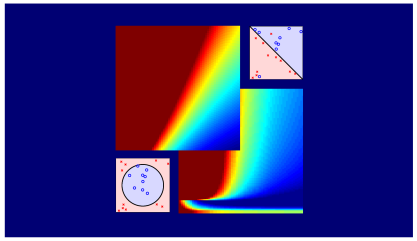
Machine Learning Techniques
(機器學習技巧)



Lecture 14: Miscellaneous Models

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

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Agenda

Lecture 14: Miscellaneous Models

- Matrix Factorization
- Gradient Boosted Decision Tree
- Naive Bayes
- Bayesian Learning

Recommender System Revisited

- 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**
- data $\mathcal{D}_j$ for j-th movie: $\{(\mathbf{x}_n = (i), y_n = r_{ij})\}_{n=1}^{N_j}$
  —abstract features $\mathbf{x}_n = (i)$

how to **learn our preferences** from all $\mathcal{D}_j$?

Linear Model for Recommender System

consider one linear model for each $\mathcal{D}_j = \{(\mathbf{x}_n = (i), y_n = r_{ij})\}_{n=1}^{N_j}$, with a shared transform $\mathbf{\Phi}$:

$$y \approx h_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{\Phi}(\mathbf{x}) \text{ for } j\text{-th movie}$$

- $\mathbf{\Phi}(i)$: named $\mathbf{v}_i$, to be **learned** from data, like NNet/RBF Net
- then, $r_{ij} = y_n \approx \mathbf{w}_j^T \mathbf{v}_i$
- overall E_{in} with squared error:

$$E_{\text{in}}(\{\mathbf{w}_j\}, \{\mathbf{v}_i\}) = \frac{\sum_j N_j E_{\text{in}}^{(j)}(\mathbf{w}_j, \{\mathbf{v}_i\})}{\sum_j N_j} = \frac{1}{N} \sum_{\text{known } (i,j)} (r_{ij} - \mathbf{w}_j^T \mathbf{v}_i)^2$$

how to minimize?
SGD by sampling known (i, j)

Matrix Factorization

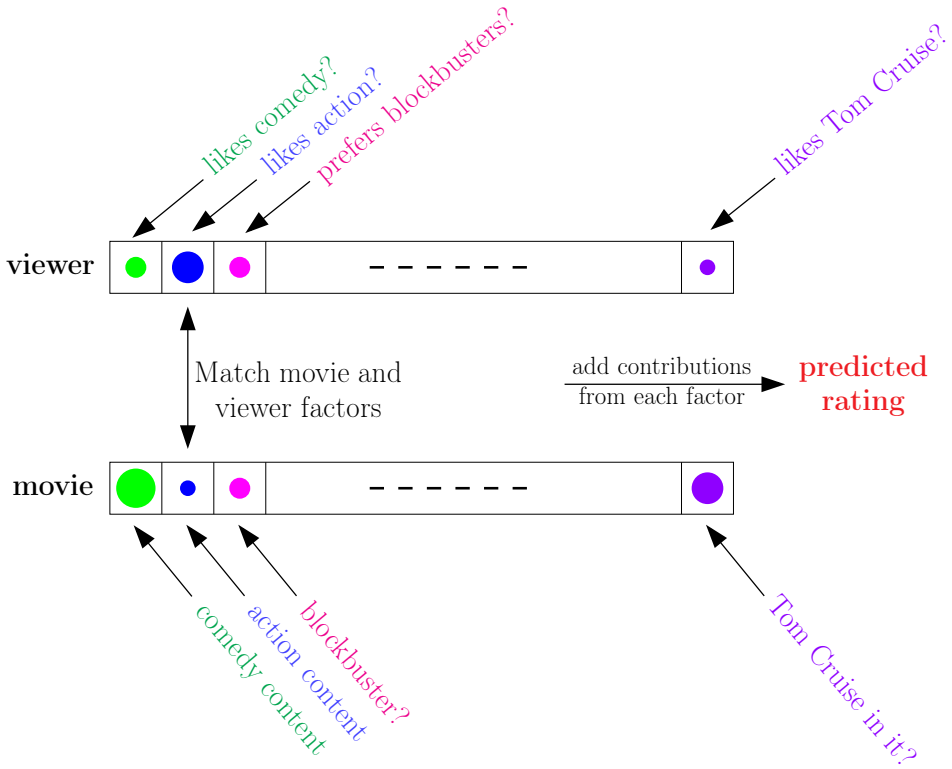
$$r_{ij} \approx \mathbf{w}_j^T \mathbf{v}_i = \mathbf{v}_i^T \mathbf{w}_j$$

R	movie_1	movie_2	$\cdots$	movie_J
user_1	100	?	$\cdots$	–
user_2	–	70	$\cdots$	–
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
user_I	?	–	$\cdots$	0

$\approx$

V
$\mathbf{v}_1^T$
$\mathbf{v}_2^T$
$\cdots$
$\mathbf{v}_I^T$

W	$\mathbf{w}_1$	$\mathbf{w}_2$	$\cdots$	$\mathbf{w}_J$



Matrix Factorization Model

- learning:
  known rating
  $\rightarrow$ learned **factors** $\mathbf{w}_j$ and $\mathbf{v}_i$
  $\rightarrow$ unknown rating prediction

similar modeling can be used for **abstract features**

Fun Time

Coordinate Descent for Linear Blending

Consider a linear blending problem: for $\mathcal{G} = \{g_\ell\}$,

$$\min_{\beta} \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \sum_{\ell=1}^{L} \beta_\ell g_\ell(\mathbf{x}_n)\right)$$

- why exponential error $\exp(-yG(\mathbf{x}))$:
  a **convex upper bound** on $\text{err}_{0/1}$ as $\widehat{\text{err}}$
- how to minimize?
  —GD, SGD, . . . if few $\{g_\ell\}$
- what if lots of or **infinitely many** g_ℓ**?**
  —pick **one good** g_i, and update its β_i only

coordinate descent: in each iteration

- pick a good **coordinate** i (the best one for the next step)
- minimize by setting $\beta_i^{new} \leftarrow \beta_i^{old} + \Delta$

Coordinate Descent View of AdaBoost

Consider a linear blending problem: for $\mathcal{G} = \{g_\ell\}$,

$$\min_{\beta} \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \sum_{\ell=1}^{L} \beta_\ell g_\ell(\mathbf{x}_n)\right)$$

coordinate descent: in each iteration

- pick a good **coordinate** i (the best one for the next step)
- minimize by setting $\beta_i^{new} \leftarrow \beta_i^{old} + \Delta$

AdaBoost: in each iteration

- pick a **good hypothesis** g_t
- set $\alpha_t^{new} \leftarrow 0 + \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

after some derivations (ML2012Fall HW7.5):

AdaBoost = coordinate descent + exponential error

Gradient Boosted Decision Tree

Consider another linear blending problem:

$$\min_{\beta} \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \sum_{\ell=1}^{L}\beta_\ell g_\ell(\mathbf{x}_n)\right)^2$$

- best coordinate at t-th iteration (under assumptions):

$$\min_{g_\ell} \frac{1}{N}\sum_{n=1}^{N}(y_n - G_{t-1}(\mathbf{x}_n) - g_\ell(\mathbf{x}_n))^2$$

  —best **hypothesis** on $\{(\mathbf{x}_n, \text{residual}_n)\}$
- best β_ℓ^{new}: one-dimensional **linear regression**

gradient boosted decision tree (GBDT):
above + find best g_ℓ by decision tree
(a 'regression' extension of AdaBoost)

Fun Time

Naive Bayes Model

want: getting $P(y|\mathbf{x})$ (e.g. logistic regression) for classification

- Bayes rule: $P(y|\mathbf{x}) \propto P(\mathbf{x}|y)P(y)$
- estimating $P(y)$: frequency of $y_n = y$ in $\mathcal{D}$ (easy!)
- joint distribution $P(\mathbf{x}|y)$: easier if

$$P(\mathbf{x}|y) = P(x_1|y)P(x_2|y)\cdots p(x_d|y)$$

—**conditional independence**

- marginal distribution $P(x_i|y)$: **piece-wise discrete**, Gaussian, etc.

Naive Bayes model:

$$h(\mathbf{x}) = P(x_1|y)P(x_2|y)\cdots p(x_d|y)P(y)$$

with your choice of distribution families

More about Naive Bayes

find $g(\mathbf{x}) = P(x_1|y)\cdots p(x_d|y)P(y)$ by 'good estimate' of all RHS terms

for binary classification

$$\text{sign}\left(\frac{P(x_1|+1)P(x_2|+1)\cdots p(x_d|+1)P(+1)}{P(x_1|-1)P(x_2|-1)\cdots p(x_d|-1)P(-1)} - 1\right)$$

$$= \text{sign}\left(\frac{P(+1)}{P(-1)}\prod_{i=1}^{d}\frac{P(x_i|+1)}{P(x_i|-1)} - 1\right)$$

$$= \text{sign}\left(\underbrace{\log\frac{P(+1)}{P(-1)}}_{w_0} + \sum_{i=1}^{d}\underbrace{\log\frac{P(x_i|+1)}{P(x_i|-1)}}_{\phi_i(\mathbf{x})}\right) = \text{sign}\left(w_0 + \sum_{i=1}^{d} 1\cdot\phi_i(\mathbf{x})\right)$$

—also naive linear model with 'heuristic/learned' transform and bias

a simple (heuristic) model, usually **super fast**

IDCM 2006 Top 10 Data Mining Algorithms

1. C4.5: decision tree
2. K-means: clustering, taught with RBF Network
3. SVM: large-margin/kernel
4. Apriori: for frequent itemset mining
5. EM: the 'gradient descent' in Bayesian learning
6. PageRank: for link-analysis, similar to matrix factorization
7. AdaBoost: aggregation
8. k-NN: taught very shortly within RBF Network
9. Naive Bayes: linear model with heuristic transform
10. CART: decision tree

personal view of four missing ML competitors:
LinReg, LogReg, Random Forest, NNet

Fun Time

Disclaimer

Part of the following lecture borrows
Prof. Yaser S. Abu-Mostafa's slides with permission.

The prior

$P(h = f \mid \mathcal{D})$ requires an additional probability distribution:

$$P(h = f \mid \mathcal{D}) \;=\; \frac{P(\mathcal{D} \mid h = f)\, P(h = f)}{P(\mathcal{D})} \;\propto\; P(\mathcal{D} \mid h = f)\, P(h = f)$$

$P(h = f)$ is the **prior**

$P(h = f \mid \mathcal{D})$ is the **posterior**

Given the prior, we have the full distribution

Example of a prior

Consider a perceptron: h is determined by $\mathbf{w} = w_0, w_1, \cdots, w_d$

A possible prior on $\mathbf{w}$: Each w_i is independent, uniform over $[-1, 1]$

This determines the prior over h - $P(h = f)$

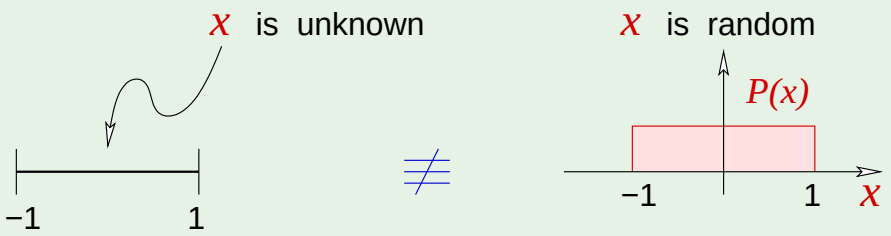
Given $\mathcal{D}$, we can compute $P(\mathcal{D} \mid h = f)$

Putting them together, we get $P(h = f \mid \mathcal{D})$

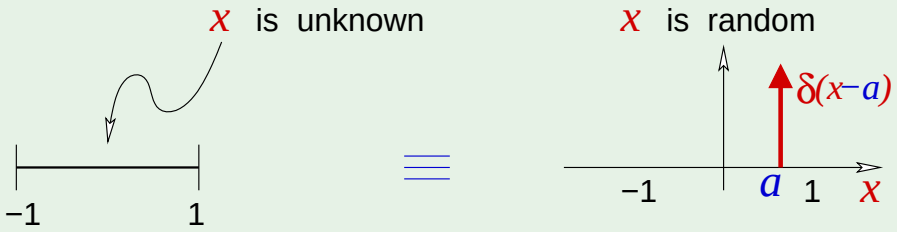
$$\propto \quad P(h = f)P(\mathcal{D} \mid h = f)$$

A prior is an assumption

Even the most "neutral" prior:



The true equivalent would be:



If we knew the prior

. . . we could compute $P(h = f \mid \mathcal{D})$ for every $h \in \mathcal{H}$

$\Longrightarrow$ we can find the most probable h given the data

we can derive $\mathbb{E}(h(\mathbf{x}))$ for every $\mathbf{x}$

we can derive the **error bar** for every $\mathbf{x}$

we can derive everything in a principled way

One Instance of Using Posterior

- logistic regression: know how to calculate likelihood $P(\mathcal{D}|\mathbf{w} = \mathbf{w}_f)$
- **define** Gaussian prior $P(\mathbf{w} = \mathbf{w}_f) = \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{I})$
- posterior = Gaussian * (logistic likelihood)
- maximize posterior
  = maximize [log Gaussian + log logistic likelihood]
  = regularized logistic regression

regularized logistic regression
= min augmented error
(with iid assumption + effective d_{VC} heuristic + surrogate error $\widehat{\mathrm{err}}$)
= max prior * likelihood
(with iid assumption + prior/likelihood assumptions)

When is Bayesian learning justified?

1. The prior is **valid**

   trumps all other methods

2. The prior is **irrelevant**

   just a computational catalyst

My Biased View

in reality:

- prior/likelihood mostly **invalid** (Gaussian, conditional independence, etc.), shooting for computational ease
- prior/likelihood **irrelevant**? **I don't know**

Fun Time

Summary

Lecture 14: Miscellaneous Models

- Matrix Factorization
- Gradient Boosted Decision Tree
- Naive Bayes
- Bayesian Learning