From Binary to Multiclass Classification
So far: Binary Classification

• We have seen linear models
• Learning algorithms
  – Perceptron
  – SVM
  – Logistic Regression

• Prediction is simple
  – Given an example $\mathbf{x}$, prediction $= sgn(\mathbf{w}^T \mathbf{x})$
  – Output is a single bit
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?

  A A A A A A A A A A A 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  Car tire  Duck  laptop
Example applications: Language

- **Input**: a news article
- **Output**: Which section of the newspaper should be be in

- **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** \( x \in \mathbb{R}^n \)
  - The inputs are represented by their feature vectors

• **Output** \( y \in \{1, 2, \ldots, K\} \)
  - These classes represent domain-specific labels

• **Learning**: Given a dataset \( D = \{(x_i, y_i)\} \)
  - Need a learning algorithm that uses D to construct a function that can predict \( x \) to \( y \)
  - Goal: find a predictor that does well on the training data and has low generalization error

• **Prediction/Inference**: Given an example \( x \) and the learned function, compute the class label for \( x \)
1. One-vs-all classification

- **Assumption**: Each class individually separable from *all* the others
1. One-vs-all classification

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

- **Learning:** Given a dataset $D = \{(x_i, y_i)\}$
  - 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 $w_1, w_2, \cdots w_K$ using any learning algorithm we have seen
1. One-vs-all classification

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

- **Learning:** Given a dataset $D = \{(x_i, y_i)\}$
  - Train K binary classifiers $w_1, w_2, \ldots w_K$ using any learning algorithm we have seen

- **Prediction:** “Winner Takes All”
  \[
  \arg\max_i w_i^T x
  \]
1. One-vs-all classification

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

- **Learning:** Given a dataset $D = \{(x_i, y_i)\}$
  - Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K$ using any learning algorithm we have seen

- **Prediction:** "*Winner Takes All*"
  \[
  \text{label} = \text{argmax}_i \mathbf{w}_i^T \mathbf{x}
  \]

**Question:** What is the dimensionality of each $\mathbf{w}_i$?
Visualizing One-vs-all
Visualizing One-vs-all

From the full dataset, construct three binary classifiers, one for each class.
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
Visualizing One-vs-all

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

- For **blue** inputs: \( w_{\text{blue}}^T x > 0 \)
- For **red** inputs: \( w_{\text{red}}^T x > 0 \)
- For **green** inputs: \( w_{\text{green}}^T x > 0 \)
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

Notation: Score for blue label
Visualizing One-vs-all

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

- For blue inputs, $w_{\text{blue}}^\top x > 0$
- For red inputs, $w_{\text{red}}^\top x > 0$
- For green inputs, $w_{\text{green}}^\top x > 0$

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

\[
\begin{align*}
\mathbf{w}_{\text{blue}}^T \mathbf{x} > 0 & \quad \text{for blue inputs} \\
\mathbf{w}_{\text{red}}^T \mathbf{x} > 0 & \quad \text{for red inputs} \\
\mathbf{w}_{\text{green}}^T \mathbf{x} > 0 & \quad \text{for green inputs} \\
\text{???} & \quad \text{???}
\end{align*}
\]
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:
    $$\arg\max_i w_i^T x$$
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

Sometimes called one-vs-one

- **Assumption**: *Every* pair of classes is separable
2. All-vs-all classification

Sometimes called one-vs-one

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

• **Learning**: Given a dataset $D = \{(x_i, y_i)\}$, $y \in \{1,2,\ldots,K\}$
  
  – 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 to separate every pair of labels from each other
2. All-vs-all classification

Sometimes called one-vs-one

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

- **Learning:** Given a dataset \( D = \{ (x_i, y_i) \}, \quad x \in \mathbb{R}^n, \quad y \in \{1, 2, \ldots, K\} \)
  
  - Train \( \binom{K}{2} = \frac{K(K-1)}{2} \) classifiers to separate every pair of labels from each other

- **Prediction:** More complex, each label get K-1 votes
  
  - How to combine the votes? Many methods
    
    - Majority: Pick the label with maximum votes
    
    - Organize a tournament between the labels
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 of the binary classifiers
  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 $\log_2 K$ 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 $\log_2 N$ bit string that uniquely decides the output

- **What could go wrong here?**

<table>
<thead>
<tr>
<th>label#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>

8 classes, code-length = 3
Using \( \log_2 K \) 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 \( \log_2 N \) 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>label#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>

8 classes, code-length = 3
Answer: Use redundancy

- 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</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)
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)

Exercise: Convince yourself that this is correct
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 have 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?
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: the distance between the hyperplane and the data point nearest to it.

Margin with respect to this hyperplane
Multiclass margin

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

Score for a label $= w_{\text{label}}^T x$
Multiclass margin

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

Score for a label $= \mathbf{w}_{\text{label}}^T \mathbf{x}$
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

\[
\min_w \quad \frac{1}{2} w^T w \\
\text{s.t.} \forall i, \quad y_i w^T x_i \geq 1
\]

\[
\min_{w_1, w_2, \ldots, w_K} \quad \frac{1}{2} \sum_k w_k^T w_k \\
\text{s.t.} \quad w_{y_i}^T x - w_k^T x \geq 1 \\
\forall (x_i, y_i) \in D, \\
k \in \{1, 2, \ldots, K\}, k \neq y_i,
\]
Multiclass SVM in the separable case

Recall hard binary SVM

\[
\min_w \quad \frac{1}{2} w^T w \\
\text{s.t. } \forall i, \quad y_i w^T x_i \geq 1
\]

\[
\min_{w_1, w_2, \ldots, w_K} \frac{1}{2} \sum_k w_k^T w_k \\
\text{s.t. } w_{y_i}^T x - w_k^T x \geq 1 \\
\forall (x_i, y_i) \in D, \\
k \in \{1, 2, \ldots, K\}, k \neq y_i,
\]

The score for the true label is higher than the score for \textit{any} other label by 1
Multiclass SVM in the separable case

Recall hard binary SVM

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

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

Size of the weights.
Effectively, regularizer

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

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

$$\forall (x_i, y_i) \in D,$$
$$k \in \{1, 2, \ldots, K\}, k \neq y_i,$$

The score for the true label is higher than the score for any other label by 1

Problems with this?
Multiclass SVM in the separable case

Recall hard binary SVM
\[
\begin{align*}
\min_{w} & \quad \frac{1}{2} w^T w \\
\text{s.t.} & \quad \forall i, \quad y_i w^T x_i \geq 1
\end{align*}
\]

Effectively, regularizer

\[
\begin{align*}
\min_{w_1, w_2, \cdots, w_K} & \quad \frac{1}{2} \sum_{k} w_k^T w_k \\
\text{s.t.} & \quad w_{y_i}^T x - w_{k}^T x \geq 1 \\
& \quad \forall (x_i, y_i) \in D, \\
& \quad k \in \{1, 2, \cdots, K\}, k \neq y_i,
\end{align*}
\]

The score for the true label is higher than the score for any other label by 1

Problems with this?

What if there is no set of weights that achieves this separation? That is, what if the data is not linearly separable?
Multiclass SVM: General case

The score for the true label is higher than the score for \textit{any} other label by $1 - \xi_i$.

Slack variables. Not all examples need to satisfy the margin constraint.
Multiclass SVM: General case

\[
\begin{align*}
\min_{w_1, w_2, \cdots, w_K} & \quad \frac{1}{2} \sum_{k} w_k^T w_k + C \sum_{(x_i, y_i) \in D} \xi_i \\
\text{s.t.} & \quad w_{y_i}^T x - w_k^T x \geq 1 - \xi_i \quad \forall (x_i, y_i) \in D, \\
& \quad k \in \{1, 2, \cdots, K\}, k \neq y_i,
\end{align*}
\]

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

Size of the weights. Effectively, regularizer

Total slack. Don’t allow too many examples to violate the margin constraint

Slack variables. Not all examples need to satisfy the margin constraint.
Multiclass SVM: General case

Size of the weights. Effectively, regularizer

\[
\begin{align*}
\min_{w_1, w_2, \cdots, w_K} & \quad \frac{1}{2} \sum_k w_k^T w_k + C \sum_{(x_i, y_i) \in D} \xi_i \\
\text{s.t.} & \quad w_{y_i}^T x - w_k^T x \geq 1 - \xi_i, \quad \forall (x_i, y_i) \in D, \\
& \quad k \in \{1, 2, \cdots, K\}, k \neq y_i, \\
& \quad \forall i.
\end{align*}
\]

\[\xi_i \geq 0.\]

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
Multiclass SVM: General case

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

Total slack. Don’t allow too many examples to violate the margin constraint

Size of the weights. Effectively, regularizer

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

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

Slack variables can only be positive.
Multiclass SVM: General case

\[
\min_{w_1, w_2, \ldots, w_K, \xi} \frac{1}{2} \sum_k w_k^T w_k + C \sum_{(x_i, y_i) \in D} \xi_i \\
\text{s.t. } w_{y_i}^T x - w_k^T x \geq 1 - \xi_i, \quad \forall (x_i, y_i) \in D,
\]

\[
k \in \{1, 2, \ldots, K\}, k \neq y_i, \quad \forall i.
\]

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. Don’t allow too many examples to violate the margin constraint

Slack variables can only be positive

The score for the true label is higher than the score for any other label by \(1 - \xi_i\)
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
    \[ \text{argmax}_i \mathbf{w}_i^T \mathbf{x} \]

• 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
Let us examine 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: $\operatorname*{argmax}_i \mathbf{w}_i^T \mathbf{x}$

• 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
     » The requirement of being positive is stronger
Linear Separability with multiple classes

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

$$
\mathbf{w} = \begin{bmatrix}
\mathbf{w}_1 \\
\mathbf{w}_2 \\
\vdots \\
\mathbf{w}_K
\end{bmatrix}_{nK \times 1}
$$
Linear Separability with multiple classes

For examples with label $i$, we want $w_i^T x > w_j^T x$ for all $j$

Rewrite inputs and weight vector

- Stack all weight vectors into an $nK$-dimensional vector

- Define a feature vector for label $i$ being associated to input $x$:

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

$\mathbf{x}$ in the $i^{th}$ block, zeros everywhere else

This is called the Kesler construction
Linear Separability with multiple classes

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

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

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

\( \mathbf{x} \) in the \( i \)th block, zeros everywhere else

Equivalent requirement:

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

Or:

\[
\mathbf{w}^T [\phi(\mathbf{x}, i) - \phi(\mathbf{x}, j)] > 0
\]
Linear Separability with multiple classes

*For examples with label* \( i \), we want \( w_i^T x > w_j^T x \) for all \( j \)

Or equivalently: 
\[
\mathbf{w}^T [\phi(x, i) - \phi(x, j)] > 0
\]
Linear Separability with multiple classes

For examples with label $i$, we want $w_i^T x > w_j^T x$ for all $j$

Or equivalently: $w^T [\phi(x, i) - \phi(x, j)] > 0$

That is, the following binary task in $nK$ dimensions that should be linearly separable

For every example $(x, i)$ in dataset, all other labels $j$
Constraint Classification

• **Training:**
  – Given a data set \{\( (x, y) \)\}, create a binary classification task
    • Positive examples: \( \phi(x, y) - \phi(x, y') \)
    • Negative examples: \( \phi(x, y') - \phi(x, y) \)
      for every example, for every \( y' \neq y \)
  – Use your favorite algorithm to train a **binary classifier**
Constraint Classification

• **Training:**
  – Given a data set \( \{(x, y)\} \), create a binary classification task
    • Positive examples: \( \phi(x, y) - \phi(x, y') \)
    • Negative examples: \( \phi(x, y') - \phi(x, y) \)
    for every example, for every \( y' \neq y \)
  – Use your favorite algorithm to train a binary classifier

• **Prediction:** Given a nK dimensional weight vector \( w \) and a new example \( x \)
  \[ \text{argmax}_y w^T \phi(x, y) \]
Constraint Classification

• **Training:**
  – Given a data set \{ (x, y) \}, create a binary classification task
    • Positive examples: \( \phi(x, y) - \phi(x, y') \)
    • Negative examples: \( \phi(x, y') - \phi(x, y) \)
      for every example, for every \( y' \neq y \)
  – Use your favorite algorithm to train a binary classifier

• **Prediction:** Given a nK dimensional weight vector \( \mathbf{w} \) and a new example \( x \)
  \[
  \text{argmax}_y \ w^T \phi(x, y)
  \]
Constraint Classification

• **Training:**
  – Given a data set \{((x, y))\}, create a binary classification task
    • Positive examples: \(\phi(x, y) - \phi(x, y')\)
    • Negative examples: \(\phi(x, y') - \phi(x, y)\)
      for every example, for every \(y' \neq y\)
  – Use your favorite algorithm to train a binary classifier

• **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 only expresses preferences over label assignments
This approach extends to training a ranker, can use partial preferences too, more on this later...
A second look at the multiclass margin

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

• The number of weights for multiclass SVM and constraint classification is still 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