COMP5211: Machine Learning
Lecture 5

Minhao Cheng
Support Vector Machine
Linear SVM

- Given training examples $(x_1, y_1), \ldots, (x_n, y_n)$
- Consider binary classification:
  $y_i \in \{+1, -1\}$
- Linear Support Vector Machine (SVM):
  \[
  \arg \min_w C \sum_{i=1}^{n} \max(1 - y_i w^T x_i, 0) + \frac{1}{2} w^T w
  \]
  - (Hinge loss with l2 regularization)
Support Vector Machine

Linear SVM

• Goal: Find a hyperplane to separate these two classes of data: if $y_i = 1, w^T x_i \geq 1$; if $y_i = -1, w^T x_i \geq -1$

• Prefer a hyperplane with maximum margin
Support Vector Machine

Margin

• Intuition
  • Confidence of A, B, C
• Let our decision function as
  • \( h_{w,b} = g(w^T x + b) \)
  • \( g(z) = 1 \) if \( z \geq 0 \), \( g(z) = -1 \) otherwise
  • \( \gamma^{(i)} = y^{(i)}(w^T x + b) \)
• However, it will double just replace \( w \) with \( 2w \), \( b \) with \( 2b \)
Support Vector Machine

Margin

• Intuition
  • Confidence of A, B, C
• Let our decision function as
  • \( h_{w,b} = g(w^T x + b) \)
  • \( g(z) = 1 \) if \( z \geq 0 \), \( g(z) = -1 \) otherwise
• \( \hat{y}^{(i)} = y^{(i)}(w^T x + b) \)
• However, it will double just replace \( w \) with \( 2w \), \( b \) with \( 2b \)
Support Vector Machine
Margin

- \( w^T(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|}) + b = 0 \)
- \( \gamma^{(i)} = \frac{w^T(x^{(i)} + b)}{\|w\|} \)
- \( \gamma = \min_{i=1,\ldots,m} \gamma^{(i)} \)
Support Vector Machine

Margin

\[ \gamma^{(i)} = \frac{w^T(x^{(i)} + b)}{\|w\|} \]

\[ \gamma = \min_{i=1,\ldots,m} \gamma^{(i)} \]

\[ \max_{\gamma,w,b} \frac{\gamma}{\|w\|} \]

\[ \text{s.t. } y_i(w^Tx_i + b) \geq \gamma, i = 1,\ldots,n \]
Support Vector Machine
Linear SVM

• Take $\gamma \frac{\|w\|}{\|w\|} = \frac{1}{\|w\|}$
Support Vector Machine
Linear SVM (hard constraint)

• SVM primal problem (with hard constraints):

\[
\begin{align*}
\min_\mathbf{w} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{s.t.} & \quad y_i (\mathbf{w}^T \mathbf{x}_i) \geq 1, i = 1, \ldots, n
\end{align*}
\]
Support Vector Machine
Linear SVM (hard constraint)

- SVM primal problem (with hard constraints):
  \[
  \min_w \frac{1}{2} w^T w \\
  \text{s.t. } y_i(w^T x_i) \geq 1, i = 1, \ldots, n
  \]
- What if there are outliers?
Support Vector Machine
Linear SVM

• Given training examples \((x_1, y_1), \ldots, (x_n, y_n)\)

• Consider binary classification:
  \(y_i \in \{+1, -1\}\)

• Linear Support Vector Machine (SVM):

\[
\begin{align*}
\min_w & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} & \quad y_i(w^T x_i) \geq 1 - \xi_i, \ i = 1, \ldots, n \\
& \quad \xi_i \geq 0
\end{align*}
\]
Support Vector Machine
Linear SVM

- SVM primal problem:
  \[
  \min_w \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
  \text{s.t. } y_i(w^T x_i) \geq 1 - \xi_i, i = 1, \ldots, n \\
  \xi_i \geq 0
  \]

- Equivalent to
  \[
  \arg \min_w C \sum_{i=1}^{n} \max(1 - y_i w^T x_i, 0) + \frac{1}{2} w^T w \\
  \text{hinge loss} \quad \text{L2 regularization}
  \]

- None-differentiable when \( y_i w^T x_i = 1 \) for some \( i \)
Support Vector Machine
Stochastic subgradient method for SVM

• A sub gradient of $\ell_i(w) = \max(0, 1 - y_i w^T x_i)$:

$$
\nabla_w \ell_i(w) = \begin{cases} 
-y_i x_i, & \text{if } 1 - y_i w^T x_i > 0 \\
0, & \text{if } 1 - y_i w^T x_i < 0 \\
0, & \text{if } 1 - y_i w^T x_i = 0 
\end{cases}
$$

• Stochastic subgradient descent for SVM:

For $t = 1, 2, \ldots$

Randomly pick an index $i$

If $y_i w^T x_i < 1$, then

$$
w \leftarrow (1 - \eta_t) w + \eta_t n C y_i x_i
$$

Else (if $y_i w^T x_i \geq 1$):

$$
w \leftarrow (1 - \eta_t) w
$$
Support Vector Machine

Linear SVM

- SVM primal problem:

\[
\begin{align*}
\min_w & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} & \quad y_i (w^T x_i) \geq 1 - \xi_i, \ i = 1, \ldots, n \\
\end{align*}
\]

- Equivalent to

\[
\begin{align*}
\begin{array}{c}
\text{arg min}_w \\
\end{array} \\
C \sum_{i=1}^{n} \max(1 - y_i w^T x_i, 0) + \frac{1}{2} w^T w
\end{align*}
\]

- None-differentiable when \(y_i w^T x_i = 1\) for some \(i\)

- Alternatively, we show how to derive the dual form of SVM
Support Vector Machine
Linear SVM dual

- SVM primal problem:
  \[
  \min_w \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i
  \]
  \[
  \text{s.t. } y_i(w^T x_i) \geq 1 - \xi_i, i = 1, \ldots, n
  \]
  \[
  \xi_i \geq 0
  \]
- Equivalent to: (using Lagrange multiplier):
  \[
  \min_{w, \xi} \max_{\alpha \geq 0, \beta \geq 0} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_i \beta_i \xi_i
  \]
Support Vector Machine

Linear SVM dual form

- SVM primal problem:
  \[
  \min_w \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
  \text{s.t.} \quad y_i (w^T x_i) \geq 1 - \xi_i, \quad i = 1, \ldots, n \]

  \[\xi_i \geq 0\]

- Equivalent to (using Lagrange multiplier):
  \[
  \min_{w, \xi} \max_{\alpha \geq 0, \beta \geq 0} \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_i \beta_i \xi_i
  \]

- Under certain condition (e.g., Slater’s condition), exchanging \(\min, \max\) will not change the optimal solution:
  \[
  \max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i w^T x_i - 1 + \xi_i) - \sum_i \beta_i \xi_i
  \]
Support Vector Machine

Linear SVM dual form

- Reorganize the equation:
  \[
  \max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} \frac{1}{2} \|w\|^2 - \sum_i \alpha_i y_i w^T x_i + \sum_i \xi_i (C - \alpha_i - \beta_i) + \sum_i \alpha_i
  \]

- Now, for any given \( \alpha, \beta \), the minimizer of \( w \) will satisfy
  \[
  \frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0 \Rightarrow w^* = \sum_i y_i \alpha_i x_i
  \]

  - Also, we have \( C = \alpha_i + \beta_i \), otherwise \( \xi_i \) can make the objective function \(-\infty\)

- Substitute these two equations back we get
  \[
  \max_{\alpha \geq 0, \beta \geq 0, C = \alpha + \beta} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i
  \]
Support Vector Machine

Linear SVM dual

• Therefore, we get the following dual problem

\[
\max_{C \geq \alpha \geq 0} \left\{ -\frac{1}{2} \alpha^T Q \alpha + e^T \alpha \right\} := D(\alpha),
\]

• Where \( Q \) is an \( n \times n \) matrix with \( Q_{ij} = y_i y_j x_i^T x_j \)

• Based on the derivations, we know
  • Primal minimum = dual maximum (under Slater’s condition)
  • Let \( \alpha^* \) be the dual solution and \( w^* \) be the primal solution, we have

\[
 w^* = \sum_i y_i \alpha_i^* x_i
\]

• We can solve the dual problem instead of primal problem
Nonlinear transformation

Linear hypotheses

- Up to now: linear hypotheses
  - Perception, Linear regression, Logistic regression, …
- Many problems are not linearly separable
Nonlinear transformation
Circular Separable

- Data is not linear separable
Nonlinear transformation
Circular Separable

- Data is not linear separable
- But circular separable by a circle of radius $\sqrt{0.6}$ centered at origin:
  - $h_{SEP}(x) = \text{sign}(-x_1^2 - x_2^2 + 0.6)$
Nonlinear transformation
Circular Separable and Linear Separable

- \[ h(x) = \text{sign}(0.6 \cdot 1 + (-1) \cdot x_1^2 + (-1) \cdot x_2^2) \]
Nonlinear transformation
Circular Separable and Linear Separable

\[ h(x) = \text{sign}(0.6 \cdot \tilde{w}_0 \cdot 1 + (-1) \cdot \tilde{x}_1^2 + (-1) \cdot \tilde{x}_2^2) \]

\[ = \text{sign}(\tilde{w}^T \tilde{z}) \]
**Nonlinear transformation**

Circular Separable and Linear Separable

\[
h(x) = \text{sign}(0.6 \cdot \tilde{w}_0 + (-1) \cdot \tilde{w}_1 + (-1) \cdot \tilde{w}_2)
\]

\[
= \text{sign}(\tilde{w}^T \tilde{z})
\]

- \{ (x_n, y_n) \} circular separable \Rightarrow \{ (z_n, y_n) \} linear separable
Nonlinear transformation
Circular Separable and Linear Separable

\[ h(x) = \text{sign}(0.6 \cdot 1 + (-1) \cdot x_1^2 + (-1) \cdot x_2^2) \]

\[ = \text{sign}(\tilde{w}^T \tilde{z}) \]

\[ \{(x_n, y_n)\} \text{ circular separable} \Rightarrow \{(z_n, y_n)\} \text{ linear separable} \]

\[ x \in \mathcal{X} \rightarrow x \in \mathcal{Z} \text{ (using a nonlinear transformation } \phi) \]
Nonlinear Transformation

Definition

• Define nonlinear transformation

  • \( \phi(x) = (1, x_1^2, x_2^2) = (z_0, z_1, z_2) = z \)
Nonlinear Transformation

Definition

• Define nonlinear transformation
  
  \[ \phi(x) = (1, x_1^2, x_2^2) = (z_0, z_1, z_2) = z \]

• Linear hypotheses in \( \mathcal{F} \) space:
  
  \[ \text{sign}(\tilde{h}(z)) = \text{sign}(\tilde{h}(\phi(x))) = \text{sign}(w^T \phi(x)) \]
Nonlinear Transformation

Definition

• Define nonlinear transformation
  
  \[ \phi(x) = (1, x_1^2, x_2^2) = (z_0, z_1, z_2) = z \]

• Linear hypotheses in \( \mathcal{F} \)-space:
  
  \[ \text{sign}(\tilde{h}(z)) = \text{sign}(\tilde{h}(\phi(x))) = \text{sign}(w^T \phi(x)) \]

• Line in \( \mathcal{F} \)-space \( \Leftrightarrow \) some quadratic curves in \( \mathcal{X} \)-space
Nonlinear Transformation
General Quadratic Hypothesis Set

• A “bigger” $\mathcal{F}$ space:

• $\phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$
Nonlinear Transformation
General Quadratic Hypothesis Set

• A “bigger” $\mathcal{E}$-space:
  
  - $\phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

• Linear in $\mathcal{E}$-space $\Leftrightarrow$ quadratic hypotheses in $\mathcal{X}$-space
Nonlinear Transformation
General Quadratic Hypothesis Set

• A “bigger” $\mathcal{E}$-space:

  • $\phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

• Linear in $\mathcal{E}$-space $\iff$ quadratic hypotheses in $\mathcal{X}$-space

• The hypotheses space:

  • $\mathcal{H}_{\phi_2} = \{ h(x) : h(x) = \tilde{w}^T \phi_2(x) \text{ for some } \tilde{w} \}$ (quadratic hypotheses)
Nonlinear Transformation
General Quadratic Hypothesis Set

- A “bigger” $\mathcal{E}$-space:
  - $\phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$
- Linear in $\mathcal{E}$-space $\Leftrightarrow$ quadratic hypotheses in $\mathcal{X}$-space
- The hypotheses space:
  - $\mathcal{H}_\phi = \{ h(x) : h(x) = \tilde{w}^T \phi_2(x) \text{ for some } \tilde{w} \}$ (quadratic hypotheses)
- Also include linear model as a degenerate case
Nonlinear transformation
Learning a good quadratic function

- Transform original data \( \{x_n, y_n\} \) to \( \{z_n = \phi(x_n), y_n\} \)
- Solve a linear problem on \( \{z_n, y_n\} \) using your favorite algorithm \( \mathcal{A} \) to get a good model \( \tilde{w} \)
- Return the model \( h(x) = \text{sign}(\tilde{w}^T \phi(x)) \)
Nonlinear transformation
Polynomial mappings

- Can now freely do quadratic classification, quadratic regression
Nonlinear transformation
Polynomial mappings

• Can now freely do quadratic classification, quadratic regression
• Can easily extend to any degree of polynomial mappings
  • E.g.,
    \[ \phi(x) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2^2, x_1x_3^2, x_1x_2^2, x_2x_3^2, x_2x_3^2, x_1^3, x_2^3, x_3^3) \]
Support Vector Machine
SVM with nonlinear mapping

- SVM with nonlinear mapping $\varphi(\cdot)$:

$$
\begin{align*}
\min_w & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} & \quad y_i(w^T \varphi(x_i)) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\
& \quad \xi_i \geq 0
\end{align*}
$$

- Hard to solve if $\varphi(\cdot)$ maps to very high or infinite dimensional space
Support Vector Machine
SVM with nonlinear mapping

• Similarly, we could derive
  
  \[- \frac{1}{2} \alpha^T Q \alpha + e^T \alpha \] := D(\alpha),

• Where \( Q \) is an \( n \times n \) matrix with \( Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j) \)

• Based on the derivations, we know
  
  • Primal minimum = dual maximum (under Slater’s condition)
  • Let \( \alpha^* \) be the dual solution and \( w^* \) be the primal solution, we have
    
    \[ w^* = \sum_i y_i \alpha_i^* \phi(x_i) \]
Nonlinear Transformation

The price we pay: computational complexity

- $Q$-th order polynomial transform:

$$\phi(x) = (1, x_1, x_2, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_d^2, \ldots, x_d^2, \ldots, x_1^Q, x_1^{0-1} x_2, \ldots, x_d^Q)$$

- $O(d^Q)$ dimensional vector $\Rightarrow$ High computational cost
Support Vector Machine

Kernel trick

- Do not directly define $\varphi(\cdot)$
- Instead, define “kernel”
  - $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$
Support Vector Machine
Kernel trick

• Do not directly define \( \phi(\cdot) \)
• Instead, define “kernel”

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]

• Examples:
  - Gaussian kernel: \( K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \)
  - Polynomial kernel: \( K(x_i, x_j) = (\gamma x_i^T x_j + c)^d \)
• Other kernels for specific problems:
  - Graph kernels (Vishwanathan et al., “Graph Kernels”, JMLR, 2010)
  - Pyramid kernel for image matching (Grauman and Darrell, “The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features”. In ICCV, 2005)
  - String kernel (Lodhi et al., “Text classification using string kernels”. JMLR, 2002)
Support Vector Machine
SVM with kernel

- Training: compute $\alpha = [\alpha_1, \ldots, \alpha_n]$ by solving the quadratic optimization problem:

$$\min_{0 \leq \alpha \leq C} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

- Where $Q_{ij} = K(x_i, x_j)$
Support Vector Machine
SVM with kernel

• Training: compute \( \alpha = [\alpha_1, \ldots, \alpha_n] \) by solving the quadratic optimization problem:

\[
\min_{0 \leq \alpha \leq C} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha
\]

• Where \( Q_{ij} = K(x_i, x_j) \)

• Prediction: for a test data \( x \),

\[
 w^T \varphi(x) = \sum_{i=1}^{n} y_i \alpha_i \varphi(x_i)^T \varphi(x) = \sum_{i=1}^{n} y_i \alpha_i K(x_i, x)
\]
Kernel trick
Kernel Ridge Regression

- Actually, this “kernel method” works for many different losses
- Example: ridge regression

\[
\min_w \frac{1}{2} \|w\|^2 + \frac{1}{2} \sum_{i=1}^{n} (w^T \phi(x_i) - y_i)^2
\]

- Dual problem:

\[
\min_{\alpha} \alpha^T Q \alpha + \|\alpha\|^2 - 2\alpha^Ty
\]
Kernel trick

Scalability

- Challenge for solving kernel SVMs (for dataset with $n$ samples):
  - **Space:** $O(n^2)$ for storing the $n \times n$ kernel matrix (can be reduced in some cases);
  - **Time:** $O(n^3)$ for computing the exact solution
Kernel trick

Scalability

• Challenge for solving kernel SVMs (for dataset with $n$ samples):
  
  • **Space:** $O(n^2)$ for storing the $n \times n$ kernel matrix (can be reduced in some cases);
  
  • **Time:** $O(n^3)$ for computing the exact solution

• Good packages available:
  
  • LIBSVM (can be called in scikit-learn)
  
  • LIBLINEAR (for linear SVM, can be called in scikit-learn)
Multi problem
Multi-class

- \( n \) data points, \( L \) labels, \( d \) features
- Input: training data \( \{x_i, y_i\}_{i=1}^n \):
  - Each \( x_i \) is a \( d \) dimensional feature vector
  - Each \( y_i \in \{1, \ldots, L\} \) is the corresponding label
  - Each training data belongs to one category
- Goal: find a function to predict the correct label
  - \( f(x) \approx y \)
Multi problems

Multi-label

• $n$ data points, $L$ labels, $d$ features

• Input: training data $\{x_i, y_i\}_{i=1}^n$:
  • Each $x_i$ is a $d$ dimensional feature vector
  • Each $y_i \in \{0,1\}^L$ is a label vector (or $Y_i \in \{1,2,\ldots,L\}$)
    • Example: $y_i = [0,0,1,0,0,1,1]$ (or $y_i = \{3,6,7\}$)
  • Each training data belongs to multiple categories

• Goal: Given a testing sample $x$, predict the correct label

<table>
<thead>
<tr>
<th>Document 1</th>
<th>{Sports, Politics}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 2</td>
<td>{Science, Politics}</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Document n</td>
<td>{Environment}</td>
</tr>
</tbody>
</table>
## Multi problems

### Multi-class vs Multi-label

- Multi-class: each row of $L$ has exact one “1”
- Multi-label: each row of $L$ can have multiple ones
Multi problems
Reduction to binary classification

• Many algorithms for binary classification

• Idea: transform multi-class or multi-label problems to multiple binary classification problems

• Two approaches:
  • One versus All (OVA)
  • One versus One (OVO)
Multi problems
One Versus All (OVA)

- Multi-class/multi-label problems with $L$ categories
- Build $L$ different binary classifiers
- For the $t$-th classifier:
  - Positive samples: all the points in class $t$ ($\{x_i : t \in y_i\}$)
  - Negative samples: all the points not in class $t$ ($\{x_i : t \notin y_i\}$)
  - $f_t(x)$: the decision value for the $t$-th classifier
    - (Larger $f_t \Rightarrow$ higher probability that $x$ in class $t$)
- Prediction:
  - $f(x) = \arg\max_t f_t(x)$
- Example: using SVM to train each binary classifier
Multi problems
One Versus One (OVO)

- Multi-class/multi-label problems with $L$ categories
- Build $L(L-1)$ different binary classifiers
- For the $(s, t)$-th classifier:
  - Positive samples: all the points in class $s$ ($\{x_i : s \in y_i\}$)
  - Negative samples: all the points in class $t$ ($\{x_i : t \notin y_i\}$)
  - $f_{s,t}(x)$: the decision value for the $t$-th classifier
    - (Larger $f_{s,t}(x)$ ⇒ label $s$ has higher probability than label $t$)
    - $f_{t,s}(x) = -f_{s,t}(x)$
- Prediction:
  - $f(x) = \arg \max_s (\sum_t f_{s,t}(x))$
- Example: using SVM to train each binary classifier
Multi problems
OVA vs OVO

• Prediction accuracy: depends on datasets

• Computational time:
  • OVA needs to train $L$ classifiers
  • OVO needs to train $\frac{L(L - 1)}{2}$ classifiers

• Is OVA always faster than OVO?
Multi problems
OVA vs OVO

• Prediction accuracy: depends on datasets
• Computational time:
  • OVA needs to train $L$ classifiers
  • OVO needs to train $L(L - 1)/2$ classifiers
• Is OVA always faster than OVO?
  • No, depends on the time complexity of the binary classifier
  • If the binary classifier requires $O(n)$ time for $n$ samples:
    • OVA and OVO have similar time complexity
  • If the binary classifier requires $O(n^{1.99})$ time:
    • OVO is faster than OVA
Multi problems
OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:
  - OVA needs to train $L$ classifiers
  - OVO needs to train $L(L - 1)/2$ classifiers
- Is OVA always faster than OVO?
  - No, depends on the time complexity of the binary classifier
    - If the binary classifier requires $O(n)$ time for $n$ samples:
      - OVA and OVO have similar time complexity
    - If the binary classifier requires $O(n^{1.5})$ time:
      - OVO is faster than OVA
- LIBSVM (kernel SVM solver): OVO
- LIBLINEAR (Linear SVM solver): OVA
Multi problems
Another approach for multi-class classification

• OVA and OVO: decompose the problem by labels
  • But good binary classifiers may not imply good multi-class prediction
• Design a multi-class loss function and solve a single optimization problem
Multi problems
Another approach for multi-class classification

• OVA and OVO: decompose the problem by labels
  • But good binary classifiers may not imply good multi-class prediction
• Design a multi-class loss function and solve a single optimization problem
• Minimize the regularized training error:

$$\min_{w_1, \ldots, w_L} \sum_{i=1}^{n} \text{loss}(x_i, y_i) + \lambda \sum_{j=1, \ldots, L} w_j^T w_j$$
Multi problems

Loss functions for multi-class classification

- Ranking based approaches: directly minimizes the ranking loss:
  - For multi-class classification, the score of $y_i$ should be larger than other labels

- Soft-max loss:
  - Measure the probability of predicting correct class