Multiclass classification: Local and Global Views

Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign

Augmented and modified by Vivek Srikumar
Multiclass classification

• Introduction

• Combining binary classifiers
  – One-vs-all
  – All-vs-all
  – Error correcting codes

• Training a single classifier
  – Multiclass SVM
  – Constraint classification
What is multiclass classification?

- An input can belong to one of K classes
- Training data: Input associated with class label (a number from 1 to K)
- Prediction: Given a new input, predict the class label

Each input belongs to exactly one class. Not more, not less.
- Otherwise, the problem is not multiclass classification
- If an input can be assigned multiple labels (think tags for emails rather than folders), it is called multi-label classification
Example applications: Images

- **Input**: hand-written character; **Output**: which character? 
  
  ![Handwritten characters](image)
  
  all map to the letter A

- **Input**: a photograph of an object; **Output**: which of a set of categories of objects is it?
  
  • Eg: the Caltech 256 dataset

![Car tire](image)

![Car tire](image)

![Duck](image)

![laptop](image)
Example applications: Language

• *Input*: a news article; *Output*: which section of the newspaper should it belong to?

• *Input*: an email; *Output*: which folder should an email be placed into?

• *Input*: an audio command given to a car; *Output*: which of a set of actions should be executed?
Where are we?

• Introduction

• Combining binary classifiers
  – One-vs-all
  – All-vs-all
  – Error correcting codes

• Training a single classifier
  – Multiclass SVM
  – Constraint classification
Binary to multiclass

• Can we use a binary classifier to construct a multiclass classifier?
  – Decompose the prediction into multiple binary decisions

• How to decompose?
  – One-vs-all
  – All-vs-all
  – Error correcting codes
General setting

- **Input** $\mathbf{x} \in \mathbb{R}^n$
  - The inputs are represented by their feature vectors
- **Output** $\mathbf{y} \in \{1, 2, \ldots, K\}$
  - These classes represent domain-specific labels

- **Learning**: Given a dataset $D = \{<\mathbf{x}_i, \mathbf{y}_i>\}$
  - Need to specify a learning algorithm that takes uses $D$ to construct a function that can predict $\mathbf{y}$ given $\mathbf{x}$
  - Goal: find a predictor that does well on the training data and has low generalization error

- **Prediction/Inference**: Given an example $\mathbf{x}$ and the learned function (often also called the model)
  - Using the learned function, compute the class label for $\mathbf{x}$
1. One-vs-all classification

- **Assumption**: Each class individually separable from *all* the others

- **Learning**: Given a dataset \( D = \{<x_i, y_i>\} \),
  - Note: \( x_i \in \mathbb{R}^n \), \( y_i \in \{1, 2, \ldots, K\} \)
  - Decompose into \( K \) binary classification tasks
  - For class \( k \), construct a binary classification task as:
    - Positive examples: Elements of \( D \) with label \( k \)
    - Negative examples: All other elements of \( D \)
  - Train \( K \) binary classifiers \( w_1, w_2, \ldots, w_K \) using any learning algorithm we have seen

- **Prediction**: “*Winner Takes All*”
  \[
  \arg\max_i w_i^T x
  \]

Question: What is the dimensionality of each \( w_i \)?
Visualizing One-vs-all

From the full dataset, construct three binary classifiers, one for each class:

\[ w_{\text{blue}}^T x > 0 \]
for blue inputs

\[ w_{\text{red}}^T x > 0 \]
for red inputs

\[ w_{\text{green}}^T x > 0 \]
for green inputs

Winner Take All will predict the right answer. Only the correct label will have a positive score.

Notation: Score for blue label
One-vs-all may not always work

Black points are not separable with a single binary classifier

*The decomposition will not work for these cases!*

\[
\begin{align*}
\mathbf{w}_{\text{blue}}^\top \mathbf{x} & > 0 \\
& \text{for blue inputs} \\
\mathbf{w}_{\text{red}}^\top \mathbf{x} & > 0 \\
& \text{for red inputs} \\
\mathbf{w}_{\text{green}}^\top \mathbf{x} & > 0 \\
& \text{for green inputs} \\
\mathbf{w} & \text{???} \\
\end{align*}
\]
One-vs-all classification: Summary

• Easy to learn
  – Use any binary classifier learning algorithm

• Problems
  – No theoretical justification
  – Calibration issues
    • We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  – Might not always work
    • Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized
Side note about Winner Take All prediction

• If the final prediction is winner take all, is a bias feature useful?
  – Recall bias feature is a constant feature for all examples
  – Winner take all:
    \[
    \arg\max_i w_i^T x
    \]

• Answer: No
  – The bias adds a constant to all the scores
  – Will not change the prediction
2. All-vs-all classification

**Assumption**: *Every* pair of classes is separable

**Learning**: Given a dataset $D = \{<x_i, y_i>\}$,

Note: $x_i \in \mathbb{R}^n$, $y_i \in \{1, 2, \ldots, K\}$

- For every pair of labels $(j, k)$, create a binary classifier with:
  - Positive examples: All examples with label $j$
  - Negative examples: All examples with label $k$

- Train $\binom{K}{2} = \frac{K(K-1)}{2}$ classifiers in all

**Prediction**: More complex, each label gets $K-1$ votes

- How to combine the votes? Many methods
  - Majority: Pick the label with maximum votes
  - Organize a tournament between the labels

Sometimes called one-vs-one
All-vs-all classification

- Every pair of labels is linearly separable here
  - When a pair of labels is considered, all others are ignored

- Problems
  1. $O(K^2)$ weight vectors to train and store
  2. Size of training set for a pair of labels could be very small, leading to overfitting
  3. Prediction is often ad-hoc and might be unstable
    Eg: What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?
3. Error correcting output codes (ECOC)

- Each binary classifier provides one bit of information
- With $K$ labels, we only need $\log_2 K$ bits
  - One-vs-all uses $K$ bits (one per classifier)
  - All-vs-all uses $O(K^2)$ bits

- Can we get by with $O(\log K)$ classifiers?
  - Yes! Encode each label as a binary string
  - Or alternatively, if we do train more than $O(\log K)$ classifiers, can we use the redundancy to improve classification accuracy?
Using log₂K classifiers

• **Learning:**
  – Represent each label by a bit string
  – Train one binary classifier for each bit

• **Prediction:**
  – Use the predictions from all the classifiers to create a log₂N bit string that uniquely decides the output

• **What could go wrong here?**
  – Even if one of the classifiers makes a mistake, final prediction is wrong!

<table>
<thead>
<tr>
<th>#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0011</td>
</tr>
<tr>
<td>2</td>
<td>0100</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td>1011</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
</tr>
<tr>
<td>7</td>
<td>1111</td>
</tr>
</tbody>
</table>

8 classes, code-length = 3
Error correcting output code

**Answer: Use redundancy**

**Learning:**

- Assign a binary string with each label
  - Could be random
  - Length of the code word $L \geq \log_2 K$ is a parameter
- Train one binary classifier for each bit
  - Effectively, split the data into random dichotomies
  - We need only $\log_2 K$ bits
    - Additional bits act as an error correcting code
- One-vs-all is a special case.
  - How?
How to predict?

• Prediction
  – Run all L binary classifiers on the example
  – Gives us a predicted bit string of length L
  – Output = label whose code word is “closest” to the prediction
  – Closest defined using Hamming distance
    • Longer code length is better, better error-correction

• Example
  – Suppose the binary classifiers here predict 11010
  – The closest label to this is 6, with code word 11000

<table>
<thead>
<tr>
<th>#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1 0 0 1 1 0</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 1 1 0 1</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 1 1</td>
</tr>
<tr>
<td>5</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td>6</td>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td>7</td>
<td>1 1 1 1 1</td>
</tr>
</tbody>
</table>

8 classes, code-length = 5
Error correcting codes: Discussion

• Assumes that columns are independent
  – Otherwise, ineffective encoding

• Strong theoretical results that depend on code length
  – If minimal Hamming distance between two rows is $d$, then the prediction can correct up to $(d-1)/2$ errors in the binary predictions

• Code assignment could be random, or designed for the dataset/task

• One-vs-all and all-vs-all are special cases
  – All-vs-all needs a ternary code (not binary)
Decomposition methods: Summary

• **General idea**
  – Decompose the multiclass problem into many binary problems
  – We know how to train binary classifiers
  – Prediction depends on the decomposition
    • Constructs the multiclass label from the output of the binary classifiers

• **Learning optimizes local correctness**
  – Each binary classifier does not need to be globally correct
    • That is, the classifiers do not need to agree with each other
  – The learning algorithm is not even aware of the prediction procedure!

• **Poor decomposition gives poor performance**
  – Difficult local problems, can be “unnatural”
    • Eg. For ECOC, why should the binary problems be separable?

Questions?
Where are we?

• Introduction

• Combining binary classifiers
  – One-vs-all
  – All-vs-all
  – Error correcting codes

• Training a single classifier
  – Multiclass SVM
  – Constraint classification
Motivation

• Decomposition methods
  – Do not account for how the final predictor will be used
  – Do not optimize any global measure of correctness

• Goal: To train a multiclass classifier that is “global”
Recall: Margin for binary classifiers

- The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
Multiclass margin

Defined as the score difference between the highest scoring label and the second one

\[
\text{Score for a label } = w_{\text{label}}^T x
\]
Multiclass SVM (Intuition)

• Recall: Binary SVM
  – Maximize margin
  – Equivalently,
    Minimize norm of weights such that the closest points to the hyperplane have a score \( \pm 1 \)

• Multiclass SVM
  – Each label has a different weight vector (like one-vs-all)
  – Maximize multiclass margin
  – Equivalently,
    Minimize total norm of the weights such that the true label is scored at least 1 more than the second best one
Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_w \frac{1}{2} w^T w$$

s.t. $$\forall i, \quad y_i w^T x_i \geq 1$$

$$\min_{w_1, w_2, \ldots, w_K} \frac{1}{2} \sum_k w_k^T w_k$$

s.t. $$w_{y_i}^T x - w_k^T x \geq 1$$

$$\forall (x_i, y_i) \in D,$$

$$k \in \{1, 2, \ldots, K\}, \quad k \neq y_i,$$

The score for the true label is higher than the score for any other label by 1
Multiclass SVM: General case

- **Size of the weights. Effectively, regularizer**
- **Total slack. Effectively, don’t allow too many examples to violate the margin constraint**

\[
\min_{w_1,w_2,\ldots,w_K} \frac{1}{2} \sum_k w_k^T w_k
\]

\[
\text{s.t. } w_{y_i}^T x - w_k^T x \geq 1 \quad \forall (x_i, y_i) \in D, k \in \{1, 2, \ldots, K\}, k \neq y_i,
\]

- **The score for the true label is higher than the score for any other label by 1**
- **Slack variables. Not all examples need to satisfy the margin constraint.**
- **Slack variables can only be positive**
Multiclass SVM: General case

\[
\min_{w_1, w_2, \ldots, w_K, \xi} \frac{1}{2} \sum_k w_k^T w_k + C \sum_{(x_i, y_i) \in D} \xi_i
\]

s.t. \[ w_{y_i}^T x - w_k^T x \geq 1 - \xi_i, \quad \forall (x_i, y_i) \in D, \quad k \in \{1, 2, \ldots, K\}, k \neq y_i, \quad \forall i. \]

\[ \xi_i \geq 0, \quad \forall i. \]

The score for the true label is higher than the score for any other label by \(1 - \xi_i\)

Slack variables. Not all examples need to satisfy the margin constraint.

Slack variables can only be positive

Size of the weights. Effectively, regularizer

Total slack. Effectively, don’t allow too many examples to violate the margin constraint
Multiclass SVM

• Generalizes binary SVM algorithm
  – If we have only two classes, this reduces to the binary (up to scale)

• Comes with similar generalization guarantees as the binary SVM

• Can be trained using different optimization methods
  – Stochastic sub-gradient descent can be generalized
    • Try as exercise
Multiclass SVM: Summary

• **Training:**
  – Optimize the SVM objective

• **Prediction:**
  – Winner takes all
    \[ \arg\max_i w_i^T x \]

• With K labels and inputs in \( \mathbb{R}^n \), we have nK weights in all
  – Same as one-vs-all

  – But comes with guarantees!

Questions?
Where are we?

• Introduction

• Combining binary classifiers
  – One-vs-all
  – All-vs-all
  – Error correcting codes

• Training a single classifier
  – Multiclass SVM
  – Constraint classification
One-vs-all again

• Training:
  – Create K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$
  – $\mathbf{w}_i$ separates class $i$ from all others

• Prediction: $\text{argmax}_i \mathbf{w}_i^T \mathbf{x}$

• Observations:
  1. At training time, we require $\mathbf{w}_i^T \mathbf{x}$ to be positive for examples of class $i$.
  2. Really, all we need is for $\mathbf{w}_i^T \mathbf{x}$ to be more than all others, not zero
1 Vs All: Learning Architecture

- k label nodes; n input features, nk weights.
- **Evaluation**: Winner Take All
- **Training**: Each set of n weights, corresponding to the i-th label, is trained
  - Independently, given its performance on example x, and
  - Independently of the performance of label j on x.
- Hence: **Local learning**; only the final decision is global, *(Winner Takes All (WTA)).*
- However, this architecture allows multiple learning algorithms; e.g., see the implementation in the SNoW Multi-class Classifier

Targets (each an LTU)

Weighted edges (weight vectors)

Features
Recall: Winnow’s Extensions

- Winnow learns **monotone Boolean functions**
- We extended to general Boolean functions via
- “Balanced Winnow”
  - 2 weights per variable;
  - **Decision**: using the “effective weight”, the difference between $w^+$ and $w^-$
  - This is equivalent to the Winner take all decision
  - **Learning**: In principle, it is possible to use the 1-vs-all rule and update each set of $n$ weights separately, but we suggested the “balanced” Update rule that takes into account how both sets of $n$ weights predict on example $x$

$$\text{If } [(w^+ - w^-) \cdot x \geq \theta] \neq y, \quad w_i^+ \leftarrow w_i^+ r^y x_i, \quad w_i^- \leftarrow w_i^- r^{-y} x_i$$

Can this be generalized to the case of $k$ labels, $k > 2$?

We need a “global” learning approach
Extending Balanced

- In a 1-vs-all training you have a target node that represents positive examples and target node that represents negative examples.
- Typically, we train each node separately (mine/not-mine example).
- Rather, given an example we could say: this is more a + example than a – example.

\[
If \ ((w^+ - w^-) \cdot x \geq \theta) \neq y, \quad w^+_i \leftarrow w^+_i r^{y x_i}, \quad w^-_i \leftarrow w^-_i r^{-y x_i}
\]

- We compared the activation of the different target nodes (classifiers) on a given example. (This example is more class + than class -)

- Can this be generalized to the case of \( k \) labels, \( k > 2 \)?
Constraint Classification

• The examples we give the learner are pairs \((x,y), y \in \{1,...,k\}\)
• The “black box learner” we described might be thought of as a function of \(x\) only but, actually, we made use of the labels \(y\)
• How is \(y\) being used?
  – \(y\) decides what to do with the example \(x\); that is, which of the \(k\) classifiers should take the example as a positive example (making it a negative to all the others).
• How do we make decisions:
  – Let: \(f_y(x) = w_y^T \cdot x\)
  – Then, we predict using: \(y^* = \text{argmax}_{y=1,...,k} f_y(x)\)
• Equivalently, we can say that we predict as follows:
  – Predict \(y\) iff
    – \(\forall y' \in \{1,...,k\}, y' \neq y\) \((w_y^T - w_{y'}^T) \cdot x \geq 0\) (**)
• So far, we did not say how we learn the \(k\) weight vectors \(w_y\) \((y = 1,...,k)\)
  – Can we train in a way that better fits the way we make decisions?
  – What does it mean?
Linear Separability for Multiclass

- We are learning \( k \) \( n \)-dimensional weight vectors, so we can concatenate the \( k \) weight vectors into

\[
\mathbf{w} = (w_1, w_2, \ldots, w_k) \in \mathbb{R}^{nk}
\]

Notice: This is just a representational trick. We did not say how to learn the weight vectors.

- **Key Construction:** (Kesler Construction; Zimak's Constraint Classification)
  - We will represent each example \((x, y)\), as an \( nk \)-dimensional vector, \( x_y \), with \( x \) embedded in the \( y \)-th part of it \((y=1, 2, \ldots, k)\) and the other coordinates are 0.

- **E.g.,** \( x_y = (0, x, 0, 0) \in \mathbb{R}^{kn} \) \((\text{here } k=4, y=2)\)

- Now we can understand the \( n \)-dimensional decision rule:

- **Prediction:**
  - Predict \( y \) iff
    \[
    \forall y' \in \{1, \ldots, k\}, \ y' \neq y \quad (w_y^T - w_{y'}^T) \cdot x \geq 0 \quad (**)
    \]
  - In the \( nk \)-dimensional space.
  - Predict \( y \) iff
    \[
    \forall y' \in \{1, \ldots, k\}, \ y' \neq y \quad w^T \cdot (x_y - x_{y'}) = w^T \cdot x_{yy'} \geq 0
    \]

- **Conclusion:** The set \((x_{yy'}, +) \equiv (x_y - x_{y'}, +)\) is linearly separable from the set \((-x_{yy'}, -)\) using the linear separator \( w \in \mathbb{R}^{kn} \).
- We solved the voroni diagram challenge.
Constraint Classification

• Training:
  – Given a data set \{(x,y)\}, (m examples) with \(x \in \mathbb{R}^n\), \(y \in \{1,2,...,k\}\)
    create a binary classification task:
    \((x_y - x_{y'}, +), (x_{y'} - x_y -), \text{ for all } y' \neq y\) (\(2m(k-1)\) examples)
    Here \(x_y \in \mathbb{R}^{kn}\)
  – Use your favorite linear learning algorithm to train a binary classifier.

• Prediction:
  – Given an \(nk\) dimensional weight vector \(w\) and a new example \(x\), predict:
    \(\arg\max_y w^T x_y\)
Linear Separability with multiple classes \((1/3)\)

For examples with label \(i\), we want \(\mathbf{w}_i^T \mathbf{x} > \mathbf{w}_j^T \mathbf{x}\) for all \(j\)

Rewrite inputs and weight vector

- Stack all weight vectors into an \(nK\)-dimensional vector

\[
\mathbf{w} = \begin{bmatrix}
\mathbf{w}_1 \\
\mathbf{w}_2 \\
\vdots \\
\mathbf{w}_K
\end{bmatrix}_{nK \times 1}
\]

- Define a feature vector for label \(i\) being associated to input \(\mathbf{x}\):

\[
\phi(\mathbf{x}, i) = \begin{bmatrix}
0_n \\
\vdots \\
\mathbf{x} \\
\vdots \\
0_n
\end{bmatrix}_{nK \times 1}
\]

\(\mathbf{x}\) in the \(i^{th}\) block, zeros everywhere else
Linear Separability with multiple classes (2/3)

For examples with label $i$, we want $\mathbf{w}_i^T \mathbf{x} > \mathbf{w}_j^T \mathbf{x}$ for all $j$

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK \times 1}$$

$$\phi(x, i) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}$$

Equivalent requirement:

$$\mathbf{w}^T \phi(x, i) > \mathbf{w}^T \phi(x, j)$$

This is called the Kesler construction
Linear Separability with multiple classes \(3/3\)

For examples with label \(i\), we want \(\mathbf{w}_i^T \mathbf{x} > \mathbf{w}_j^T \mathbf{x}\) for all \(j\)

\[
\mathbf{w} = \begin{bmatrix}
\mathbf{w}_1 \\
\mathbf{w}_2 \\
\vdots \\
\mathbf{w}_K
\end{bmatrix}_{nK \times 1}
\]

\[
\mathbf{w}^T [\phi(\mathbf{x}, i) - \phi(\mathbf{x}, j)] > 0
\]

Equivalently, the following binary task in \(nK\) dimensions that should be linearly separable

Positive examples

\[
\phi(\mathbf{x}, i) - \phi(\mathbf{x}, j)
\]

Negative examples

\[
-\phi(\mathbf{x}, i) + \phi(\mathbf{x}, j)
\]

Example \((\mathbf{x}, i)\) in dataset, all other labels \(j\)
Constraint Classification

• **Training:**
  – Given a data set \{<x, y>\}, create a binary classification task as 
    \[ <\phi(x, y) - \phi(x, y'), +1>, <\phi(x, y') - \phi(x, y), -1> \] for all \( y' \neq y \)
  – Use your favorite algorithm to train a binary classifier
    • Exercise: What do the perceptron update rules look like in terms of the \( \phi \)s?

• **Prediction:** Given a nK dimensional weight vector \( w \) and a new example \( x \)
  \[ \arg\max_y w^T \phi(x, y) \]

• **Note:** The binary classification task expresses preferences over label assignments
  – Approach extends training a ranker, can use partial preferences too, more on this later...
Perceptron in Kesler Construction

• A perceptron update rule applied in the **nk-dimensional space** due to a mistake in $w^T \cdot x_{ij} \geq 0$
• Or, equivalently to $(w_i^T - w_j^T) \cdot x \geq 0$ (in the **n-dimensional** space)
• Implies the following update:

  Given example $(x, i)$ (example $x \in \mathbb{R}^n$, labeled $i$)
  - $\forall (i, j), i, j = 1, \ldots k, \ i \neq j$ (***)
  - If $(w_i^T - w_j^T) \cdot x < 0$ (mistaken prediction; equivalent to $w^T \cdot x_{ij} < 0$)
    - $w_i \leftarrow w_i + x$ (promotion) and $w_j \leftarrow w_j - x$ (demotion)

• Note that this is a generalization of balanced Winnow rule.
• Note that we promote $w_i$ and demote $k-1$ weight vectors $w_j$
Conservative update

- The general scheme suggests:
- Given example \((x, i)\) (example \(x \in \mathbb{R}^n\), labeled \(i\))
  - \(\forall (i,j), i,j = 1, \ldots k, i \neq j\) (***)
  - If \((w_i^T - w_j^T) \cdot x < 0\) (mistaken prediction; equivalent to \(w^T \cdot x_{ij} < 0\))
  - \(w_i \leftarrow w_i + x\) (promotion) and \(w_j \leftarrow w_j - x\) (demotion)

- Promote \(w_i\) and demote \(k-1\) weight vectors \(w_j\)
- A conservative update: (SNoW/LBJava’s implementation):
  - In case of a mistake: only the weights corresponding to the target node \(i\) and that closest node \(j\) are updated.
  - Let: \(j^* = \arg \max_{j=1,\ldots,k} w_j^T \cdot x\) (highest activation among competing labels)
  - If \((w_i^T - w_{j^*}^T) \cdot x < 0\) (mistaken prediction)
  - \(w_i \leftarrow w_i + x\) (promotion) and \(w_{j^*} \leftarrow w_{j^*} - x\) (demotion)
  - Other weight vectors are not being updated.
Significance

- The hypothesis learned above is more expressive than when the OvA assumption is used.
- Any linear learning algorithm can be used, and algorithmic-specific properties are maintained (e.g., attribute efficiency if using winnow.)
- E.g., the multiclass support vector machine can be implemented by learning a hyperplane to separate $P(S)$ with maximal margin.

- As a byproduct of the linear separability observation, we get a natural notion of a margin in the multi-class case, inherited from the binary separability in the nk-dimensional space.
  - Given example $x_{ij} \in \mathbb{R}^{nk}$, $\text{margin}(x_{ij}, w) = \min_{ij} w^T \cdot x_{ij}$
  - Consequently, given $x \in \mathbb{R}^n$, labeled $i$
    \[
    \text{margin}(x, w) = \min_j (w_i^T - w_j^T) \cdot x
    \]
A second look at the multiclass margin

Defined as the score difference between the highest scoring label and the second one

In terms of Kesler construction:

$$\min_{y' \neq y} w^T [\phi(x, y) - \phi(x, y')]$$

Here $y$ is the label that has the highest score.
Discussion

• What is the number of weights for multiclass SVM and constraint classification?
  – nK. Same as One-vs-all, much less than all-vs-all K(K-1)/2

• But both still account for all pairwise label preferences
  – Multiclass SVM via the definition of the learning objective
    \[ w_{y_i}^T x - w_k^T x \geq 1 - \xi_i \]
  – Constraint classification by constructing a binary classification problem

• Both come with theoretical guarantees for generalization

• Important idea that is applicable when we move to arbitrary structures

Questions?
Training multiclass classifiers: Wrap-up

• Label belongs to a set that has more than two elements

• Methods
  – Decomposition into a collection of binary \((local)\) decisions
    • One-vs-all
    • All-vs-all
    • Error correcting codes
  – Training a single \((global)\) classifier
    • Multiclass SVM
    • Constraint classification

• Exercise: Which of these will work for this case?

Questions?
Next steps...

• Build up to structured prediction
  – Multiclass is really a simple structure

• Different aspects of structured prediction
  – Deciding the structure, training, inference

• Sequence models