From Binary to Multiclass Classification

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

We have seen binary classification

- We have seen linear models
- Learning algorithms
  - Perceptron
  - SVM
  - Logistic Regression

- Prediction is simple
  - Given an example $\mathbf{x}$, output $= \operatorname{sgn}(\mathbf{w}^T\mathbf{x})$
  - Output is a single bit

What if we have more than two labels?

Multiclass classification

- Introduction

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

- Training a single classifier
  - Multiclass SVM
  - Constraint classification

Where are we?

- 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: examples 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?

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 an algorithm for training binary classifiers to construct a multiclass classifier?
  – Answer: Decompose the prediction into multiple binary decisions

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

General setting

- Input $\mathbf{x} \in \Re^n$
  - The inputs are represented by their feature vectors
- Output $\mathbf{y} \in \{1, 2, \cdots, K\}$
  - These classes represent domain-specific labels
- Learning: Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$
  - Need a learning algorithm that uses D to construct a function that can predict $\mathbf{x}$ to $\mathbf{y}$
  - 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, compute the class label for $\mathbf{x}$

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 $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

  $\boldsymbol{x} \in \Re^n$
  $\boldsymbol{y} \in \{1,2, \cdots, K\}$

  – 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

1. One-vs-all classification

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

- Learning: Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

  $\mathbf{x} \in \Re^n$
  $\mathbf{y} \in \{1,2,\cdots,K\}$

  – 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 $\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_K$ 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 $D = \{(\mathbf{x}_\mathrm{i}, \mathbf{y}_i)\}$

  $\boldsymbol{x} \in \Re^n$
  $\boldsymbol{y} \in \{1,2,\cdots,K\}$

  – 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*"

$$\mathrm{argmax}_i\ \mathbf{w}_i^T\mathbf{x}$$

1. One-vs-all classification

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

$x \in \Re^n$
$y \in \{1,2,\cdots,K\}$

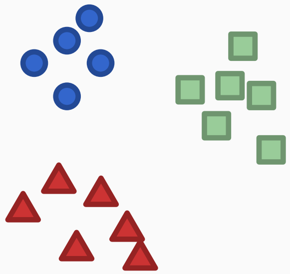
- Learning: Given a dataset $D = \{(\mathbf{x}_{\mathrm{i}}, \mathbf{y}_i)\}$
  - Train K binary classifiers $\mathbf{w}_1$, $\mathbf{w}_2$, $\cdots$ $\mathbf{w}_{\mathrm{K}}$ using any learning algorithm we have seen

> Question: What is the dimensionality of each $\mathbf{w}_{\mathrm{i}}$?

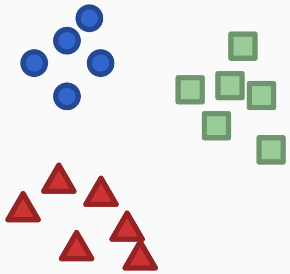
- Prediction: "*Winner Takes All*"

$$\mathrm{argmax}_i\ \mathbf{w}_i{}^T\mathbf{x}$$

Visualizing One-vs-all

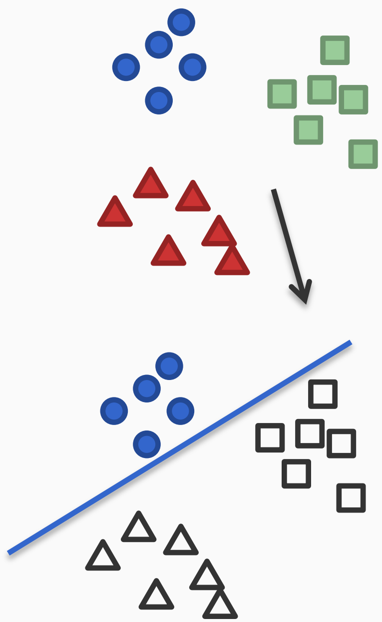


Visualizing One-vs-all



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

Visualizing One-vs-all

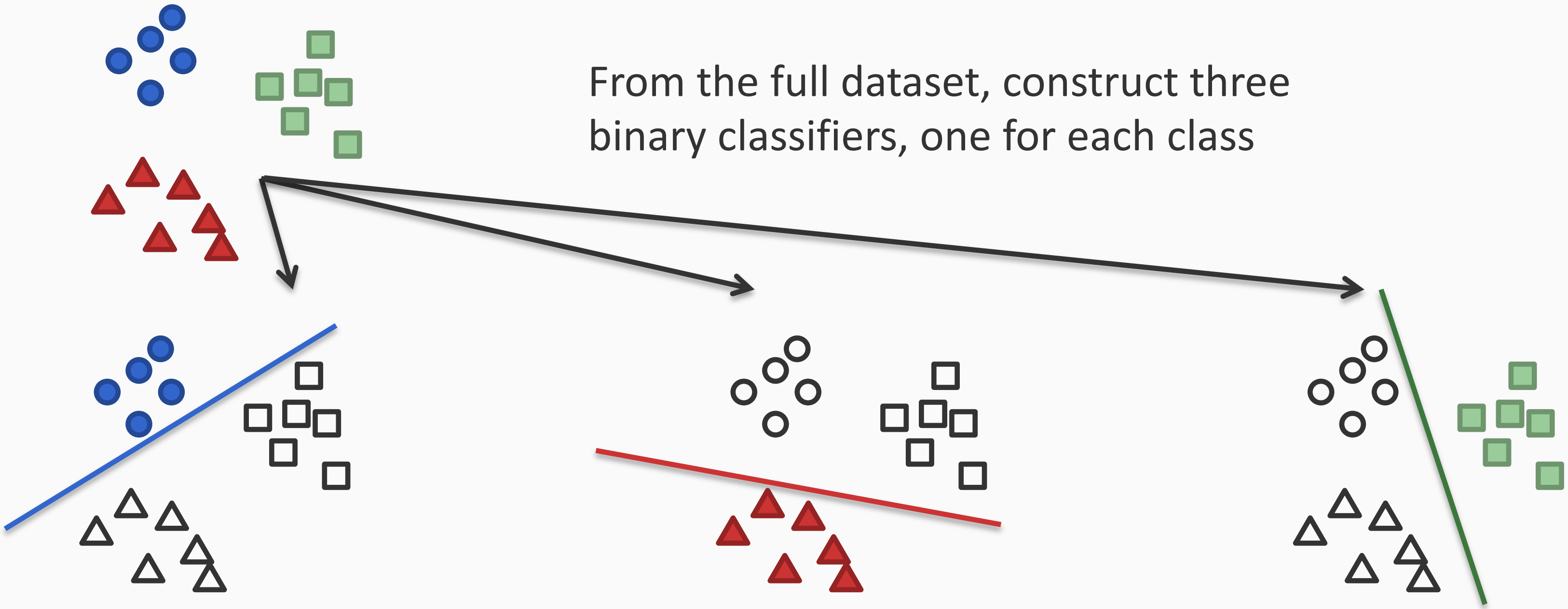


$\mathbf{w}_{\mathbf{blue}}^T\mathbf{x} > 0$
for **blue circle inputs**

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

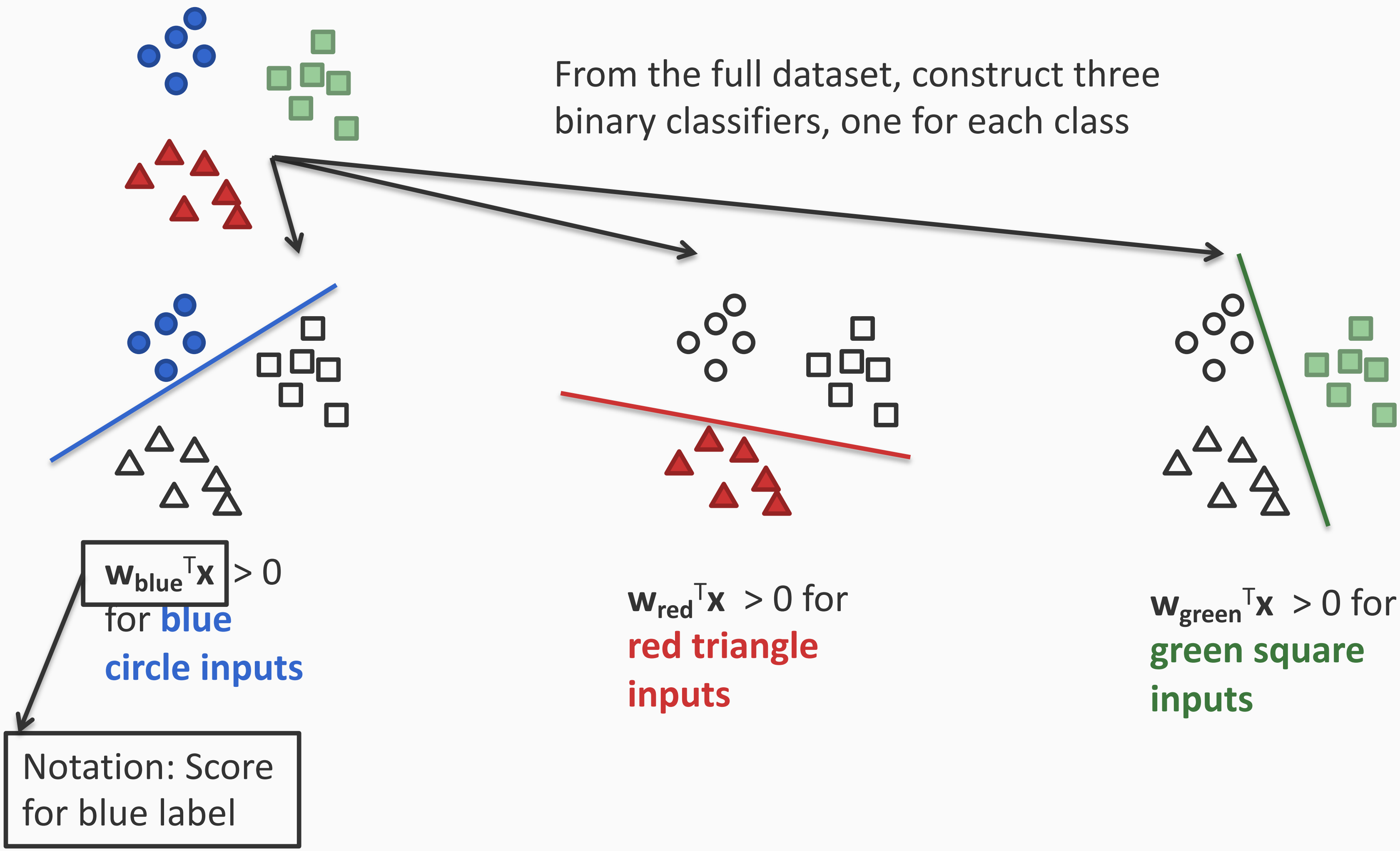


$\mathbf{w}_{blue}^T\mathbf{x} > 0$ for **blue circle inputs**

$\mathbf{w}_{red}^T\mathbf{x} > 0$ for **red triangle inputs**

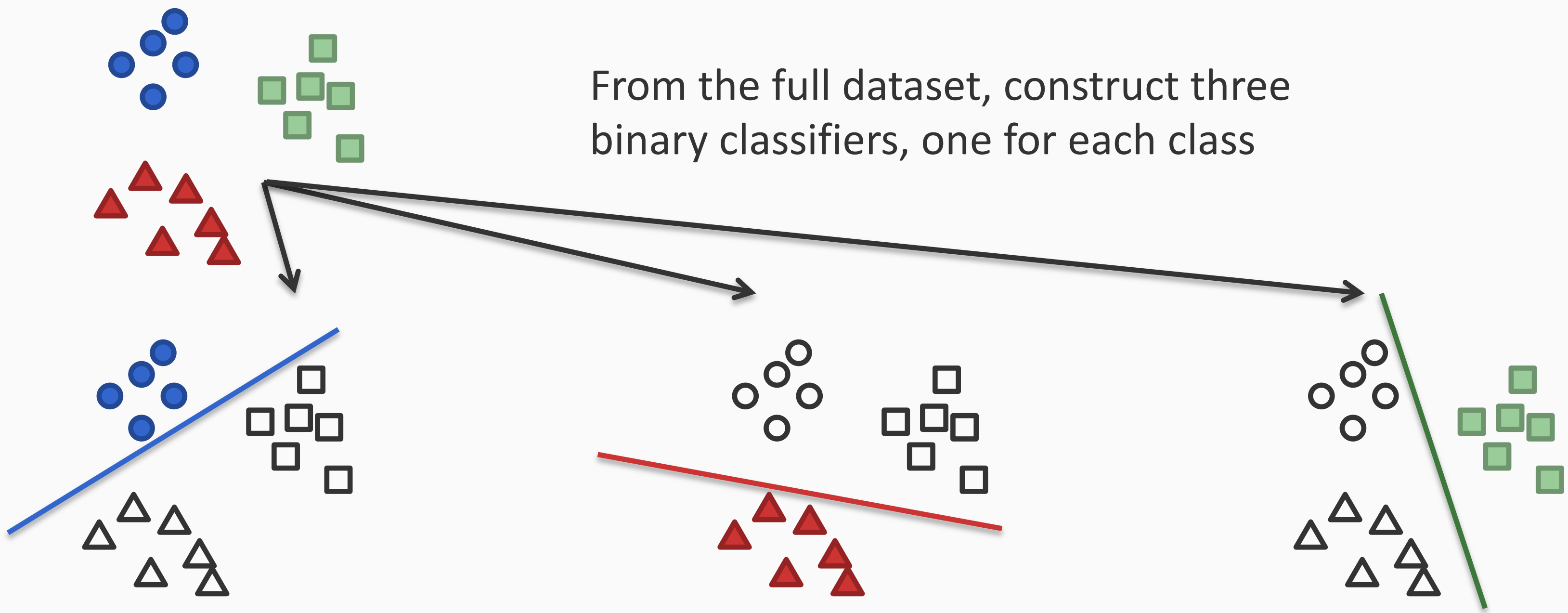
$\mathbf{w}_{green}^T\mathbf{x} > 0$ for **green square inputs**

Visualizing One-vs-all



Visualizing One-vs-all

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



$\boxed{\mathbf{w}_{\mathbf{blue}}^T\mathbf{x}} > 0$ for **blue circle inputs**

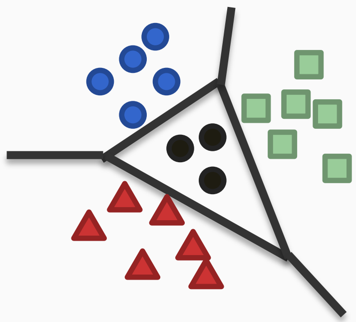
Notation: Score for blue label

$\mathbf{w}_{\mathbf{red}}^T\mathbf{x} > 0$ for **red triangle inputs**

$\mathbf{w}_{\mathbf{green}}^T\mathbf{x} > 0$ for **green square inputs**

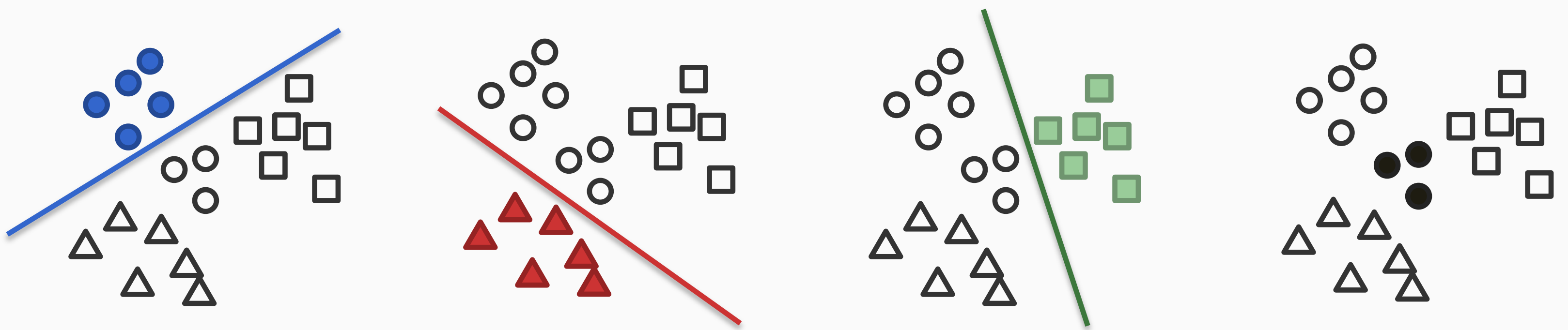
Winner Take All will predict the right answer. Only the correct label will have a positive score

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!



$\mathbf{w}_{blue}^T\mathbf{x} > 0$ for **blue circle inputs**

$\mathbf{w}_{red}^T\mathbf{x} > 0$ for **red triangle inputs**

$\mathbf{w}_{green}^T\mathbf{x} > 0$ for **green square inputs**

???

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

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 $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, $\quad \mathbf{x} \in \Re^n$, $\mathbf{y} \in \{1,2,\cdots,K\}$
  - 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

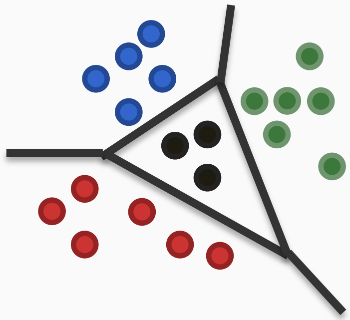
Sometimes called one-vs-one

- Assumption: *Every* pair of classes is separable

- Learning: Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, $\mathbf{x} \in \Re^n$, $\mathbf{y} \in \{1, 2, \cdots, K\}$
  - 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 to represent the label
  - 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

label#	Code		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

8 classes, code-length = 3

- Learning:
  - Represent each label by a bit string (i.e., its code)
  - Train one binary classifier for each bit
- Prediction:
  - Use the predictions from all the classifiers to create a $\log_2 K$ bit string that uniquely decides the output

Example: For some example, if the three classifiers predict 0, 1 and 1, then the label is 3

Using $\log_2 K$ classifiers

label#	Code		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

8 classes, code-length = 3

- Learning:
  - Represent each label by a bit string (i.e., its code)
  - Train one binary classifier for each bit
- Prediction:
  - Use the predictions from all the classifiers to create a $\log_2 K$ bit string that uniquely decides the output
- What could go wrong here?

Using $\log_2 K$ classifiers

label#	Code		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

8 classes, code-length = 3

- Learning:
  - Represent each label by a bit string (i.e., its code)
  - Train one binary classifier for each bit

- Prediction:
  - Use the predictions from all the classifiers to create a $\log_2 K$ 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!

Error correcting output coding

#	Code				
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

8 classes, code-length = 5

Answer: Use redundancy

- Assign a binary string with each label
  - Could be random
  - Length of the code word **L** >= $\log_2 K$ 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

How to predict?

#	Code				
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

8 classes, code-length = 5

- 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

How to predict?

#	Code				
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

8 classes, code-length = 5

- 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

One-vs-all is a special case of this scheme. How?

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

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 $\frac{d-1}{2}$ errors in the binary predictions
- Code assignment could be random, or designed for the dataset or 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 $\frac{d-1}{2}$ errors in the binary predictions
- Code assignment could be random, or designed for the dataset or 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 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

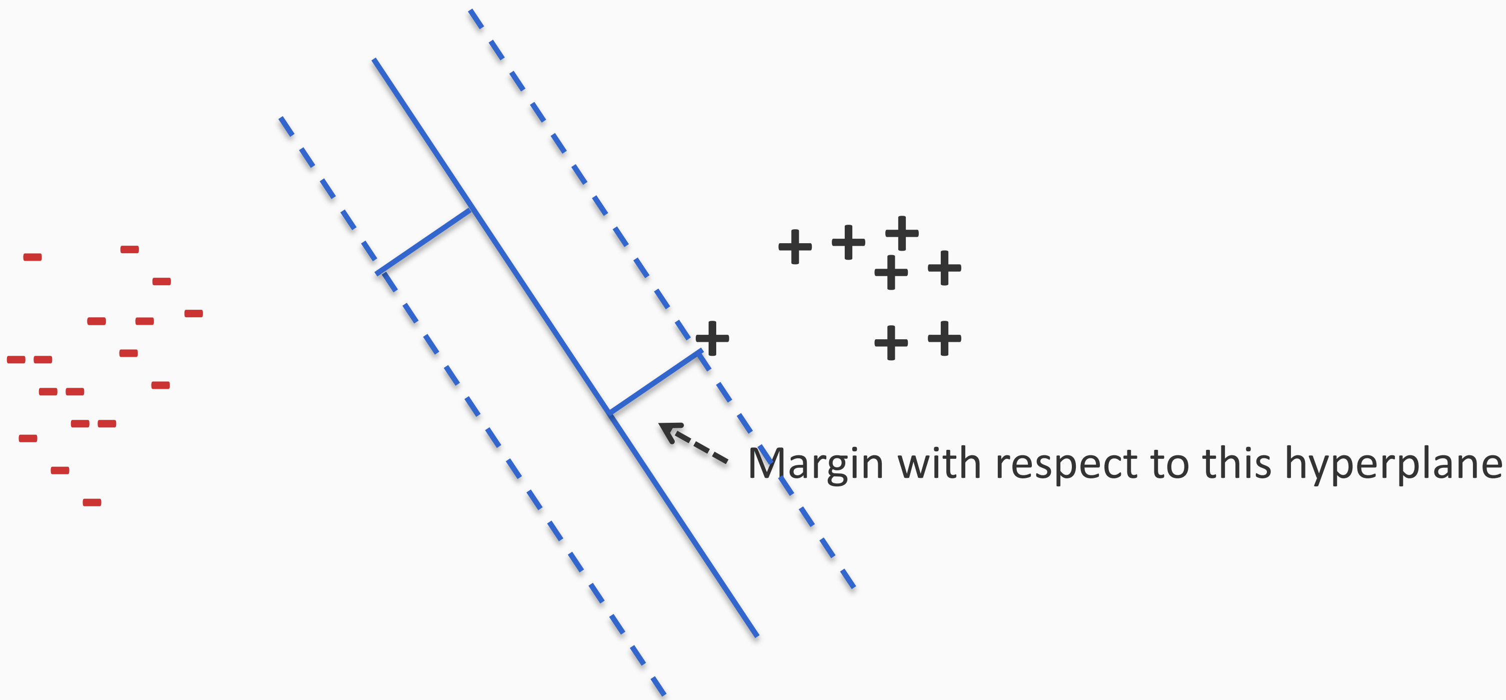
- Training a single classifier
  - Multiclass SVM
  - Constraint classification
  - Multiclass logistic regression

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”

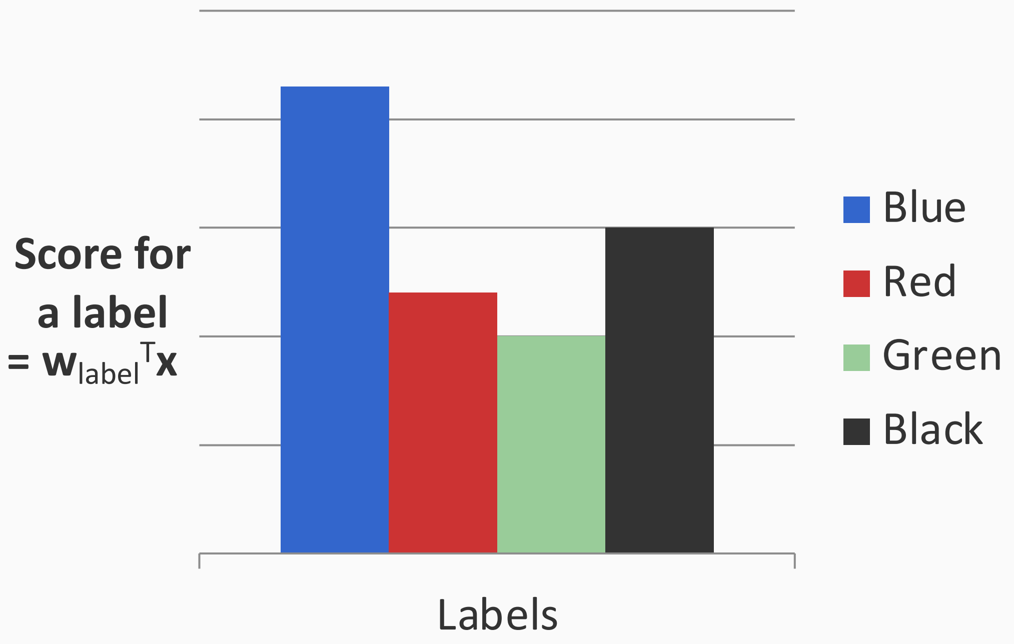
Recall: Margin for binary classifiers

The margin of a hyperplane for a dataset: the distance between the hyperplane and the data point nearest to it



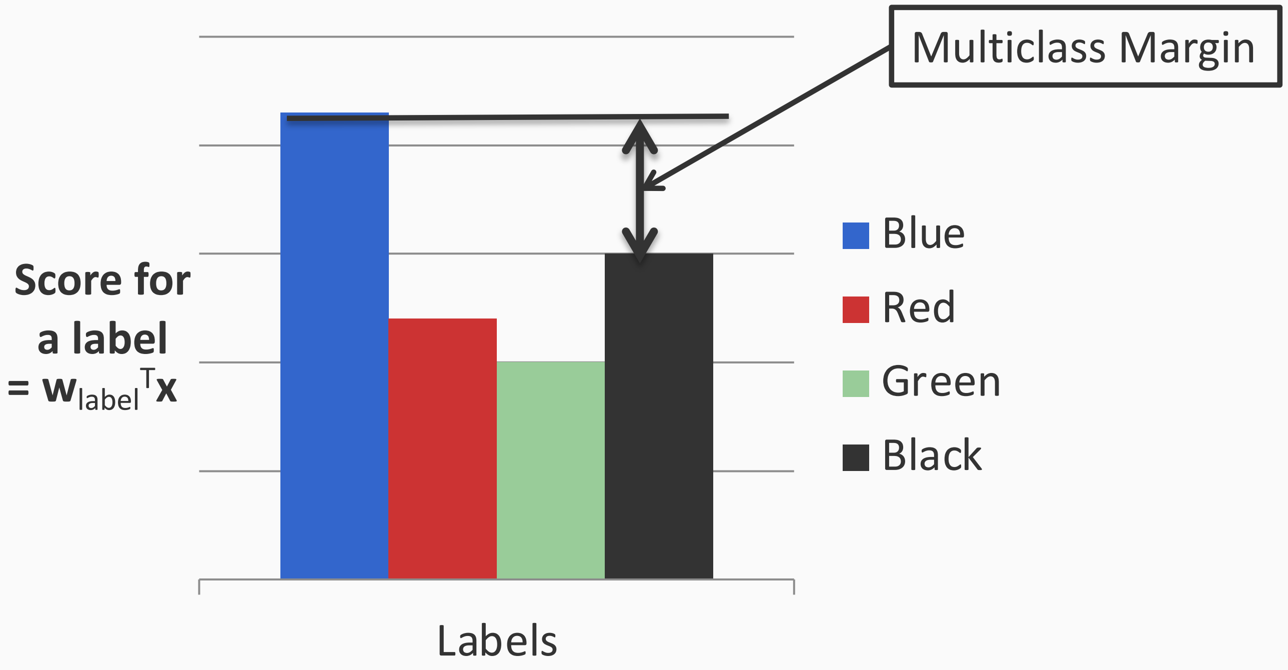
Multiclass margin

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



Multiclass margin

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



Multiclass SVM (Intuition)

- Recall: Binary SVM
  - Maximize margin
  - Equivalently,

    Minimize norm of weights such that the closest points to the hyperplane have a score ≥ 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

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.}\forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad Regularizer(\mathbf{w}_1,\cdots,\boldsymbol{w}_K)$$

$$\text{s.t.} \quad score(y_i) - score(k) \geq 1 \quad \forall(\mathbf{x}_i,\mathbf{y}_i)\in D,$$
$$k\in\{1,2,\cdots,K\}, k\neq \mathbf{y}_i,$$

Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K} \quad Regularizer(\mathbf{w}_1, \cdots, \boldsymbol{w}_K)$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 \qquad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$
$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.}\forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k$$
$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 \qquad \forall(\mathbf{x}_i,\mathbf{y}_i) \in D,$$
$$k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$

Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T \mathbf{w}_k$$
$$\text{s.t.} \quad \underline{\mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1} \qquad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$
$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

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

Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.}\forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

Size of the weights. Effectively, regularizer

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 \qquad \forall(\mathbf{x}_i,\mathbf{y}_i) \in D, \; k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$

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

Multiclass SVM in the separable case

Recall hard binary SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

Size of the weights.
Effectively, regularizer

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T \mathbf{w}_k$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 \qquad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$
$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

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

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.}\forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

Size of the weights.
Effectively, regularizer

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 \qquad \forall(\mathbf{x}_i,\mathbf{y}_i) \in D, \; k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$

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

Size of the weights. Effectively, regularizer

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T \mathbf{w}_k$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 - \xi_i, \qquad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$
$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

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.

Multiclass SVM: General case

Size of the weights. Effectively, regularizer

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

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k + \boxed{C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} \xi_i}$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 - \xi_i, \qquad \forall(\mathbf{x}_i,\mathbf{y}_i)\in D,$$
$$k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$

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.

Multiclass SVM: General case

Size of the weights. Effectively, regularizer

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

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T \mathbf{w}_k + \boxed{C \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in D} \xi_i}$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 - \xi_i, \qquad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$

$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

$$\boxed{\xi_i \geq 0,} \qquad \forall i.$$

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

Multiclass SVM: General case

Size of the weights. Effectively, regularizer

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

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K,\xi} \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k + \boxed{C\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D}\xi_i}$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 - \xi_i, \qquad \forall(\mathbf{x}_i,\mathbf{y}_i)\in D,$$

$$k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$

$$\boxed{\xi_i \geq 0,} \qquad \forall i.$$

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

Multiclass SVM: General case

Size of the weights. Effectively, regularizer

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

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K, \xi} \quad \frac{1}{2} \sum_k \mathbf{w}_k^T \mathbf{w}_k + C \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in D} \xi_i$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 - \xi_i, \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D,$$

$$k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

$$\xi_i \geq 0, \quad \forall i.$$

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

Multiclass SVM: General case

Solving

$$\begin{array}{lll} \min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K,\xi} & \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} \xi_i & \\ \text{s.t.} & \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 - \xi_i, & \forall(\mathbf{x}_i,\mathbf{y}_i)\in D, \\ & & k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i, \\ & \xi_i \geq 0, & \forall i. \end{array}$$

Is equivalent to solving

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \frac{1}{2}\sum_i \mathbf{w}_i^T\mathbf{w}_i + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in \mathrm{D}} \max\left(0, \max_{k\neq \mathbf{y}_i}\left(\mathbf{w}_k^T\mathbf{x}_i - \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x}_i + 1\right)\right)$$

Why?

Multiclass SVM: General case

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \frac{1}{2}\sum_i \mathbf{w}_i^T \mathbf{w}_i + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in \mathrm{D}} \max\left(0, \max_{k\neq \mathbf{y}_i}\left(\mathbf{w}_k^T \mathbf{x}_i - \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x}_i + 1\right)\right)$$

Size of the weights.
Effectively, regularizer

Multiclass SVM: General case

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \frac{1}{2}\sum_i \mathbf{w}_i^T\mathbf{w}_i + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} \boxed{\max\left(0, \max_{k\neq\mathbf{y}_i}\left(\mathbf{w}_k^T\mathbf{x}_i - \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x}_i + 1\right)\right)}$$

Size of the weights.
Effectively, regularizer

The multiclass hinge loss

Multiclass SVM: General case

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \frac{1}{2}\sum_i \mathbf{w}_i^T\mathbf{w}_i + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in \mathrm{D}} \max\left(0, \max_{k\neq \mathbf{y}_i}\left(\mathbf{w}_k^T\mathbf{x}_i - \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x}_i + 1\right)\right)$$

The tradeoff hyperparameter

Size of the weights. Effectively, regularizer

The multiclass hinge loss

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 $\Re^n$, 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
  - Multiclass logistic regression

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

- **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

     The requirement of being positive is more strict

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 $\mathbf{w}_i^T\mathbf{x} > \mathbf{w}_j^T\mathbf{x}$ *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 **x**:

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

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*

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK\times 1} \qquad \phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK\times 1}$$

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

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*

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

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

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

i^th block

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*

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

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

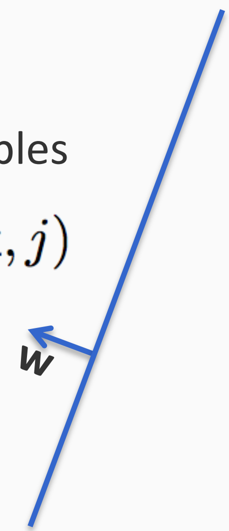
That is, the following binary task in nK dimensions that should be linearly separable

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

i^{th} block

Positive examples

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



Negative examples

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

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**

$$\mathrm{argmax}_{\mathrm{y}} \mathbf{w}^{\mathrm{T}} \phi(x, y)$$

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**

Exercise: What do the perceptron update rule look like in terms of the ϕ's?

Interpret the update step

- Prediction: Given a nK dimensional weight vector **w** and a new example **x**

$$\mathrm{argmax}_{\mathrm{y}} \mathbf{w}^{\mathrm{T}} \phi(x, y)$$

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**

- Pre … r **w**

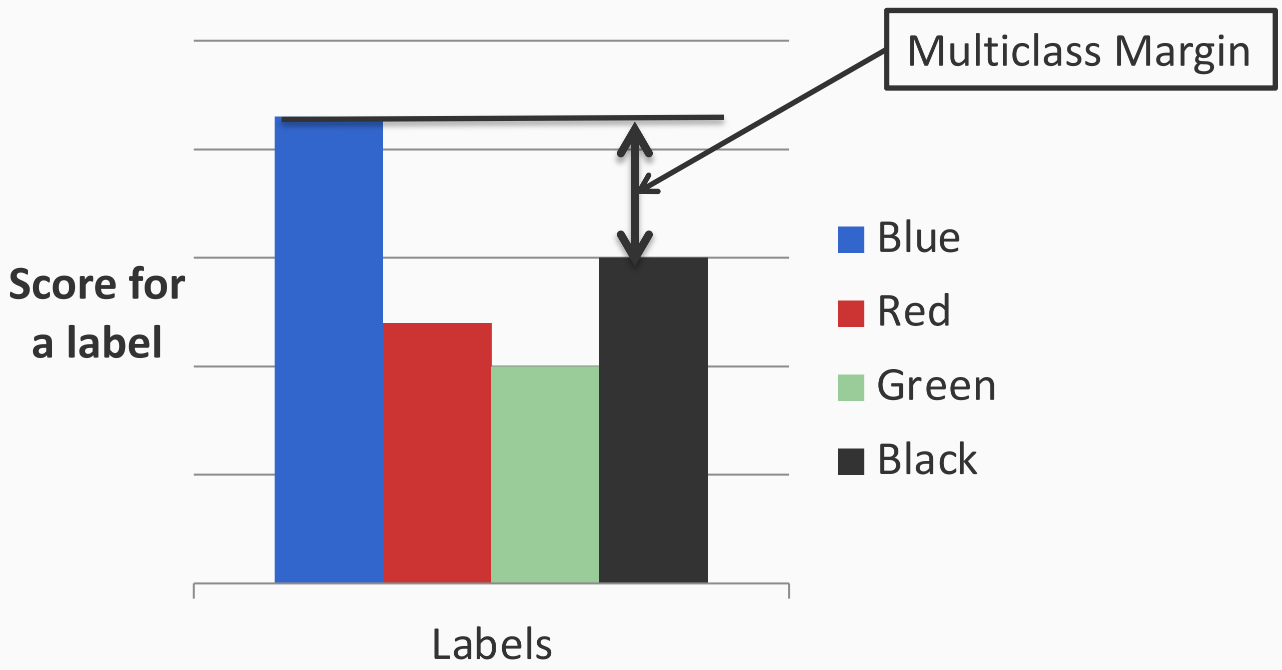
  an …

  $\operatorname{argmax}_y \mathbf{w}^T \phi(x,y)$

> ***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…

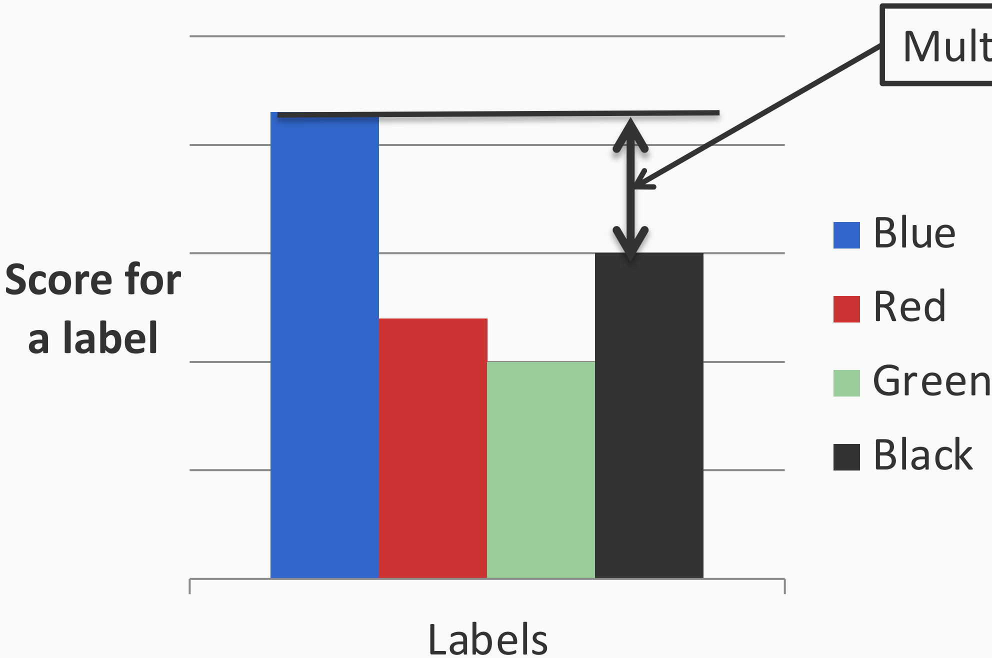
A second look at the multiclass margin

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



Multiclass Margin

In terms of Kesler construction

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

Here **y** is the label that has the highest score

Where are we?

- Introduction
- Combining binary classifiers
  - One-vs-all
  - All-vs-all
  - Error correcting codes
- Training a single classifier
  - Multiclass SVM
  - Constraint classification
  - Multiclass logistic regression

Multiclass logistic regression

Known by many other names:

- Polytomous logistic regression
- Multinomial logistic regression
- Softmax logistic regression
- Log-linear model for logistic regression

Multiclass logistic regression

General setting (same as before)

- Inputs: $\mathbf{x}$
- Output: $\mathbf{y} \in \{1, 2, \cdots, K\}$
- Feature representation: $\phi(\mathbf{x}, \mathbf{y})$

Kesler construction

Multiclass logistic regression

General setting (same as before)

- Inputs: $\mathbf{x}$
- Output: $\mathbf{y} \in \{1, 2, \cdots, K\}$
- Feature representation: $\phi(\mathbf{x}, \mathbf{y})$

Define probability of an input $\mathbf{x}$ taking a label $\mathbf{y} = i$ as

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_{j=1}^{K} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

Multiclass logistic regression

Define probability of an input $\mathbf{x}$ taking a label $\mathbf{y}$ as

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_{j=1}^{K} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

Interpretation: Score each label, and then convert to a well-formed probability distribution by exponentiating + normalizing

Multiclass logistic regression

Define probability of an input $\mathbf{x}$ taking a label $\mathbf{y}$ as

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_{j=1}^{K} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

Interpretation: Score each label, and then
convert to a well-formed probability distribution by exponentiating + normalizing

Multiclass logistic regression

Define probability of an input $\mathbf{x}$ taking a label $\mathbf{y}$ as

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_{j=1}^{K} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

This expression uses the softmax function:

$$\text{softmax}(z_1, z_2, \cdots) = \left(\frac{\exp z_1}{\sum_j \exp z_j}, \frac{\exp z_2}{\sum_j \exp z_j}, \cdots\right)$$

Multiclass logistic regression

Define probability of an input $\mathbf{x}$ taking a label $\mathbf{y}$ as

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_{j=1}^{K} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

When we take log of the probability, we have a linear term and a term that doesn't depend on the label

$$\log P(y \mid \mathbf{x}, \mathbf{w}) = w^T \phi(\mathbf{x}, y) - \log Z(\mathbf{x})$$

Such models are also called *log-linear* models

Training for multiclass logistic regression

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_j \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

Given a data set $\mathrm{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

– Apply the maximum likelihood principle

$$\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

– Maybe with a regularizer

$$\max_{\mathbf{w}} -\frac{\lambda}{2} \mathbf{w}^T \mathbf{w} + \sum_i \log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Training for multiclass logistic regression

$$P(\mathbf{y} = i \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, i))}{\sum_j \exp\left(\mathbf{w}^T \phi(\mathbf{x}, j)\right)}$$

Given a data set $\mathrm{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

- Apply the maximum likelihood principle

$$\max_{\mathbf{w}} \sum_i \underbrace{\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})}$$

The cross-entropy loss

- Maybe with a regularizer

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \sum_i \underbrace{\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})}$$

Another training idea: MaxEnt

(Minor detour)

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_{i} \phi_i(x_i, y_i) = \sum_{i} \sum_{y} P(\, y \mid x_i, w \,) \phi_j(x_i, y)$$

For every feature j

Another training idea: MaxEnt

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_i \phi_i(x_i, y_i) = \sum_i \sum_y P(y \mid x_i, w)\phi_j(x_i, y)$$

There can be many conditional probability distributions that satisfy this constraint.

What is a trivial one that does so?

Another training idea: MaxEnt

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_i \phi_i(x_i, y_i) = \sum_i \sum_y P(\, y \mid x_i, w \,)\phi_j(x_i, y)$$

There can be many conditional probability distributions that satisfy this constraint.

We need a principled way to choose between such distributions.

Another training idea: MaxEnt

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_i \phi_i(x_i, y_i) = \sum_i \sum_y P(\, y \mid x_i, w\,)\phi_j(x_i, y)$$

There can be many conditional probability distributions that satisfy this constraint.

We need a principled way to choose between such distributions:

Find a distribution that satisfies the constraint,
and does not make any other commitments otherwise.
That is, given the constraint, it is maximally uncertain otherwise.

Another training idea: MaxEnt

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_i \phi_i(x_i, y_i) = \sum_i \sum_y P(\, y \mid x_i, w\,)\phi_j(x_i, y)$$

Recall: Entropy of a distribution $P(y \mid x)$ is

$$H(P) = -\sum_i P(\, y_i \mid x_i\,) \log P(\, y_i \mid x_i\,)$$

- A measure of smoothness
- Without any other information, maximized by the uniform distribution

Another training idea: MaxEnt

Consider all distributions P such that the empirical counts of the features matches the expected counts

$$\sum_i \phi_i(x_i, y_i) = \sum_i \sum_y P(y \mid x_i, w)\phi_j(x_i, y)$$

Recall: Entropy of a distribution $P(y \mid x)$ is

$$H(P) = -\sum_i P(y_i \mid x_i) \log P(y_i \mid x_i)$$

- A measure of smoothness
- Without any other information, maximized by the uniform distribution

Maximum entropy learning

argmax$_p$ H(p) such that it satisfies this constraint

Maximum Entropy distribution = log-linear

Theorem

The maximum entropy distribution among those satisfying the constraint has an exponential form

Among exponential distributions, the maximum entropy distribution is the most likely distribution

Questions?

Discussion

- The number of weights for multiclass SVM, constraint classification, multiclass logistic regression are still same as One-vs-all, much less than all-vs-all

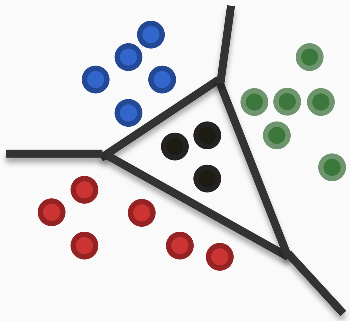
- All account for pairwise label preferences
  - Multiclass SVM via the definition of the learning objective
    $$\mathbf{w}_{y_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 - \xi_i$$
  - Constraint classification by constructing a binary classification problem
  - Multiclass logistic regression because the probability is normalized (i.e. softmax)

- Important ideas that are 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
    - Multiclass logistic regression
- 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

Extra: Training a log-linear model

Training a log-linear model

- Gradient based methods to minimize

$$L(\mathbf{w}) = -\sum_i \log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

- Usual stochastic gradient descent
  – Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
  – Iterate through examples for multiple epochs
    - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
      – Update $\boldsymbol{w} \leftarrow \boldsymbol{w}\ - \mathrm{r_t} \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
  – Return $\boldsymbol{w}$

Training a log-linear model

- Gradient based methods to minimize

$$L(\mathbf{w}) = -\sum_i \log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

- Usual stochastic gradient descent
  - Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
  - Iterate through examples for multiple epochs
    - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
      - Update $\boldsymbol{w} \leftarrow \boldsymbol{w}\ - \mathrm{r}_\mathrm{t} \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
  - Return $\boldsymbol{w}$

Other methods exist

For example the L-BFGS algorithm

Training a log-linear model

- Gradient based methods to minimize

$$L(\mathbf{w}) = -\sum_i \log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

- Usual stochastic gradient descent
  - Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
  - Iterate through examples for multiple epochs
    - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
      - Update $\boldsymbol{w} \leftarrow \boldsymbol{w}\ - \mathrm{r}_t \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
  - Return $\boldsymbol{w}$

A vector, whose j[th] element is the derivative of L with $\mathbf{w}_j$.
Has a neat interpretation

Gradients of the loss function

Let us compute this derivative of L with respect to w

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$$

Gradients of the loss function

Let us compute this derivative of L with respect to w

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$\begin{aligned} L(\mathbf{w}, \mathbf{x}, \mathbf{y}) &= -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \\ &= -\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \log \sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right) \end{aligned}$$

Gradients of the loss function

Let us compute this derivative of L with respect to w

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$\begin{aligned} L(\mathbf{w}, \mathbf{x}, \mathbf{y}) &= -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \\ &= -\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \log \sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right) \end{aligned}$$

The derivative of the loss with respect to the weights is:

$$\frac{\partial L}{\partial \mathbf{w}} = -\phi(\mathbf{x}, \mathbf{y}) + \frac{\sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right) \phi(\mathbf{x}, \mathbf{y}')}{\sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

Gradients of the loss function

Let us compute this derivative of L with respect to w

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$\begin{aligned} L(\mathbf{w}, \mathbf{x}, \mathbf{y}) &= -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \\ &= -\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \log \sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right) \end{aligned}$$

The derivative of the loss with respect to the weights is:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= -\phi(\mathbf{x}, \mathbf{y}) + \frac{\sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right) \phi(\mathbf{x}, \mathbf{y}')}{\sum_{y'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)} \\ &= -\phi(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{y}'} P(\mathbf{y}' \mid \mathbf{x}, \mathbf{w}) \phi(\mathbf{x}, \mathbf{y}') \end{aligned}$$

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}'))}$$

$$L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$$

Gradients of the loss function

- Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
- Iterate through examples for multiple epochs
  - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
    - Update $\boldsymbol{w} \leftarrow \boldsymbol{w} - \mathrm{r_t} \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
- Return $\boldsymbol{w}$

A vector, whose j^{th} element is the derivative of L with $\mathbf{w}_j$.
Has a neat interpretation

$$\frac{\partial}{\partial \mathbf{w}} L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i) = \phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \sum_{\boldsymbol{y}'} P(\boldsymbol{y}' | \boldsymbol{x}_i, \mathbf{w}) \phi(\boldsymbol{x}_i, \boldsymbol{y}')$$

Gradients of the loss function

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$$

- Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
- Iterate through examples for multiple epochs
  - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
    - Update $\boldsymbol{w} \leftarrow \boldsymbol{w} - r_t \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
- Return $\boldsymbol{w}$

A vector, whose j^th element is the derivative of L with $\mathbf{w}_j$.
Has a neat interpretation

$$\frac{\partial}{\partial \mathbf{w}} L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i) = \phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \sum_{\boldsymbol{y}'} P(\boldsymbol{y}' | \boldsymbol{x}_i, \mathbf{w}) \phi(\boldsymbol{x}_i, \boldsymbol{y}')$$

Features for the true output

Gradients of the loss function

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp\left(\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}')\right)}$$

$$L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = -\log P(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$$

- Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$
- Iterate through examples for multiple epochs
  - For each example $(\boldsymbol{x}_i\ \boldsymbol{y}_i)$ take gradient step for the loss at that example
    - Update $\boldsymbol{w} \leftarrow \boldsymbol{w}\ - \mathrm{r_t} \nabla L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$
- Return $\boldsymbol{w}$

A vector, whose j^{th} element is the derivative of L with $\mathbf{w}_j$.
Has a neat interpretation

$$\frac{\partial}{\partial \mathbf{w}} L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i) = \phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \sum_{\boldsymbol{y}'} P(\boldsymbol{y}' | \boldsymbol{x}_i, \mathbf{w}) \phi(\boldsymbol{x}_i, \boldsymbol{y}')$$

Features for the true output

The expected feature vector according to the current model