete ones
Time Features: Stock Prediction Problem

Stock Prediction Problem
- given previous (time, price) pairs, predict whether the price would go up or down tomorrow?
- a ‘binary classification’ problem (or a regression one if predicting the price itself)
- $\mathcal{X} \subset \mathbb{R}$ representing time, $\mathcal{Y} = \mathbb{R}^+$ representing price

Other Problems with Time Features
- timestamp when student performance in online tutoring system (KDDCup 2010)
- rating time given by user in recommender system (KDDCup 2011)


time features: can carry trend
Abstract Features: Rating Prediction Problem

Rating Prediction Problem (KDDCup 2011)

- given previous (userid, itemid, rating) tuples, predict the rating that some userid would give to itemid?
- a regression problem with $\mathcal{Y} \subseteq \mathbb{R}$ as rating and $\mathcal{X} \subseteq \mathbb{N} \times \mathbb{N}$ as (userid, itemid)
- ‘no physical meaning’; thus even more difficult for ML

Other Problems with Abstract Features

- student ID in online tutoring system (KDDCup 2010)
- advertisement ID in online ad system

abstract: again need ‘feature conversion/extraction/construction’
Mini Summary

Learning with Different Input Space $\mathcal{X}$

- **concrete**: sophisticated (and related) physical meaning
- **raw**: simple physical meaning
- **time**: some trends
- **abstract**: no (or little) physical meaning
- ... and more!!

unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$

| training examples $\mathcal{D} : (x_1, y_1), \cdots, (x_N, y_N)$ |
| learning algorithm $A$ |
| final hypothesis $g \approx f$ |

hypothesis set $\mathcal{H}$

'easy' input: concrete
Machine Learning Research in CLLab
Oracle: truth $f(x) + \text{noise } e(x)$

(4) data (instance $x_n$, label $y_n$)

(1) learning algorithm

(3) good learning system $g(x)$

(2) learning model \{h(x)\}

CLLab Works: **Loosen the Limits of ML**

1. cost-sensitive classification: limited protocol (classification) + **auxiliary info. (cost)**
2. multi-label classification: limited protocol (classification) + **structure info. (label relation)**
3. active learning: limited protocol (unlabeled data) + **requested info. (query)**
4. online learning: limited protocol (streaming data) + **feedback info. (loss)**

next: **(1) cost-sensitive classification**
Which Digit Did You Write?

one (1)  two (2)  three (3)

a *classification* problem —grouping “pictures” into different “categories”
Traditional Classification Problem

Oracle: truth \( f(x) + \text{noise } e(x) \)

\[
data \text{ (instance } x_n, \text{ label } y_n)\]

\[
good \text{ learning system } g(x)\]

\[
\text{learning model } \{g_\alpha(x)\}\]

1. input: a batch of examples (digit \( x_n \), intended label \( y_n \))
2. desired output: some \( g(x) \) such that \( g(x) \neq y \text{ seldom} \) for future examples \((x, y)\)
3. evaluation for some digit

\[
(x = 2, y = 2)\]

\[
\neg g(x) = \begin{cases} 
1 : \text{ wrong}; \\
2 : \text{ right}; \\
3 : \text{ wrong}
\end{cases}
\]

Are all the \textbf{wrongs} equally bad?
What is the Status of the Patient?

H1N1-infected  
cold-infected  
healthy

another **classification** problem  
—grouping “patients” into different “status”
The Learning Problem

Machine Learning Research in CLLab

Patient Status Prediction

error measure = society cost

<table>
<thead>
<tr>
<th>actual</th>
<th>predicted</th>
<th>H1N1</th>
<th>cold</th>
<th>healthy</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1N1</td>
<td>0</td>
<td>1000</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>cold</td>
<td>100</td>
<td>0</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>healthy</td>
<td>100</td>
<td>30</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- H1N1 mis-predicted as healthy: very high cost
- cold mis-predicted as healthy: high cost
- cold correctly predicted as cold: no cost

human doctors consider costs of decision; can computer-aided diagnosis do the same?
### Our Contributions

<table>
<thead>
<tr>
<th></th>
<th>binary</th>
<th>multiclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>well-studied</td>
<td>well-studied</td>
</tr>
<tr>
<td>cost-sensitive</td>
<td>known (Zadrozny, 2003)</td>
<td>ongoing (our works)</td>
</tr>
</tbody>
</table>

**theoretic, algorithmic and empirical studies of cost-sensitive classification**

- ICML 2010: a theoretically-supported algorithm with superior experimental results
- BIBM 2011: application to real-world bacteria classification with promising experimental results
- KDD 2012: a cost-sensitive and error-sensitive methodology (achieving both low cost and few wrongs)
Making Machine Learning Realistic: Next

Teacher

\[ \text{cost } c(t) \]

\[ \text{query } x(t) \& \text{ guess } \hat{y}(t) \]

learning algorithm

knowledge \( \mathcal{X} \)

learning model

Interactive Machine Learning

1. environment
2. exploration
3. dynamic
4. partial feedback

let us teach machines as “easily” as teaching students
Case: Interactive Learning for Online Advertisement

### Traditional Machine Learning for Online Advertisement

- **data gathering**: system randomly shows ads to some previous users
- **expert building**: system analyzes data gathered to determine best (fixed) strategy

### Interactive Machine Learning for Online Advertisement

- **environment**: system serves online users with profile
- **exploration**: system decides to show an ad to the user
- **dynamic**: system receives data from real-time user click
- **partial feedback**: system receives reward only if clicking
The Learning Problem

Machine Learning Research in CLab

Preliminary Success: ICML 2012 Exploration & Exploitation Challenge

Interactive Machine Learning for Online Advertisement

- **environment**: system serves **online users with profile**
- **exploration**: system **decides to show an ad** to the user
- **dynamic**: system receives data from **real-time user click**
- **partial feedback**: system receives **reward only if clicking**

---

**NTU beats two MIT teams to be the phase 1 winner!**

<table>
<thead>
<tr>
<th>NAME</th>
<th>AFFILIATION</th>
<th>LAST SCORE (CTR * 10,000)</th>
<th>BEST SCORE (CTR * 10,000)</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ku-Chun</td>
<td>NTU</td>
<td>882.9</td>
<td>905.9</td>
<td>1</td>
</tr>
<tr>
<td>tvirot</td>
<td>MIT</td>
<td>903.9</td>
<td>903.9</td>
<td>2</td>
</tr>
<tr>
<td>edjoesu</td>
<td>MIT</td>
<td>889.9</td>
<td>903.4</td>
<td>3</td>
</tr>
</tbody>
</table>

interactive: more challenging than traditional machine learning, but **realistic**
More on KDDCup
What is KDDCup?

Background

- an annual competition on KDD (knowledge discovery and data mining)
- organized by ACM SIGKDD, starting from 1997, now the most prestigious data mining competition
- usually lasts 3-4 months
- participants include famous research labs (IBM, AT&T) and top universities (Stanford, Berkeley)
Aim of KDDCup

Aim

- bridge the gap between theory and **practice**, such as
  - scalability and efficiency
  - missing data and noise
  - heterogeneous data
  - unbalanced data
  - combination of different models
- define the **state-of-the-art**
The Learning Problem

More on KDDCup

KDDCups: 2008 to 2013

2008

- organizer: Siemens
- topic: breast cancer prediction (medical)
- data size: 0.2M
- teams: > 200
- NTU: **co-champion** with IBM (led by Prof. Shou-de Lin)

2009

- organizer: Orange
- topic: customer behavior prediction (business)
- data size: 0.1M
- teams: > 400
- NTU: **3rd place** of slow track
## KDDCups: 2008 to 2013 II

### 2010
- **organizer:** PSLC Data Shop
- **topic:** student performance prediction (education)
- **data size:** 30M
- **teams:** > 100
- **NTU:** champion and student-team champion

### 2011
- **organizer:** Yahoo!
- **topic:** music preference prediction (recommendation)
- **data size:** 300M
- **teams:** > 1000
- **NTU:** double champions
### KDDCups: 2008 to 2013 III

#### 2012
- **organizer:** Tencent
- **topic:** webuser behavior prediction (Internet)
- **data size:** 150M
- **teams:** > 800
- **NTU:** *champion of track 2*

#### 2013
- **organizer:** Microsoft Research
- **topic:** paper-author relationship prediction (academia)
- **data size:** 600M
- **teams:** > 500
- **NTU:** *double champions*
KDDCup 2011

Music Recommendation Systems

- host: Yahoo!
- 11 years of Yahoo! music data
- 2 tracks of competition
- official dates: March 15 to June 30
- 1878 teams submitted to track 1; 1854 teams submitted to track 2

from

Yahoo! Labs

KDD Cup

from

Yahoo! Labs
NTU Team for KDDCup 2011

- 3 faculties: Profs. Chih-Jen Lin, Hsuan-Tien Lin and Shou-De Lin
- 1 course (starting in 2010) Data Mining and Machine Learning: Theory and Practice
- 3 TAs and 19 students: most were inexperienced in music recommendation in the beginning
- official classes: April to June; actual classes: December to June

our motto: study state-of-the-art approaches and then creatively improve them
Previously: How Much Did You Like These Movies?

http://www.netflix.com

(1M dollar competition between 2007-2009)

Get Recommendations (27)  Rate Movies  Movies You've Rated (5)

How much did you like these movies?

The Wedding Planner  How to Lose a Guy in 10 Days  Sweet Home Alabama  Pretty Woman

goal: use “movies you’ve rated” to automatically predict your preferences on future movies
The Track 1 Problem (1/2)

### Given Data

263M examples (user $u$, item $i$, rating $r_{ui}$, date $t_{ui}$, time $\tau_{ui}$)

<table>
<thead>
<tr>
<th>user</th>
<th>item</th>
<th>rating</th>
<th>date</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>10</td>
<td>102</td>
<td>23:52</td>
</tr>
<tr>
<td>1</td>
<td>213</td>
<td>90</td>
<td>1032</td>
<td>21:01</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>95</td>
<td>768</td>
<td>09:15</td>
</tr>
</tbody>
</table>

- $u$, $i$: abstract IDs
- $r_{ui}$: integer between 0 and 100, **mostly multiples of 10**

### Additional Information: Item Hierarchy

- track (46.85%)
- album (19.01%)
- artist (28.84%)
- genre (5.30%)
Data Partitioned by Organizers

- training: $253M$; validation: $4M$; test (w/o rating): $6M$
- per user, training $<$ validation $<$ test in time
  - $\geq 20$ examples total
  - 4 examples in validation; 6 in test
- fixed random half of test: leaderboard; another half: award decision

Goal

Predictions $\hat{r}_{ui} \approx r_{ui}$ on the test set, measured by

$$RMSE = \sqrt{\text{average}(\hat{r}_{ui} - r_{ui})^2}$$

— one submission allowed every eight hours
### Three Properties of Track 1 Data

<table>
<thead>
<tr>
<th></th>
<th>track(_1)</th>
<th>track(_2)</th>
<th>album(_3)</th>
<th>author(_4)</th>
<th>(\cdots)</th>
<th>genre(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>user(_1)</td>
<td>100</td>
<td>80</td>
<td>70</td>
<td>?</td>
<td>(\cdots)</td>
<td>(\cdot)</td>
</tr>
<tr>
<td>user(_2)</td>
<td>(\cdot)</td>
<td>(\cdot)</td>
<td>(\cdot)</td>
<td>80</td>
<td>(\cdots)</td>
<td>(\cdot)</td>
</tr>
<tr>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>user(_U)</td>
<td>?</td>
<td>(\cdot)</td>
<td>20</td>
<td>(\cdot)</td>
<td>(\cdots)</td>
<td>0</td>
</tr>
</tbody>
</table>

Similar to Netflix data, but with the following differences......

- **scale**: larger data
  — study mature models that are **computationally feasible**

- **taxonomy**: relation graph of tracks, albums, authors and genres
  — **include as features** for combining models nonlinearly

- **time**: detailed; training earlier than test
  — **include as features** for combining models nonlinearly; respect time-closeness during training
The Learning Problem

More on KDDCup

Framework of Our Solution

System Architecture

- **improve standard models**: design variants within 6 families of state-of-the-art models (reaches RMSE 22.7915)

- **blend the models**: improve prediction power by blending the variants carefully (reaches RMSE 21.3598)

- **aggregate the blended predictors**: construct a linear ensemble with test performance estimators (reaches RMSE 21.0253)

- **post-process the ensemble**: add a final touch based on observations from data analysis (reaches RMSE 21.0147)

not only **hard work** (200+ models included), but also **key techniques**
That’s about all. Thank you!