Lecture 16: Finale

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)
Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models
3. Distilling Implicit Features: Extraction Models

Lecture 15: Matrix Factorization

linear models of movies on extracted user features (or vice versa) jointly optimized with stochastic gradient descent

Lecture 16: Finale

- Feature Exploitation Techniques
- Error Optimization Techniques
- Overfitting Elimination Techniques
- Machine Learning in Practice

Hsuan-Tien Lin (NTU CSIE)
numerous features within some $\Phi$:
  embedded in kernel $K_\Phi$ with inner product operation
Exploiting Numerous Features via Kernel

numerous features within some $\Phi$: embedded in kernel $K_\Phi$ with inner product operation

Polynomial Kernel
‘scaled’ polynomial transforms
Exploiting Numerous Features via Kernel

Numerous features within some $\Phi$:
- Embedded in kernel $K_\Phi$ with inner product operation

### Polynomial Kernel
- ‘scaled’ polynomial transforms

### Gaussian Kernel
- Infinite-dimensional transforms

---

Hsuan-Tien Lin (NTU CSIE)
Exploiting Numerous Features via Kernel

**Numerous features within some** \( \Phi \): embedded in kernel \( K_\Phi \) with inner product operation

- **Polynomial Kernel**: ‘scaled’ polynomial transforms
- **Gaussian Kernel**: infinite-dimensional transforms
- **Stump Kernel**: decision-stumps as transforms
Exploiting Numerous Features via Kernel

**Numerous features within some $\Phi$:**
- Embedded in kernel $K_\Phi$ with inner product operation

### Kernels

<table>
<thead>
<tr>
<th>Polynomial Kernel</th>
<th>Gaussian Kernel</th>
<th>Stump Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scaled</code> polynomial transforms</td>
<td>infinite-dimensional transforms</td>
<td>decision-stumps as transforms</td>
</tr>
</tbody>
</table>

### Kernels

<table>
<thead>
<tr>
<th>Sum of Kernels</th>
<th>Transform union</th>
</tr>
</thead>
</table>
Exploiting Numerous Features via Kernel

### Feature Exploitation Techniques

- **numerous features within some** \( \Phi \):  
  - embedded in kernel \( K_\Phi \) with **inner product operation**

#### Kernels

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial Kernel</td>
<td>‘scaled’ polynomial transforms</td>
</tr>
<tr>
<td>Gaussian Kernel</td>
<td>infinite-dimensional transforms</td>
</tr>
<tr>
<td>Stump Kernel</td>
<td>decision-stumps as transforms</td>
</tr>
<tr>
<td>Sum of Kernels</td>
<td>transform <strong>union</strong></td>
</tr>
<tr>
<td>Product of Kernels</td>
<td>transform <strong>combination</strong></td>
</tr>
</tbody>
</table>

**Mercer Kernels**
- transform implicitly
- kernel ridge regression
- kernel logistic regression
- SVM
- SVR
- probabilistic SVM

**Other Techniques**
- Kernel PCA
- Kernel k-Means
- ...
### Exploiting Numerous Features via Kernel

Numerous features within some $\Phi$ are embedded in kernel $K_\Phi$ with inner product operation.

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial Kernel</td>
<td>‘scaled’ polynomial transforms</td>
</tr>
<tr>
<td>Gaussian Kernel</td>
<td>Infinite-dimensional transforms</td>
</tr>
<tr>
<td>Stump Kernel</td>
<td>Decision-stumps as transforms</td>
</tr>
<tr>
<td>Sum of Kernels</td>
<td>Transform union</td>
</tr>
<tr>
<td>Product of Kernels</td>
<td>Transform combination</td>
</tr>
<tr>
<td>Mercer Kernels</td>
<td>Transform implicitly</td>
</tr>
</tbody>
</table>

SVM, SVR, probabilistic SVM, possibly Kernel PCA, Kernel $k$-Means, ...
Exploiting Numerous Features via Kernel

### Numerous features within some $\Phi$:
- embedded in kernel $K_\Phi$ with inner product operation

<table>
<thead>
<tr>
<th>Polynomial Kernel</th>
<th>Gaussian Kernel</th>
<th>Stump Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘scaled’ polynomial transforms</td>
<td>infinite-dimensional transforms</td>
<td>decision-stumps as transforms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sum of Kernels</th>
<th>Product of Kernels</th>
<th>Mercer Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform union</td>
<td>transform combination</td>
<td>transform implicitly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SVM</th>
<th>SVR</th>
<th>probabilistic SVM</th>
</tr>
</thead>
</table>
Exploiting Numerous Features via Kernel

Numerous features within some $\Phi$:
- embedded in kernel $K_\Phi$ with inner product operation

<table>
<thead>
<tr>
<th>Polynomial Kernel</th>
<th>Gaussian Kernel</th>
<th>Stump Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘scaled’ polynomial transforms</td>
<td>infinite-dimensional transforms</td>
<td>decision-stumps as transforms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sum of Kernels</th>
<th>Product of Kernels</th>
<th>Mercer Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform union</td>
<td>transform combination</td>
<td>transform implicitly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SVM</th>
<th>SVR</th>
<th>probabilistic SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>kernel logistic regression</td>
</tr>
</tbody>
</table>
### Exploiting Numerous Features via Kernel

**Numerous features within some** $\Phi$: embedded in kernel $K_\Phi$ with **inner product operation**

- **Polynomial Kernel**: ‘scaled’ polynomial transforms
- **Gaussian Kernel**: infinite-dimensional transforms
- **Stump Kernel**: decision-stumps as transforms

- **Sum of Kernels**: transform union
- **Product of Kernels**: transform combination
- **Mercer Kernels**: transform implicitly

- **Kernel ridge regression**
- **Kernel logistic regression**
- **Probabilistic SVM**

- **SVM**
- **SVR**

**Possibly**: Kernel PCA, Kernel $k$-Means, ...
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

**Decision Stump**

*simplest* perceptron; *simplest* DecTree
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

Decision Stump

simplest perceptron; simplest DecTree

Decision Tree

branching (divide) + leaves (conquer)
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

- **Decision Stump**
  - simplest perceptron;
  - simplest DecTree

- **Decision Tree**
  - branching (divide) +
  - leaves (conquer)

- **(Gaussian) RBF**
  - prototype (center) +
  - influence
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

- **Decision Stump**: simplest perceptron; simplest DecTree
- **Decision Tree**: branching (divide) + leaves (conquer)
- **(Gaussian) RBF**: prototype (center) + influence

- **Uniform**
- **Non-Uniform**
- **Conditional**
Exploiting Predictive Features via Aggregation

**predictive features within some $\Phi$:**

$$\phi_t(x) = g_t(x)$$

<table>
<thead>
<tr>
<th>Decision Stump</th>
<th>Decision Tree</th>
<th>(Gaussian) RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>simplest perceptron; simplest DecTree</td>
<td>branching (divide) + leaves (conquer)</td>
<td>prototype (center) + influence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uniform</th>
<th>Non-Uniform</th>
<th>Conditional</th>
</tr>
</thead>
</table>

Bagging; Random Forest

Hsuan-Tien Lin (NTU CSIE)  
Machine Learning Techniques  
3/21
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

- **Decision Stump**
  - simplest perceptron;
  - simplest DecTree

- **Decision Tree**
  - branching (divide) + leaves (conquer)

- **(Gaussian) RBF**
  - prototype (center) + influence

- **Uniform**

- **Non-Uniform**

- **Conditional**

- **Bagging; Random Forest**

- **AdaBoost; GradientBoost**
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

<table>
<thead>
<tr>
<th>Decision Stump</th>
<th>Decision Tree</th>
<th>(Gaussian) RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>simplest perceptron; simplest DecTree</td>
<td>branching (divide) + leaves (conquer)</td>
<td>prototype (center) + influence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uniform</th>
<th>Non-Uniform</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging; Random Forest</td>
<td>AdaBoost; GradientBoost</td>
<td>Decision Tree;</td>
</tr>
</tbody>
</table>
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

<table>
<thead>
<tr>
<th>Decision Stump</th>
<th>Decision Tree</th>
<th>(Gaussian) RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>simplest perceptron; simplest DecTree</td>
<td>branching (divide) + leaves (conquer)</td>
<td>prototype (center) + influence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uniform</th>
<th>Non-Uniform</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging; Random Forest</td>
<td>AdaBoost; GradientBoost</td>
<td>Decision Tree; Nearest Neighbor</td>
</tr>
</tbody>
</table>
Exploiting Predictive Features via Aggregation

predictive features within some $\Phi$:
$$\phi_t(x) = g_t(x)$$

- **Decision Stump**
  - simplest perceptron; simplest DecTree

- **Decision Tree**
  - branching (divide) + leaves (conquer)

- **(Gaussian) RBF**
  - prototype (center) + influence

- **Uniform**

- **Non-Uniform**

- **Conditional**

- **Bagging; Random Forest**

- **AdaBoost; GradientBoost**

- **Decision Tree; Nearest Neighbor**

- **probabilistic SVM**
Exploiting Predictive Features via Aggregation

Predictive features within some $\Phi$:

$$\phi_t(x) = g_t(x)$$

<table>
<thead>
<tr>
<th>Decision Stump</th>
<th>Decision Tree</th>
<th>(Gaussian) RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>simplest perceptron; simplest DecTree</td>
<td>branching (divide) + leaves (conquer)</td>
<td>prototype (center) + influence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uniform</th>
<th>Non-Uniform</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging; Random Forest</td>
<td>AdaBoost; GradientBoost</td>
<td>Decision Tree; Nearest Neighbor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>probabilistic SVM</th>
</tr>
</thead>
</table>

possibly Infinite Ensemble Learning, Decision Tree SVM, ...
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights
hidden features within some \( \Phi \):

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning

Neural Network; Deep Learning
neuron weights
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning

- Neural Network; Deep Learning
  - neuron weights

- RBF Network
  - RBF centers

Hsuan-Tien Lin (NTU CSIE)
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning

**Neural Network; Deep Learning**
- neuron weights

**RBF Network**
- RBF centers

**Matrix Factorization**
- user/movie factors
hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning

Neural Network; Deep Learning
  - neuron weights

RBF Network
  - RBF centers

Matrix Factorization
  - user/movie factors

$k$-Means
  - cluster centers
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights
—possibly with the help of unsupervised learning

- Neural Network; Deep Learning
  - neuron weights
- RBF Network
  - RBF centers
- Matrix Factorization
  - user/movie factors
- $k$-Means
  - cluster centers
- Autoencoder; PCA
  - ‘basis’ directions
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights
—possibly with the help of unsupervised learning

- Neural Network; Deep Learning: neuron weights
- RBF Network: RBF centers
- Matrix Factorization: user/movie factors
- AdaBoost; GradientBoost: $g_t$ parameters
- $k$-Means: cluster centers
- Autoencoder; PCA: ‘basis’ directions
Exploiting Hidden Features via Extraction

hidden features within some $\Phi$:

as hidden variables to be ‘jointly’ optimized with usual weights

—possibly with the help of unsupervised learning

<table>
<thead>
<tr>
<th>Neural Network; Deep Learning</th>
<th>RBF Network</th>
<th>Matrix Factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>neuron weights</td>
<td>RBF centers</td>
<td>user/movie factors</td>
</tr>
<tr>
<td>AdaBoost; GradientBoost</td>
<td>$k$-Means</td>
<td>Autoencoder; PCA</td>
</tr>
<tr>
<td>$g_t$ parameters</td>
<td>cluster centers</td>
<td>‘basis’ directions</td>
</tr>
</tbody>
</table>

possibly GradientBoosted Neurons, NNet on Factorized Features, ...
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features

- Decision Stump; DecTree Branching
  - ‘best’ naïve projection to $\mathbb{R}$
- Random Forest
  - 'random' low-dim. projection
- Autoencoder; PCA
  - info.-preserving compression
- Matrix Factorization
  - projection from abstract to concrete
- Feature Selection
  - 'most-helpful' low-dimensional projection
  - possibly other 'dimension reduction' models
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

- compressed from original features

**Decision Stump; DecTree Branching**

- ‘best’ naïve projection to $\mathbb{R}$

**Random Forest Tree Branching**

- ‘random’ low-dim. projection
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features

Decision Stump; DecTree Branching
‘best’ naïve projection to $\mathbb{R}$

Random Forest Tree Branching
‘random’ low-dim. projection

Autoencoder; PCA
info.-preserving compression
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features

Decision Stump; DecTree Branching

‘best’ naïve projection to $\mathbb{R}$

Random Forest Tree Branching

‘random’ low-dim. projection

Autoencoder; PCA

info.-preserving compression

Matrix Factorization

projection from abstract to concrete
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features

Decision Stump; DecTree Branching

‘best’ naïve projection to $\mathbb{R}$

Random Forest Tree Branching

‘random’ low-dim. projection

Autoencoder; PCA

info.-preserving compression

Matrix Factorization

projection from abstract to concrete

Feature Selection

‘most-helpful’ low-dimensional projection
Exploiting Low-Dim. Features via Compression

low-dimensional features within some $\Phi$:

compressed from original features

**Decision Stump; DecTree Branching**

‘best’ naïve projection to $\mathbb{R}$

**Random Forest Tree Branching**

‘random’ low-dim. projection

**Autoencoder; PCA**

info.-preserving compression

**Matrix Factorization**

projection from abstract to concrete

**Feature Selection**

‘most-helpful’ low-dimensional projection

possibly other ‘dimension reduction’ models
Consider running AdaBoost-Stump on a PCA-preprocessed data set. Then, in terms of the original features $\mathbf{x}$, what does the final hypothesis $G(\mathbf{x})$ look like?

1. a neural network with $\tanh(\cdot)$ in the hidden neurons
2. a neural network with $\text{sign}(\cdot)$ in the hidden neurons
3. a decision tree
4. a random forest
Consider running AdaBoost-Stump on a PCA-preprocessed data set. Then, in terms of the original features $\mathbf{x}$, what does the final hypothesis $G(\mathbf{x})$ look like?

1. a neural network with $\tanh(\cdot)$ in the hidden neurons
2. a neural network with $\text{sign}(\cdot)$ in the hidden neurons
3. a decision tree
4. a random forest

Reference Answer: 2

PCA results in a linear transformation of $\mathbf{x}$. Then, when applying a decision stump on the transformed data, it is as if a perceptron is applied on the original data. So the resulting $G$ is simply a linear aggregation of perceptrons.
Numerical Optimization via Gradient Descent

when $\nabla E$ ‘approximately’ defined, use it for 1st order approximation:

$$\text{new variables} = \text{old variables} - \eta \nabla E$$
Numerical Optimization via Gradient Descent

when $\nabla E$ ‘approximately’ defined, use it for **1st order approximation**:

$$\text{new variables} = \text{old variables} - \eta \nabla E$$

**SGD/Minibatch/GD**

(Kernel) LogReg;
Neural Network [backprop];
Matrix Factorization;
Linear SVM (maybe)
Numerical Optimization via Gradient Descent

when $\nabla E$ ‘approximately’ defined, use it for **1st order approximation**:

new variables = old variables $- \eta \nabla E$

---

**SGD/Minibatch/GD**

(Kernel) LogReg;
Neural Network [backprop];
Matrix Factorization;
Linear SVM (maybe)

**Functional GD**

AdaBoost;
GradientBoost
Numerical Optimization via Gradient Descent

when $\nabla E$ ‘approximately’ defined, use it for **1st order approximation**:

\[
\text{new variables} = \text{old variables} - \eta \nabla E
\]

---

**SGD/Minibatch/GD**
- (Kernel) LogReg;
- Neural Network [backprop];
- Matrix Factorization;
- Linear SVM (maybe)

**Steepest Descent**
- AdaBoost;
- GradientBoost

**Functional GD**
- AdaBoost;
- GradientBoost
Numerical Optimization via Gradient Descent

when $\nabla E$ ‘approximately’ defined, use it for **1st order approximation**:

$$\text{new variables} = \text{old variables} - \eta \nabla E$$

**SGD/Minibatch/GD**
(Kernel) LogReg;
Neural Network [backprop];
Matrix Factorization;
Linear SVM (maybe)

**Steepest Descent**
AdaBoost;
GradientBoost

**Functional GD**
AdaBoost;
GradientBoost

possibly **2nd order techniques,**
**GD under constraints,** ...
Indirect Optimization via Equivalent Solution

when difficult to solve original problem, seek for **equivalent solution**
Indirect Optimization via Equivalent Solution

when difficult to solve original problem, seek for equivalent solution

Dual SVM

equivalence via convex QP
Indirect Optimization via Equivalent Solution

when difficult to solve original problem, seek for equivalent solution

Dual SVM
equivalence via convex QP

Kernel LogReg
Kernel RidgeReg
equivalence via representer

Hsuan-Tien Lin (NTU CSIE)
Indirect Optimization via Equivalent Solution

when difficult to solve original problem, seek for equivalent solution

Dual SVM
- equivalence via convex QP

Kernel LogReg
- Kernel RidgeReg
- equivalence via representer

PCA
- equivalence to eigenproblem
Indirect Optimization via Equivalent Solution

when difficult to solve original problem, seek for equivalent solution

- **Dual SVM**
  - equivalence via convex QP

- **Kernel LogReg**
  - Kernel RidgeReg
  - equivalence via representer

- **PCA**
  - equivalence to eigenproblem

some other boosting models and modern solvers of kernel models rely on such a technique heavily
Complicated Optimization via Multiple Steps

when difficult to solve original problem,
seek for ‘easier’ sub-problems
Complicated Optimization via Multiple Steps

when difficult to solve original problem,
seek for ‘easier’ sub-problems

Multi-Stage
probabilistic SVM;
linear blending;
stacking;
RBF Network;
DeepNet pre-training
Complicated Optimization via Multiple Steps

when difficult to solve original problem, seek for ‘easier’ sub-problems

Multi-Stage
- probabilistic SVM;
- linear blending;
- stacking;
- RBF Network;
- DeepNet pre-training

Alternating Optim.
- $k$-Means;
- alternating LeastSqr;
Complicated Optimization via Multiple Steps

when difficult to solve original problem, seek for ‘easier’ sub-problems

Multi-Stage
- probabilistic SVM;
- linear blending;
- stacking;
- RBF Network;
- DeepNet pre-training

Alternating Optim.
- $k$-Means;
- alternating LeastSqr;
  (steepest descent)
Complicated Optimization via Multiple Steps

when difficult to solve original problem, seek for ‘easier’ sub-problems

- **Multi-Stage**
  - probabilistic SVM;
  - linear blending;
  - stacking;
  - RBF Network;
  - DeepNet pre-training
- **Alternating Optim.**
  - $k$-Means;
  - alternating LeastSqr;
    - (steepest descent)
- **Divide & Conquer**
  - decision tree;
Complicated Optimization via Multiple Steps

when difficult to solve original problem, seek for ‘easier’ sub-problems

Multi-Stage
- probabilistic SVM;
- linear blending;
- stacking;
- RBF Network;
- DeepNet pre-training

Alternating Optim.
- $k$-Means;
- alternating LeastSqr;
- (steepest descent)

Divide & Conquer
- decision tree;

useful for complicated models
When running the DeepNet algorithm introduced in Lecture 213 on a PCA-preprocessed data set, which optimization technique is used?

1. variants of gradient-descent
2. locating equivalent solutions
3. multi-stage optimization
4. all of the above
When running the DeepNet algorithm introduced in Lecture 213 on a PCA-preprocessed data set, which optimization technique is used?

1. variants of gradient-descent
2. locating equivalent solutions
3. multi-stage optimization
4. all of the above

Reference Answer: 4

minibatch GD for training; equivalent eigenproblem solution for PCA; multi-stage for pre-training
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere

large-margin

SVM;
AdaBoost (indirectly)
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere

large-margin
SVM;
AdaBoost (indirectly)

L2
SVR;
kernel models;
NNet [weight-decay]
### Overfitting Elimination via Regularization

When model too ‘powerful’:

> add **brakes** somewhere

<table>
<thead>
<tr>
<th>Large-margin</th>
<th>L2</th>
<th>Voting/averaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM; AdaBoost (indirectly)</td>
<td>SVR; kernel models; NNet [weight-decay]</td>
<td>uniform blending; Bagging; Random Forest</td>
</tr>
</tbody>
</table>
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere

large-margin
SVM;
AdaBoost (indirectly)

L2
SVR;
kernel models;
NNet [weight-decay]

denoising
autoencoder

voting/averaging
uniform blending;
Bagging;
Random Forest
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere

large-margin
SVM; AdaBoost (indirectly)

L2
SVR; kernel models; NNet [weight-decay]

voting/averaging
uniform blending; Bagging; Random Forest

denoising
autoencoder

pruning
decision tree
## Overfitting Elimination via Regularization

When model too ‘powerful’:

Add **brakes** somewhere

<table>
<thead>
<tr>
<th>Large-margin</th>
<th>L2</th>
<th>Voting/averaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM; AdaBoost (indirectly)</td>
<td>SVR; kernel models; NNet [weight-decay]</td>
<td>uniform blending; Bagging; Random Forest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Denoising</th>
<th>Weight-elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoencoder</td>
<td>NNet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pruning</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree</td>
<td></td>
</tr>
</tbody>
</table>
Overfitting Elimination via Regularization

when model too ‘powerful’:

add brakes somewhere

large-margin
- SVM;
- AdaBoost (indirectly)

L2
- SVR;
- kernel models;
- NNet [weight-decay]

voting/averaging
- uniform blending;
- Bagging;
- Random Forest

denoising
- autoencoder

weight-elimination
- NNet

pruning
- decision tree

early stopping
- NNet (any GD-like)
## Overfitting Elimination via Regularization

When model too ‘powerful’:

- *add brakes somewhere*

<table>
<thead>
<tr>
<th>Large-margin</th>
<th>L2</th>
<th>Voting/averaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM; AdaBoost (indirectly)</td>
<td>SVR; kernel models; NNet [weight-decay]</td>
<td>uniform blending; Bagging; Random Forest</td>
</tr>
<tr>
<td>Denoising</td>
<td>Weight-elimination</td>
<td>Constraining</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>NNet</td>
<td>Autoenc. [weights]; RBF [# centers];</td>
</tr>
<tr>
<td>Pruning</td>
<td>Early stopping</td>
<td>NNet (any GD-like)</td>
</tr>
</tbody>
</table>
Overfitting Elimination via Regularization

**when model too ‘powerful’:**

- add **brakes** somewhere

**Large-Margin**
- SVM;
- AdaBoost (indirectly)

**L2**
- SVR;
- kernel models;
- NNet [weight-decay]

**Voting/Averaging**
- uniform blending;
- Bagging;
- Random Forest

**Denoising**
- autoencoder

**Weight-Elimination**
- NNet

**Constraining**
- autoenc. [weights];
- RBF [# centers];

**Pruning**
- decision tree

**Early Stopping**
- NNet (any GD-like)

- **arguably** most important techniques
Overfitting Elimination via Validation

when model too ‘powerful’:

check performance carefully and honestly
Overfitting Elimination via Validation

when model too ‘powerful’:

check performance carefully and honestly

OOB
Random Forest
Overfitting Elimination via Validation

when model too ‘powerful’:

check performance carefully and honestly

<table>
<thead>
<tr>
<th># SV</th>
<th>OOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM/ SVR</td>
<td>Random Forest</td>
</tr>
</tbody>
</table>
## Overfitting Elimination via Validation

When model too ‘powerful’:

Check **performance carefully and honestly**

### # SV
- SVM/SVR

### OOB
- Random Forest

### Internal Validation
- Blending;
- DecTree pruning
Overfitting Elimination via Validation

when model too ‘powerful’:

check performance carefully and honestly

- SVM/SVR
- Random Forest
- DecTree pruning

simple but necessary
What is the major technique for eliminating overfitting in Random Forest?

1. voting/averaging
2. pruning
3. early stopping
4. weight-elimination

Reference Answer:
Random Forest, based on uniform blending, relies on voting/averaging for regularization.
Hsuan-Tien Lin (NTU CSIE)
What is the major technique for eliminating overfitting in Random Forest?

1. voting/averaging
2. pruning
3. early stopping
4. weight-elimination

Reference Answer: 1

Random Forest, based on uniform blending, relies on voting/averaging for regularization.
NTU KDDCup 2010 World Champion Model

Feature engineering and classifier ensemble for KDD Cup 2010, Yu et al., KDDCup 2010

linear blending of
NTU KDDCup 2010 World Champion Model

Feature engineering and classifier ensemble for KDD Cup 2010, Yu et al., KDDCup 2010

linear blending of

Logistic Regression +
many rawly encoded features
NTU KDDCup 2010 World Champion Model

Feature engineering and classifier ensemble for KDD Cup 2010, Yu et al., KDDCup 2010

linear blending of

Logistic Regression +
many rawly encoded features

Random Forest +
human-designed features
NTU KDDCup 2010 World Champion Model

Feature engineering and classifier ensemble for KDD Cup 2010, Yu et al., KDDCup 2010

linear blending of

- Logistic Regression + many rawly encoded features
- Random Forest + human-designed features

yes, you’ve learned everything! :-)

Hsuan-Tien Lin (NTU CSIE)
NTU KDDCup 2011 Track 1 World Champion Model

A linear ensemble of individual and blended models for music rating prediction,
Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

• Matrix Factorization variants, including probabilistic PCA
• Restricted Boltzmann Machines: an 'extended' autoencoder
• $k$ Nearest Neighbors
• Probabilistic Latent Semantic Analysis: an extraction model that has 'soft clusters' as hidden variables
• linear regression, NNet, & GBDT

yes, you can 'easily' understand everything! :-)}
A linear ensemble of individual and blended models for music rating prediction, Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- Matrix Factorization variants, including probabilistic PCA
NTU KDDCup 2011 Track 1 World Champion Model

A linear ensemble of individual and blended models for music rating prediction, Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- Matrix Factorization variants, including probabilistic PCA
- Restricted Boltzmann Machines: an ‘extended’ autoencoder
NTU KDDCup 2011 Track 1 World Champion Model

A linear ensemble of individual and blended models for music rating prediction,
Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- Matrix Factorization variants, including probabilistic PCA
- Restricted Boltzmann Machines: an ‘extended’ autoencoder
- \(k\) Nearest Neighbors
NTU KDDCup 2011 Track 1 World Champion Model

A linear ensemble of individual and blended models for music rating prediction, Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- **Matrix Factorization** variants, including probabilistic PCA
- **Restricted Boltzmann Machines**: an ‘extended’ autoencoder
- **k Nearest Neighbors**
- **Probabilistic Latent Semantic Analysis**: an extraction model that has ‘soft clusters’ as hidden variables
A linear ensemble of individual and blended models for music rating prediction,

Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- Matrix Factorization variants, including probabilistic PCA
- Restricted Boltzmann Machines: an ‘extended’ autoencoder
- k Nearest Neighbors
- Probabilistic Latent Semantic Analysis:
  an extraction model that has ‘soft clusters’ as hidden variables
- linear regression, NNet, & GBDT
NTU KDDCup 2011 Track 1 World Champion Model

A linear ensemble of individual and blended models for music rating prediction,
Chen et al., KDDCup 2011

NNet, DecTree-like, and then linear blending of

- Matrix Factorization variants, including probabilistic PCA
- Restricted Boltzmann Machines: an ‘extended’ autoencoder
- \( k \) Nearest Neighbors
- Probabilistic Latent Semantic Analysis: an extraction model that has ‘soft clusters’ as hidden variables
- linear regression, NNet, & GBDT

yes, you can ‘easily’ understand everything! :-}
NTU KDD Cup 2012 Track 2 World Champion Model

A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012, Wu et al., KDD Cup 2012

NNet, GBDT-like, and then linear blending of
A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012, Wu et al., KDDCup 2012

NNet, GBDT-like, and then linear blending of

- Linear Regression variants, including linear SVR
A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012, Wu et al., KDDCup 2012

NNet, GBDT-like, and then linear blending of
- Linear Regression variants, including linear SVR
- Logistic Regression variants
A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012,

NNet, GBDT-like, and then linear blending of

- Linear Regression variants, including linear SVR
- Logistic Regression variants
- Matrix Factorization variants
- ...
NTU KDDCup 2012 Track 2 World Champion Model

A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012, Wu et al., KDDCup 2012

NNet, GBDT-like, and then linear blending of

- Linear Regression variants, including linear SVR
- Logistic Regression variants
- Matrix Factorization variants
- ...

‘key’ is to blend properly without overfitting
NTU KDDCup 2013 Track 1 World Champion Model

Combination of feature engineering and ranking models for paper-author identification in KDD Cup 2013, Li et al., KDDCup 2013

linear blending of
NTU KDD Cup 2013 Track 1 World Champion Model

Combination of feature engineering and ranking models for paper-author identification in KDD Cup 2013, Li et al., KDD Cup 2013

linear blending of

- Random Forest with many many many trees
NTU KDD Cup 2013 Track 1 World Champion Model

Combination of feature engineering and ranking models for paper-author identification in KDD Cup 2013, Li et al., KDD Cup 2013

**Linear blending of**

- Random Forest with many many many many trees
- GBDT variants
NTU KDDCup 2013 Track 1 World Champion Model

Combination of feature engineering and ranking models for paper-author identification in KDD Cup 2013, Li et al., KDDCup 2013

linear blending of

- Random Forest with many many many trees
- GBDT variants

with tons of efforts in designing features
NTU KDDCup 2013 Track 1 World Champion Model

Combination of feature engineering and ranking models for paper-author identification in KDD Cup 2013, Li et al., KDDCup 2013

linear blending of
- Random Forest with many many many trees
- GBDT variants

with tons of efforts in designing features

‘another key’ is to construct features with domain knowledge
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another **decision tree**
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. k-Means
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. k-Means
3. SVM
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. $k$-Means
3. SVM
4. Apriori: for frequent itemset mining
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. k-Means
3. SVM
4. Apriori: for frequent itemset mining
5. EM: ‘alternating optimization’ algorithm for some models
ICDM 2006 Top 10 Data Mining Algorithms

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C4.5: another <strong>decision tree</strong></td>
</tr>
<tr>
<td>2</td>
<td><em>k</em>-Means</td>
</tr>
<tr>
<td>3</td>
<td>SVM</td>
</tr>
<tr>
<td>4</td>
<td>Apriori: for frequent itemset mining</td>
</tr>
<tr>
<td>5</td>
<td>EM: ‘<strong>alternating optimization</strong>’ algorithm for some models</td>
</tr>
<tr>
<td>6</td>
<td>PageRank: for link-analysis, similar to <strong>matrix factorization</strong></td>
</tr>
</tbody>
</table>
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another **decision tree**
2. $k$-Means
3. SVM
4. Apriori: for frequent itemset mining
5. **EM:** ‘alternating optimization’ algorithm for some models
6. PageRank: for link-analysis, similar to **matrix factorization**
7. AdaBoost
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another **decision tree**
2. *k*-Means
3. SVM
4. Apriori: for frequent itemset mining
5. EM: ‘**alternating optimization**’ algorithm for some models
6. PageRank: for link-analysis, similar to **matrix factorization**
7. AdaBoost
8. *k* Nearest Neighbor
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. k-Means
3. SVM
4. Apriori: for frequent itemset mining
5. EM: ‘alternating optimization’ algorithm for some models
6. PageRank: for link-analysis, similar to matrix factorization
7. AdaBoost
8. k Nearest Neighbor
9. Naive Bayes: a simple linear model with ‘weights’ decided by data statistics

Hsuan-Tien Lin (NTU CSIE)
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another **decision tree**
2. k-Means
3. SVM
4. Apriori: for frequent itemset mining
5. EM: ‘**alternating optimization**’ algorithm for some models
6. PageRank: for link-analysis, similar to **matrix factorization**
7. AdaBoost
8. k Nearest Neighbor
9. Naive Bayes: a simple **linear model** with ‘weights’ decided by data statistics
10. C&RT
ICDM 2006 Top 10 Data Mining Algorithms

1. C4.5: another decision tree
2. k-Means
3. SVM
4. Apriori: for frequent itemset mining
5. EM: ‘alternating optimization’ algorithm for some models
6. PageRank: for link-analysis, similar to matrix factorization
7. AdaBoost
8. k Nearest Neighbor
9. Naive Bayes: a simple linear model with ‘weights’ decided by data statistics
10. C&RT

personal view of five missing ML competitors:
   LinReg, LogReg, Random Forest, GBDT, NNet
Machine Learning Jungle

welcome to the jungle!
Fun Time

Which of the following is the official lucky number of this class?

1. 9876
2. 1234
3. 1126
4. 6211

Reference Answer: 3
Fun Time

Which of the following is the official lucky number of this class?

1. 9876
2. 1234
3. 1126
4. 6211

Reference Answer: 3

May the luckiness always be with you!
Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models
3. Distilling Implicit Features: Extraction Models

Lecture 16: Finale

- Feature Exploitation Techniques
  kernel, aggregation, extraction, low-dimensional
- Error Optimization Techniques
  gradient, equivalence, stages
- Overfitting Elimination Techniques
  (lots of) regularization, validation
- Machine Learning in Practice
  welcome to the jungle

- next: happy learning!