Lecture 3:
Multiclass Classification

Kai-Wei Chang
CS @ University of Virginia
kw@kwchang.net

Some slides are adapted from Vivek Skirmar and Dan Roth

Announcement

- Please enroll in Piazza and Collab
- Makeup class for 2/8 (Wed) (2/10 or 2/17)

Previous Lecture

- Binary linear classification models
  - Perceptron, SVMs, Logistic regression
- Prediction is simple:
  - Given an example x, prediction is $sgn(w^T \mathrm{x})$
  - Note that all these linear classifier have the same inference rule
  - In logistic regression, we can further estimate the probability

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

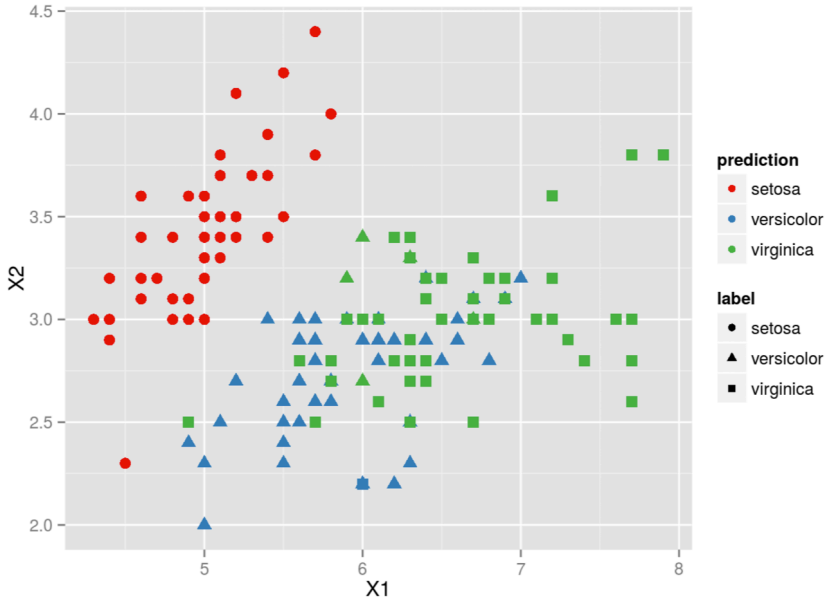
- Question?

This Lecture

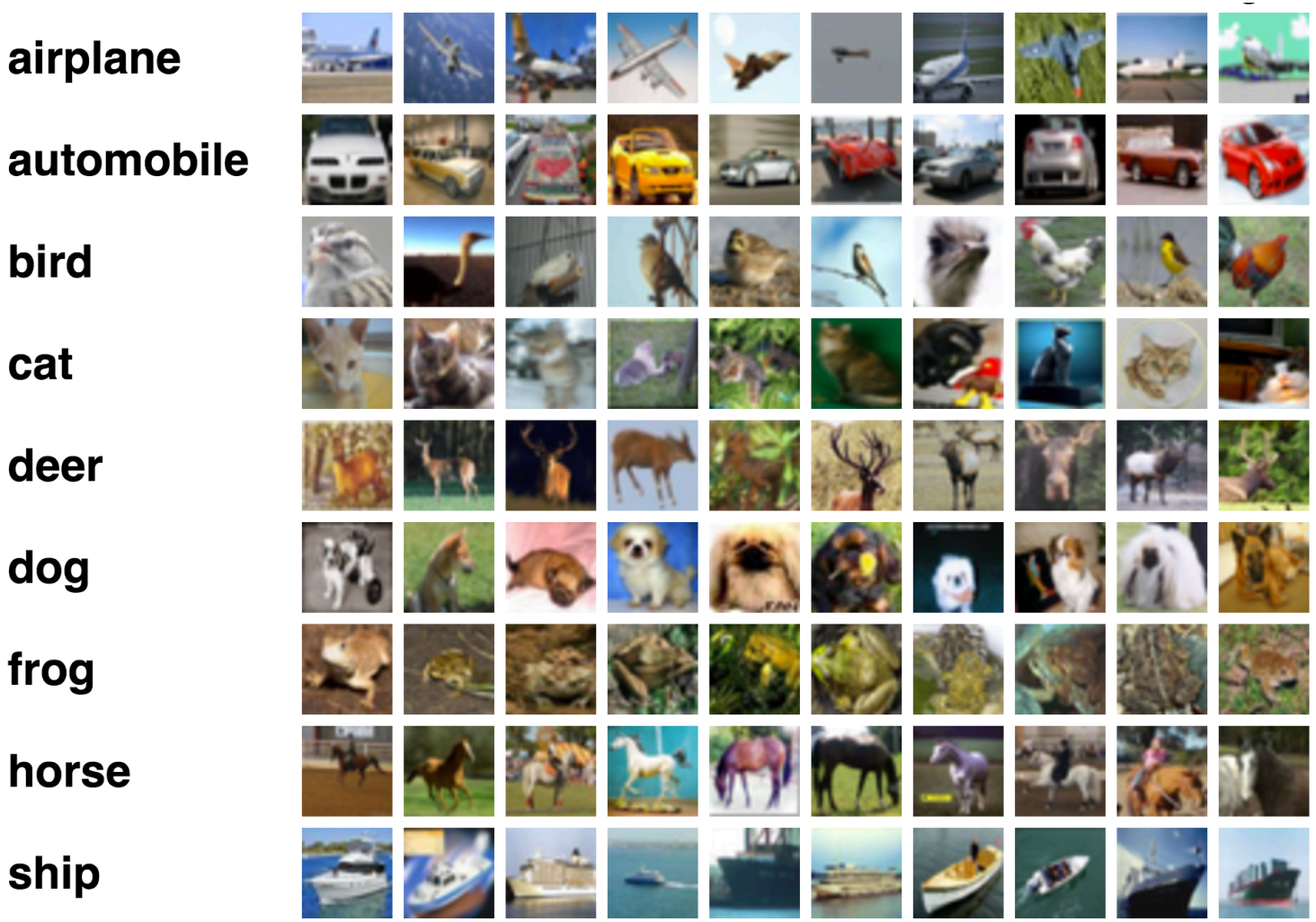
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler's construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression

What is multiclass

- Output $\in \{1,2,3, \ldots K\}$
  - In some cases, output space can be very large (i.e., K is very large)
- Each input belongs to exactly one class (c.f. in multilabel, input belongs to many classes)



Example applications



Two key ideas to solve multiclass

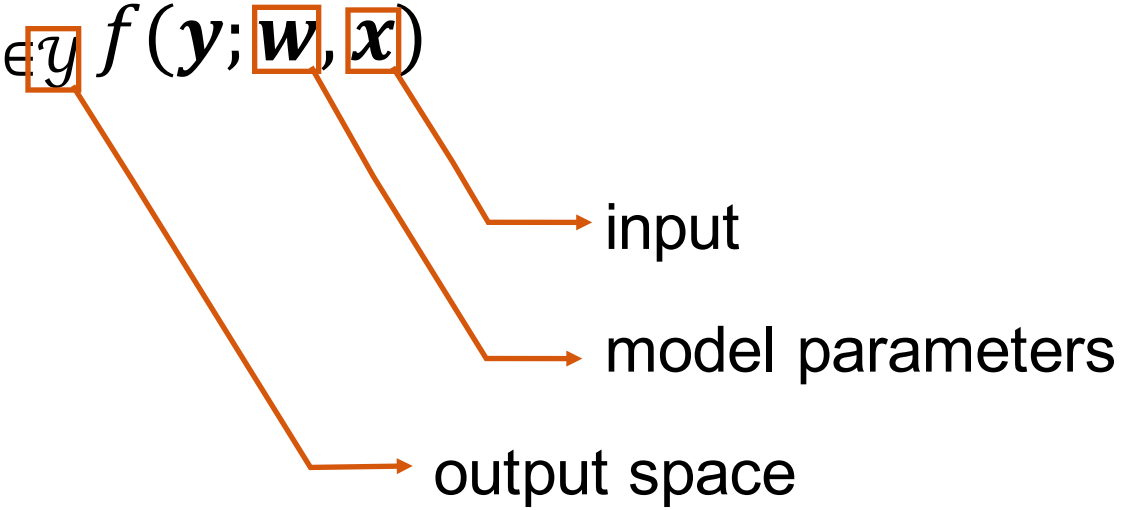
- Reducing multiclass to binary
  - Decompose the multiclass prediction into multiple binary decisions
  - Make final decision based on multiple binary classifiers
- Training a single classifier
  - Minimize the empirical risk
  - Consider all classes simultaneously

Reduction v.s. single classifier

- Reduction
  - Future-proof: binary classification improved so does muti-class
  - Easy to implement
- Single classifier
  - Global optimization: directly minimize the empirical loss; easier for joint prediction
  - Easy to add constraints and domain knowledge

A General Formula

$$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$$



- Inference/Test: given $\boldsymbol{w}, \boldsymbol{x}$, solve argmax
- Learning/Training: find a good $\boldsymbol{w}$
- Today: $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}, \mathcal{Y} = \{1, 2, \ldots K\}$ (multiclass)

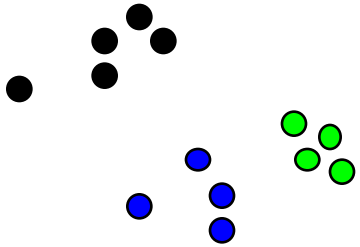
This Lecture

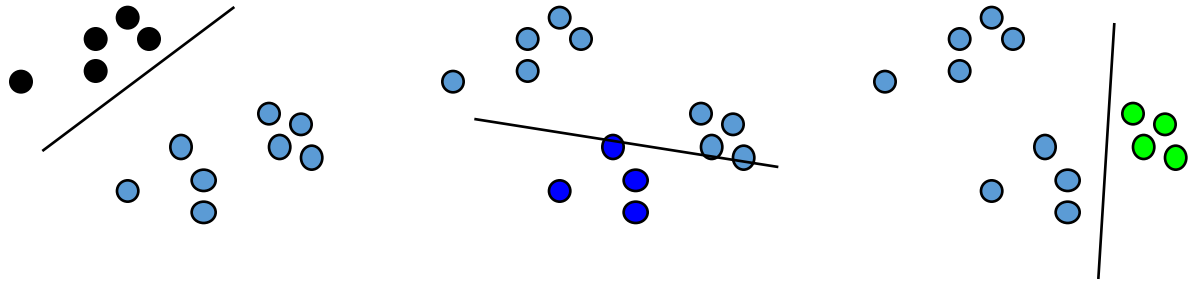
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler's construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression

One against all strategy

One against All learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,...,k\}$
- Decompose into binary problems



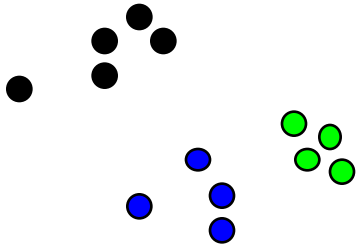


One-again-All learning algorithm

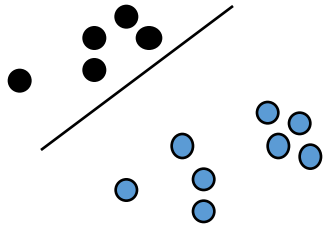
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - The binary classification can be solved by any algorithm we have seen

One against All learning

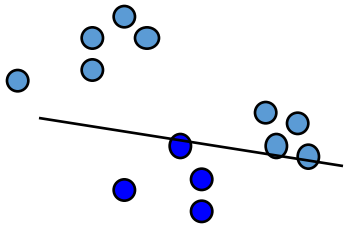
- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,...,k\}$
- Decompose into binary problems



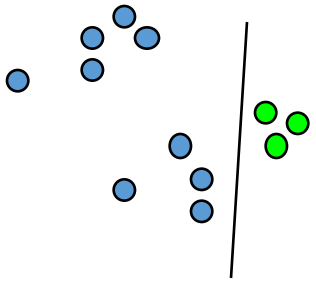
Ideal case: only the correct label will have a positive score



$w_{black}^T x > 0$



$w_{blue}^T x > 0$



$w_{green}^T x > 0$

One-again-All Inference

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \dots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \dots w_K$
- Inference: "Winner takes all"
  - $\hat{y} = \text{argmax}_{y \in \{1,2,\dots K\}} w_y^T x$

For example: $\text{y} = \text{argmax}(w_{black}^T x\ ,\ w_{blue}^T x,\ w_{green}^T x)$

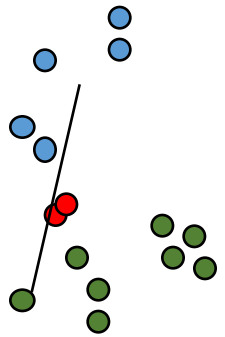
  - An instance of the general form

$$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$$

$w = \{w_1, w_2, \dots w_K\}$, $f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w}_y^T x$

One-again-All analysis

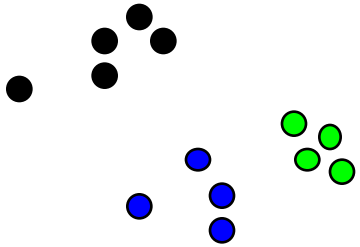
- Not always possible to learn
  - Assumption: each class individually separable from all the others
- No theoretical justification
  - Need to make sure the range of all classifiers is the same – we are comparing scores produced by K classifiers trained independently.
- Easy to implement; work well in practice

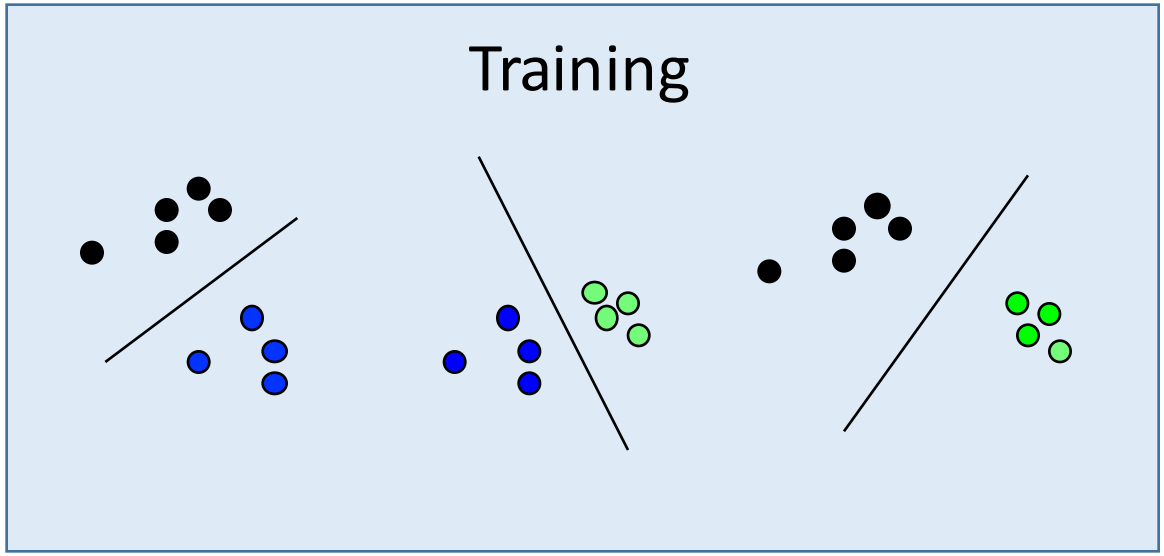


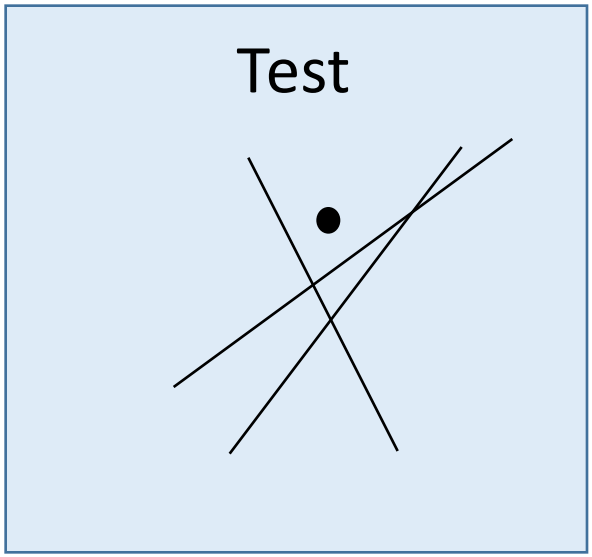
One v.s. One (All against All) strategy

One v.s. One learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,...,k\}$
- Decompose into binary problems







One-v.s-One learning algorithm

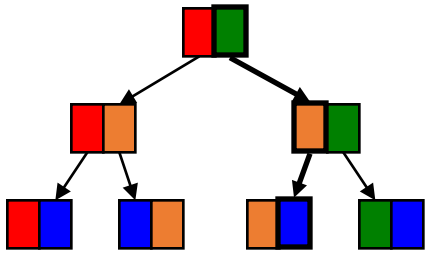
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into C(K,2) binary classification tasks
  - Learn C(K,2) models: $w_1, w_2, w_3, \ldots w_{K*(K-1)/2}$
  - For each class pair (i,j), construct a binary classification task as:
    - Positive examples: Elements of D with label i
    - Negative examples Elements of D with label j
    - The binary classification can be solved by any algorithm we have seen

One-v.s-One Inference algorithm

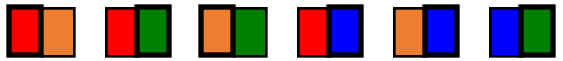
- Decision Options:
  - More complex; each label gets k-1 votes
  - Output of binary classifier may not cohere.
  - Majority: classify example x to take label i if i wins on x more often than j (j=1,…k)
  - A tournament: start with n/2 pairs; continue with winners

Classifying with One-vs-one

Tournament



Majority Vote



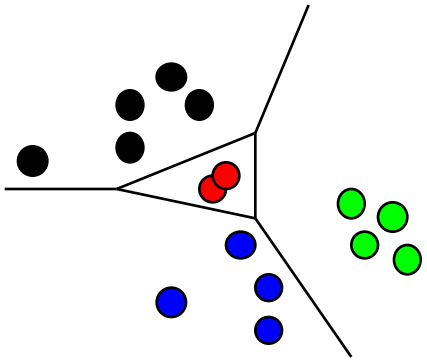
1 red, 2 yellow, 2 green

➔ ?

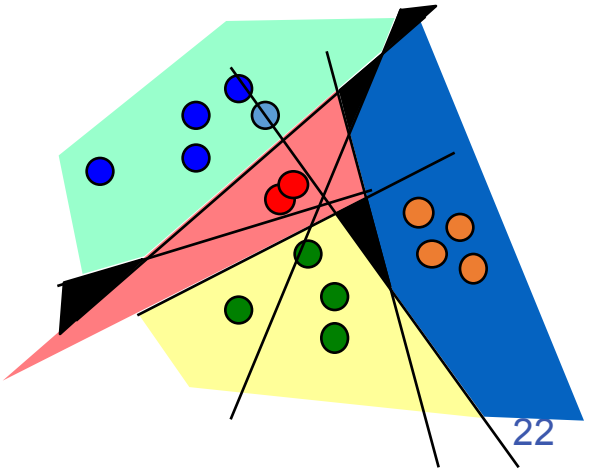
All are post-learning and *might* cause weird stuff

One-v.s.-one Assumption

- Every pair of classes is separable



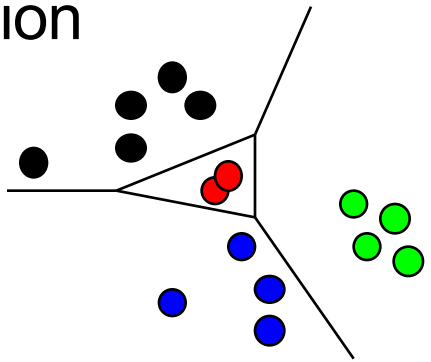
It is possible to separate all k classes with the $O(k^2)$ classifiers



Decision Regions

Comparisons

- One against all
  - O(K) weight vectors to train and store
  - Training set of the binary classifiers may unbalanced
  - Less expressive; make a strong assumption



- One v.s. One (All v.s. All)
  - O(K^2) weight vectors to train and store
  - Size of training set for a pair of labels could be small ⇒ overfitting of the binary classifiers
  - Need large space to store model

Problems with Decompositions

- Learning optimizes over *local* metrics
  - Does not guarantee good *global* performance
  - We don't care about the performance of the *local* classifiers
- Poor decomposition ⇒ poor performance
  - Difficult local problems
  - Irrelevant local problems
- Efficiency: e.g., All vs. All vs. One vs. All
- Not clear how to generalize multi-class to problems with a very large # of output

Still an ongoing research direction

Key questions:

- How to deal with large number of classes
- How to select "right samples" to train binary classifiers

- Error-correcting tournaments
  [Beygelzimer, Langford, Ravikumar 09]
- Logarithmic Time One-Against-Some
  [Daume, Karampatziakis, Langford, Mineiro 16]
- Label embedding trees for large multi-class tasks.
  [Bengio, Weston, Grangier 10]
- …

Decomposition methods: Summary

- General Ideas:
  - Decompose the multiclass problem into many binary problems
  - Prediction depends on the decomposition
    - Constructs the multiclass label from the output of the binary classifiers
- Learning optimizes local correctness
  - Each binary classifier don't need to be globally correct and isn't aware of the prediction procedure

This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler's construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression

Revisit One-again-All learning algorithm

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - w_k: separate class k from others
- Prediction

$$\hat{y} = \text{argmax}_{y \in \{1,2,\ldots K\}} \, w_y^T \, x$$

Observation

- At training time, we require $w_i^T x$ to be positive for examples of class i.
- Really, all we need is for $w_i^T x$ to be more than all others ⇒ this is a weaker requirement

For examples with label i, we need

$$w_i^T x > w_j^T x \qquad \text{for all } j$$

Perceptron-style algorithm

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

- For each training example (x, y)
  - If for some y', $w_y^T x \leq w_{y'}^T x$ mistake!
    - $w_y \leftarrow w_y + \eta x$ update to promote y
    - $w_{y'} \leftarrow w_{y'} - \eta x$ update to demote y'

Why add ηx to w_y promote label y:

Before update $s(y) = < w_y^{old}, x >$

After update $s(y) = < w_y^{new}, x > = < w_y^{old} + \eta x, x >$

$= < w_y^{old}, x > + \eta < x, x >$

Note! $< x, x > = x^T x > 0$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

For $y' \neq y$

if $w_y^T x < w_{y'}^T x$ make a mistake

$w_y \leftarrow w_y + \eta x$ promote y

$w_{y'} \leftarrow w_{y'} - \eta x$ demote y'

Return $\boldsymbol{w}$

How to analyze this algorithm and simplify the update rules?

Prediction: $\text{argmax}_y \; w_y^T x$

Linear Separability with multiple classes

- Let's rewrite the equation

$$w_i^T x > w_j^T x \qquad \text{for all } j$$

  - Instead of having $w_1, w_2, w_3, \ldots w_K$, we want to represent the model using a single vector w

$$w^T \boxed{?} > w^T \boxed{?} \quad \text{for all j}$$

- How?

Change the input representation
Let's define $\phi(x, y)$, such that

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

multiple models v.s. multiple data points

Kesler construction

Assume we have a multi-class problem with K class and n features.

$$w_i^T x > w_j^T x \quad \forall \text{ j}$$

- models:

  $w_1, w_2, \ldots w_K, \qquad w_k \in R^n$

- Input:

  $x \in R^n$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

Kesler construction

Assume we have a multi-class problem with K class and n features.

$$w_i^T x > w_j^T x \quad \forall \text{ j}$$

- models:

  $w_1, w_2, \ldots w_K, \qquad w_k \in R^n$

- Input:

  $x \in R^n$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

- Only one model:

  $w \in R^{n \times K}$

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

Kesler construction

Assume we have a multi-class problem with K class and n features.

$$w_i^T x > w_j^T x \quad \forall j$$

- models:

  $w_1, w_2, \ldots w_K, \qquad w_k \in R^n$

- Input:

  $x \in R^n$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

- Only one model:

  $w \in R^{nK}$

- Define $\phi(x, y)$ for label y being associated to input x

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

x in y^{th} block; Zeros elsewhere

Kesler construction

Assume we have a multi-class problem with K class and n features.

$w_i^T x > w_i^T x \quad \forall$ i

- models:

  $w_1, w_2, \ldots w_K,$

  $w_k \in R^n$

- Input:

  $x \in R^n$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \qquad \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}$$

x in y^{th} block; Zeros elsewhere

$$w^T \phi(x, y) = w_y^T x$$

Kesler construction

Assume we have a multi-class problem with K class and n features.

binary classification problem

$$w^T \phi(x,i) > w^T \phi(x,j) \quad \forall\, j$$

$$\Rightarrow w^T [\phi(x,i) - \phi(x,j)] > 0 \quad \forall j$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK\times 1} \qquad [\phi(x,i) - \phi(x,j)] = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ -x \\ \vdots \\ 0_n \end{bmatrix}_{nK\times 1}$$

x in i^{th} block;

$-x$ in j^{th} block;

Linear Separability with multiple classes

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

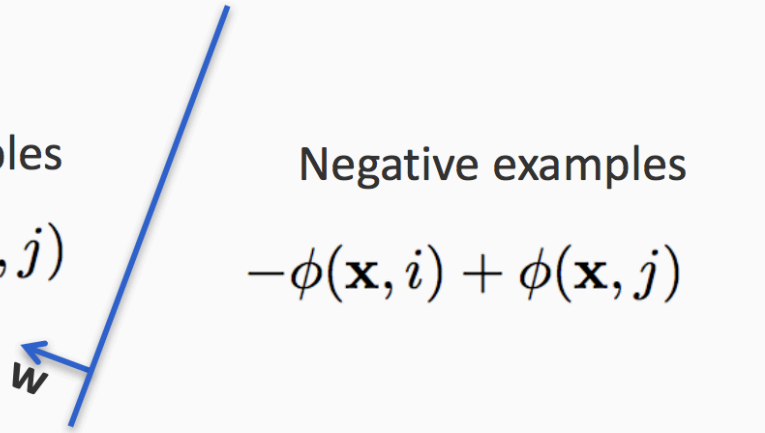
What we want

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall\, j$$

$$\Rightarrow w^T [\phi(x, i) - \phi(x, j)] > 0 \quad \forall j$$

For all example (x, y) with all other labels y' in dataset, w in nK dimension should linearly separate $\phi(x, i) - \phi(x, j)$ and $-[\phi(x, i) - \phi(x, j)]$

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

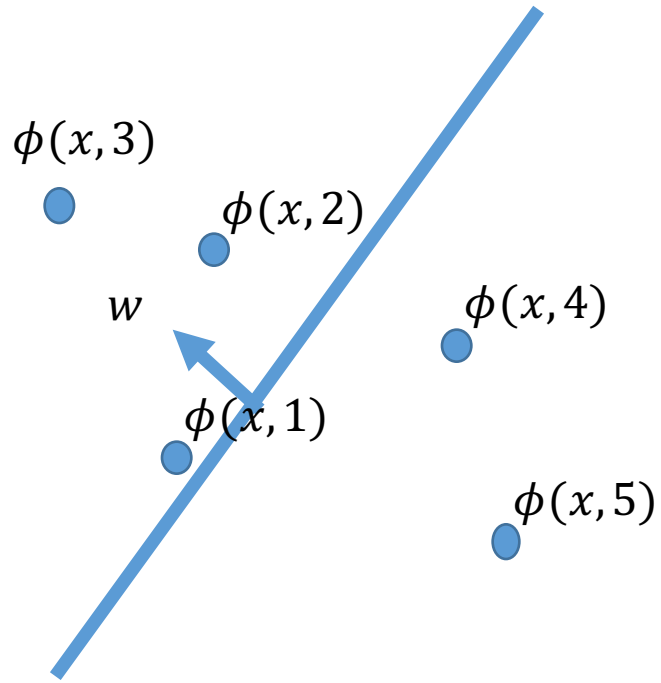


How can we predict?

$$\text{argmax}_y \; w^T \phi(x, y)$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \qquad \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}$$

For input an input x,
the model predict label is 3

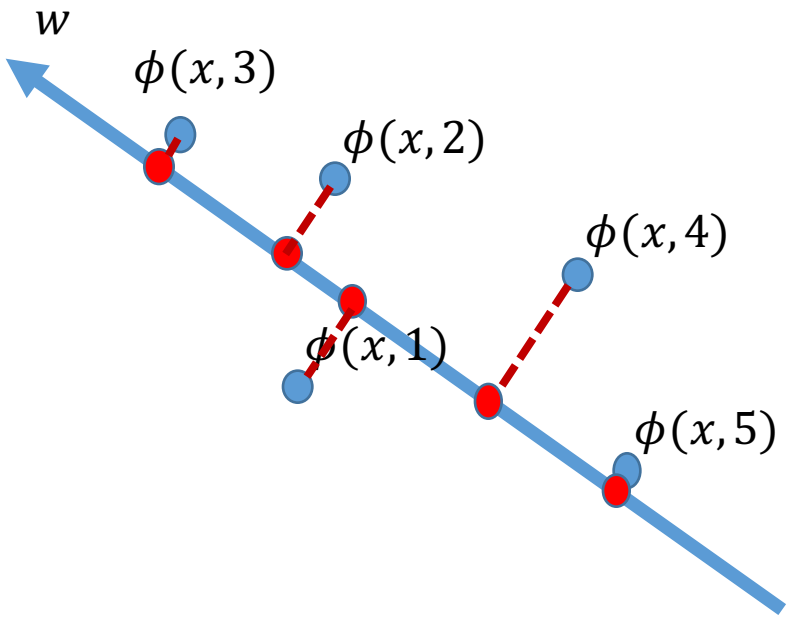


How can we predict?

$$\text{argmax}_y \; w^T \phi(x, y)$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \qquad \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}$$

For input an input x,
the model predict label is 3

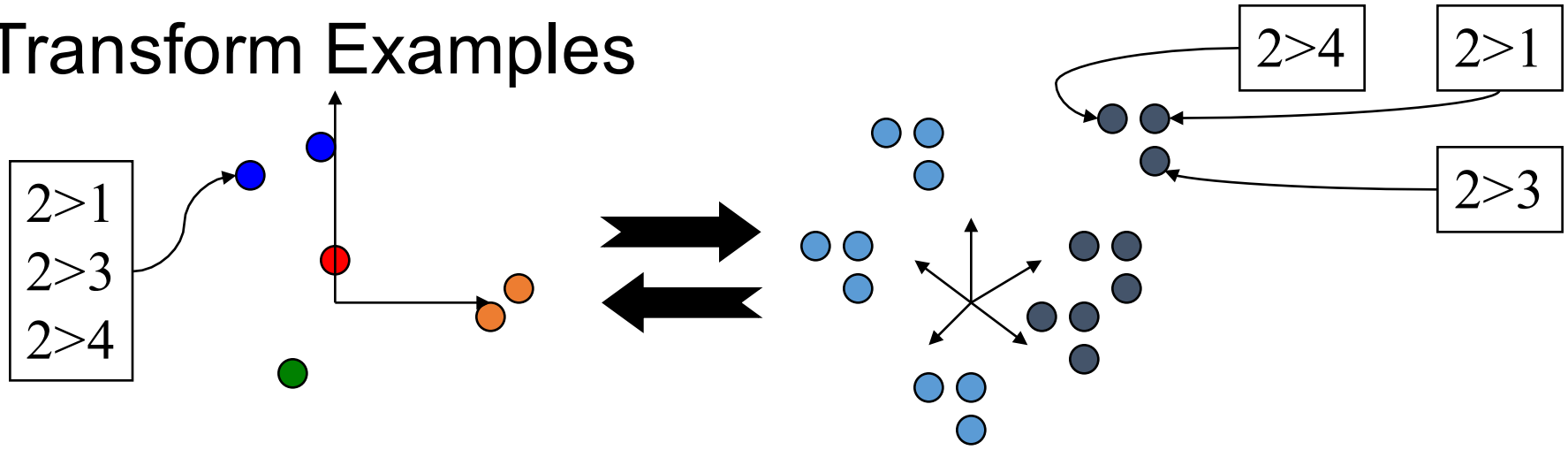


This is equivalent to
$\text{argmax}_{y \in \{1,2,\ldots K\}} \; w_y^T \; x$

Constraint Classification

- Goal: $w[\phi(x,i) - \phi(x,j)] \geq 0 \quad \forall j$
- Training:
  - For each example (x, i)
    - Update model if $w^T[\phi(x,i) - \phi(x,j)] < 0 \,, \forall\, j$

Transform Examples



A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

For $y' \neq y$

if $w^T[\phi(x, y) - \phi(x, y')] < 0$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta[\phi(x, y) - \phi(x, y')]$

Return $\boldsymbol{w}$

This needs $|D| \times K$ updates, do we need all of them?

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

An alternative training algorithm

- Goal: $w[\phi(x,i) - \phi(x,j)] \geq 0 \quad \forall j$

  $\Rightarrow w^T\phi(x,i) - \max_{j \neq i}[w^T\phi(x,i)] \geq 0$

  $\Rightarrow w^T\phi(x,i) - \max_{j}[w^T\phi(x,i)] \geq 0$

- Training:
  - For each example (x,i)
    - Find the prediction of the current model:

      $\hat{y} = \text{argmax}_j \; w^T\phi(x,j)$

    - Update model if $w^T[\phi(x,i) - \phi(x,\hat{y})] < 0 \,, \forall\, y'$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y')$

if $w^T[\phi(x, y) - \phi(x, \hat{y})] < 0$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta[\phi(x, y) - \phi(x, \hat{y})]$

Return $\boldsymbol{w}$

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y')$

~~if $w^T[\phi(x, y) - \phi(x, \hat{y})] < 0$~~

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta[\phi(x, y) - \phi(x, \hat{y})]$

Return $\boldsymbol{w}$

There are only two situations:

1. $\hat{y} = y$: $\phi(x, y) - \phi(x, \hat{y}) = 0$
2. $w^T[\phi(x, y) - \phi(x, \hat{y})] < 0$

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y')$

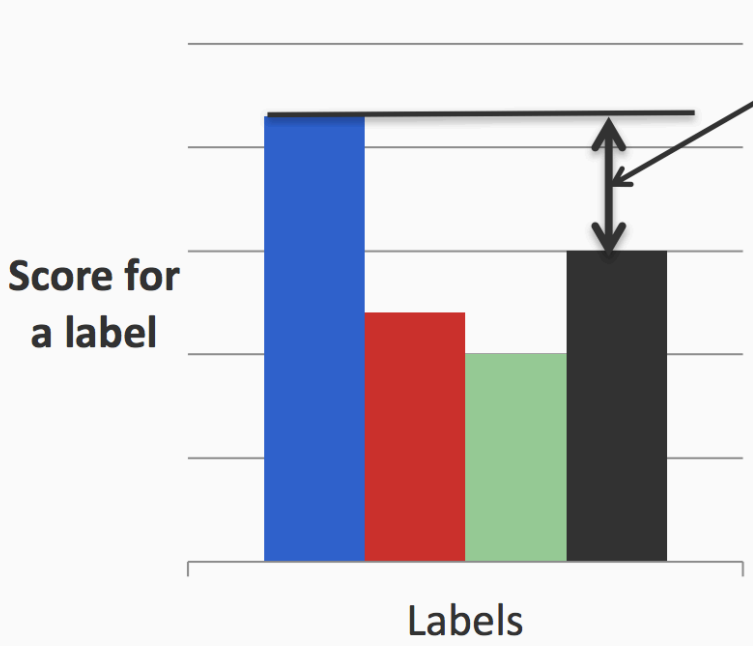
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \left[\phi(x, y) - \phi(x, \hat{y})\right]$

Return $\boldsymbol{w}$

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

Consider multiclass margin



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

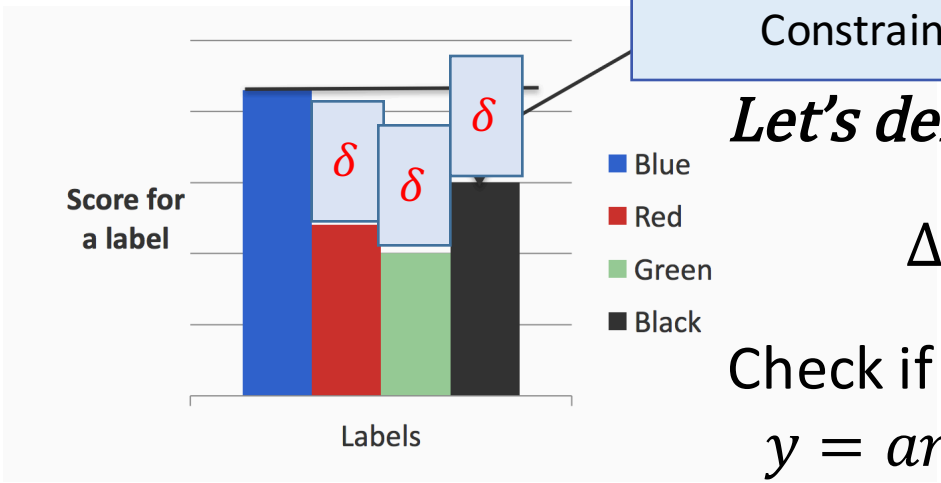
Marginal constraint classifier

- Goal: for every (x,y) in the training data set

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

$$\Rightarrow w^T\phi(x,y) - \max_{y \neq y'} w^T\phi(x,y') \geq \delta$$

$$\Rightarrow w^T\phi(x,i) - [\max_{y \neq y'} w^T\phi(x,j) + \delta] \geq 0$$



Constraints violated ⇒ need an update

Let's define:

$$\Delta(y,y') = \begin{cases} \delta & \text{if } y \neq y' \\ 0 & if\ y = y' \end{cases}$$

Check if

$$y = argmax_{y'} w^T\phi(x,y') + \Delta(y,y')$$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

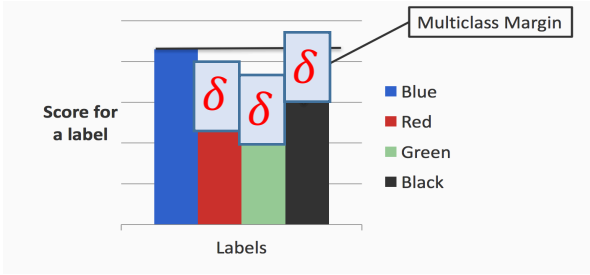
For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y') + \Delta(y, y')$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \left[\phi(x, y) - \phi(x, \hat{y})\right]$

Return $\boldsymbol{w}$



How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

Remarks

- This approach can be generalized to train a ranker; in fact, any output structure
  - We have preference over label assignments
  - E.g., rank search results, rank movies / products

A peek of a generalized Perceptron model

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y') + \Delta(y, y')$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \left[\phi(x, y) - \phi(x, \hat{y})\right]$

Return $\boldsymbol{w}$

Structured output

Structural prediction/Inference

Structural loss

Model update

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

Feb 1.

Recap: A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

For $y' \neq y$

if $w_y^T x < w_{y'}^T x$ make a mistake

$w_y \leftarrow w_y + \eta x$ promote y

$w_{y'} \leftarrow w_{y'} - \eta x$ demote y'

Return $\boldsymbol{w}$

Prediction: $\text{argmax}_y \; w_y^T x$

Recap: Kesler construction

Assume we have a multi-class problem with K class and n features.

$$w_i^T x > w_i^T x \quad \forall \text{ i}$$

- models:

  $w_1, w_2, \ldots w_K,$

  $w_k \in R^n$

- Input:

  $x \in R^n$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall\, j$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \qquad \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}$$

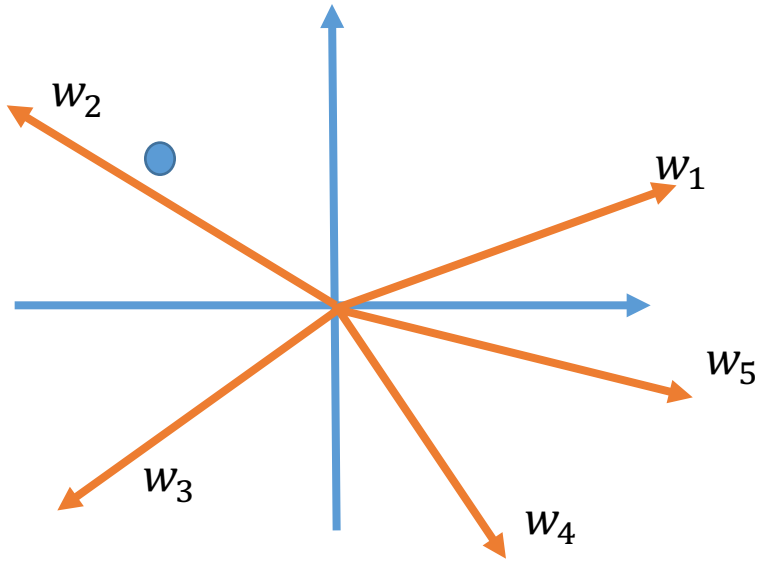
x in y^{th} block; Zeros elsewhere

$$w^T \phi(x, y) = w_y^T x$$

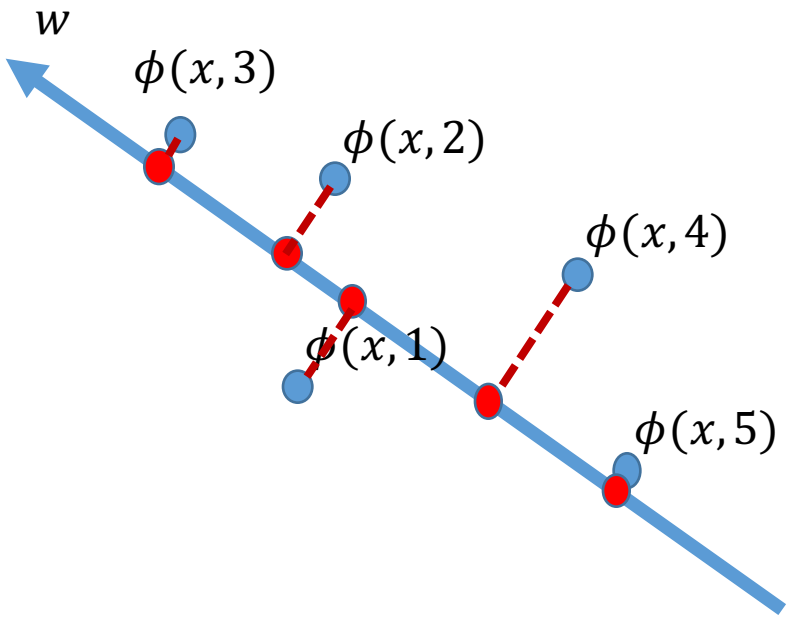
Geometric interpretation

features = n; # classes = K

In R^N space



In R^{NK} space



$\text{argmax}_{y\in\{1,2,\ldots K\}}\ w_y^T\ x$

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

Recap: A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_{y'} \; w^T \phi(x, y')$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \left[\phi(x, y) - \phi(x, \hat{y})\right]$

Return $\boldsymbol{w}$

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$

Multi-category to Constraint Classification

- Multiclass
  - (x, A) $\Rightarrow$ (x, (A>B, A>C, A>D))
- Multilabel
  - (x, (A, B)) $\Rightarrow$ (x, ((A>C, A>D, B>C, B>D))
- Label Ranking
  - (x, (5>4>3>2>1)) $\Rightarrow$ (x, ((5>4, 4>3, 3>2, 2>1))

Generalized constraint classifiers

- In all cases, we have examples (x,y) with $y \in \mathbf{S_k}$
- Where $\mathbf{S_k}$: partial order over class labels {1,...,k}
  - defines "*preference*" relation (>) for class labeling
- Consequently, the Constraint Classifier is: h: $\mathbf{X} \rightarrow \mathbf{S_k}$
  - h(x) is a partial order
  - h(x) is *consistent* with y if $(i<j) \in y \rightarrow (i<j) \in h(x)$

Just like in the multiclass we learn one $w_i \in R^n$ for each label, the same is done for multi-label and ranking. The weight vectors are updated according with the requirements from $y \in S_k$

Multi-category to Constraint Classification

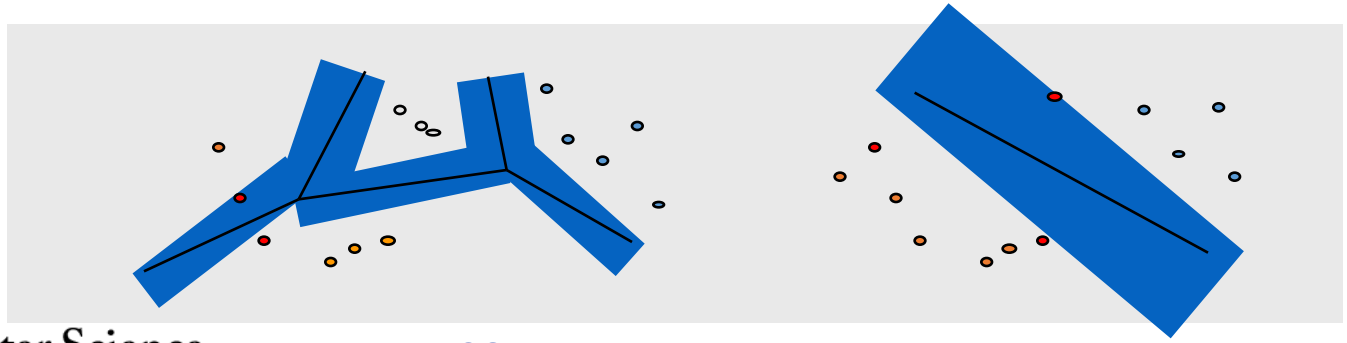
Solving structured prediction problems by ranking algorithms

- Multiclass
  - (x, A) $\Rightarrow$ (x, (A>B, A>C, A>D))
- Multilabel
  - (x, (A, B)) $\Rightarrow$ (x, ((A>C, A>D, B>C, B>D))
- Label Ranking
  - (x, (5>4>3>2>1)) $\Rightarrow$ (x, ((5>4, 4>3, 3>2, 2>1))

$y \in \mathbf{S}_k$ $\quad$ $h: \mathbf{X} \rightarrow \mathbf{S}_k$

Properties of Construction (Zimak et. al 2002, 2003)

- Can learn *any* argmax $w_i.x$ function (even when i isn't linearly separable from the union of the others)
- Can use *any* algorithm to find linear separation
  - Perceptron Algorithm
    - *ultraconservative online algorithm* [Crammer, Singer 2001]
  - Winnow Algorithm
    - *multiclass winnow* [Masterharm 2000]
- Defines a *multiclass margin* by binary margin in R^{KN}
  - multiclass SVM [Crammer, Singer 2001]

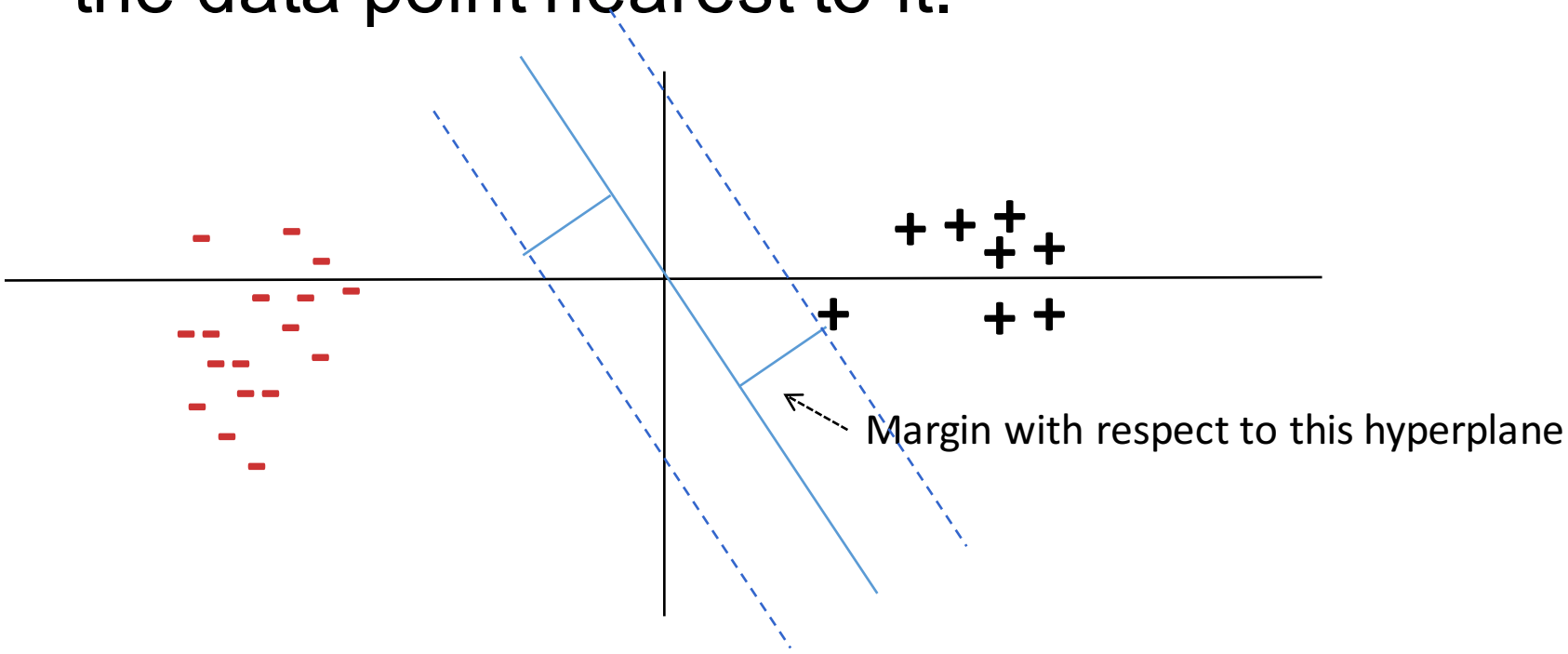


This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler's construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression

Recall: Margin for binary classifiers

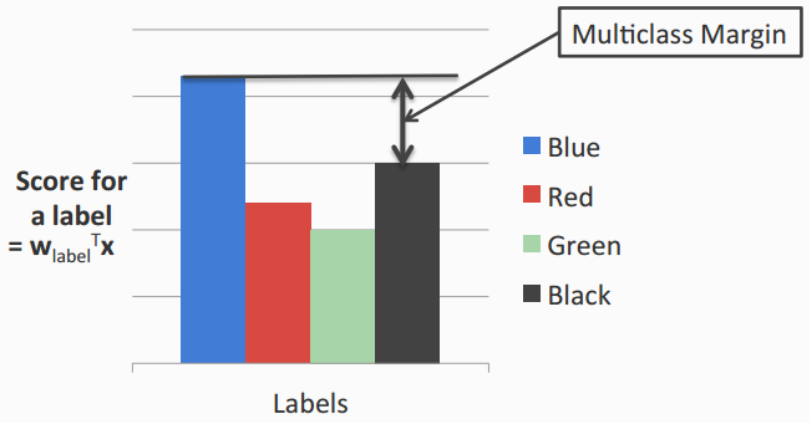
- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Multi-class SVM

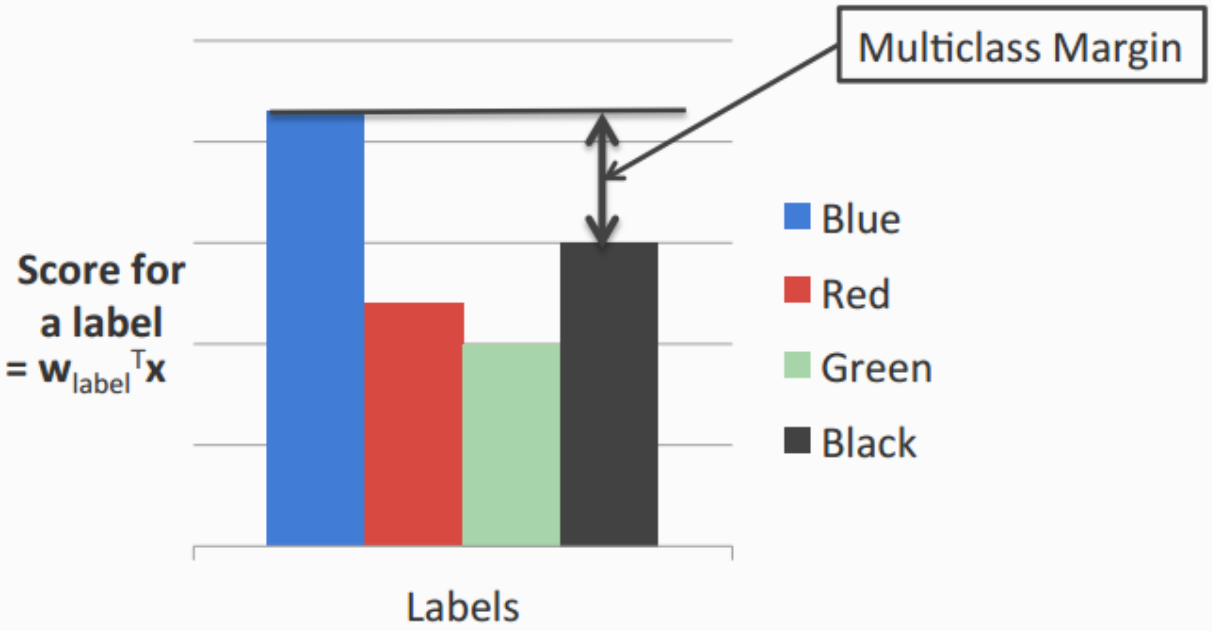
- In a risk minimization framework
- Goal: $\mathrm{D} = \{(\mathrm{x_i}, \mathrm{y_i})\}_{i=1}^{N}$
  1. $w_{y_i}^T x_i > w_{y'}^T x_i$ for all i, y'
  2. Maximizing the margin

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)

- 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

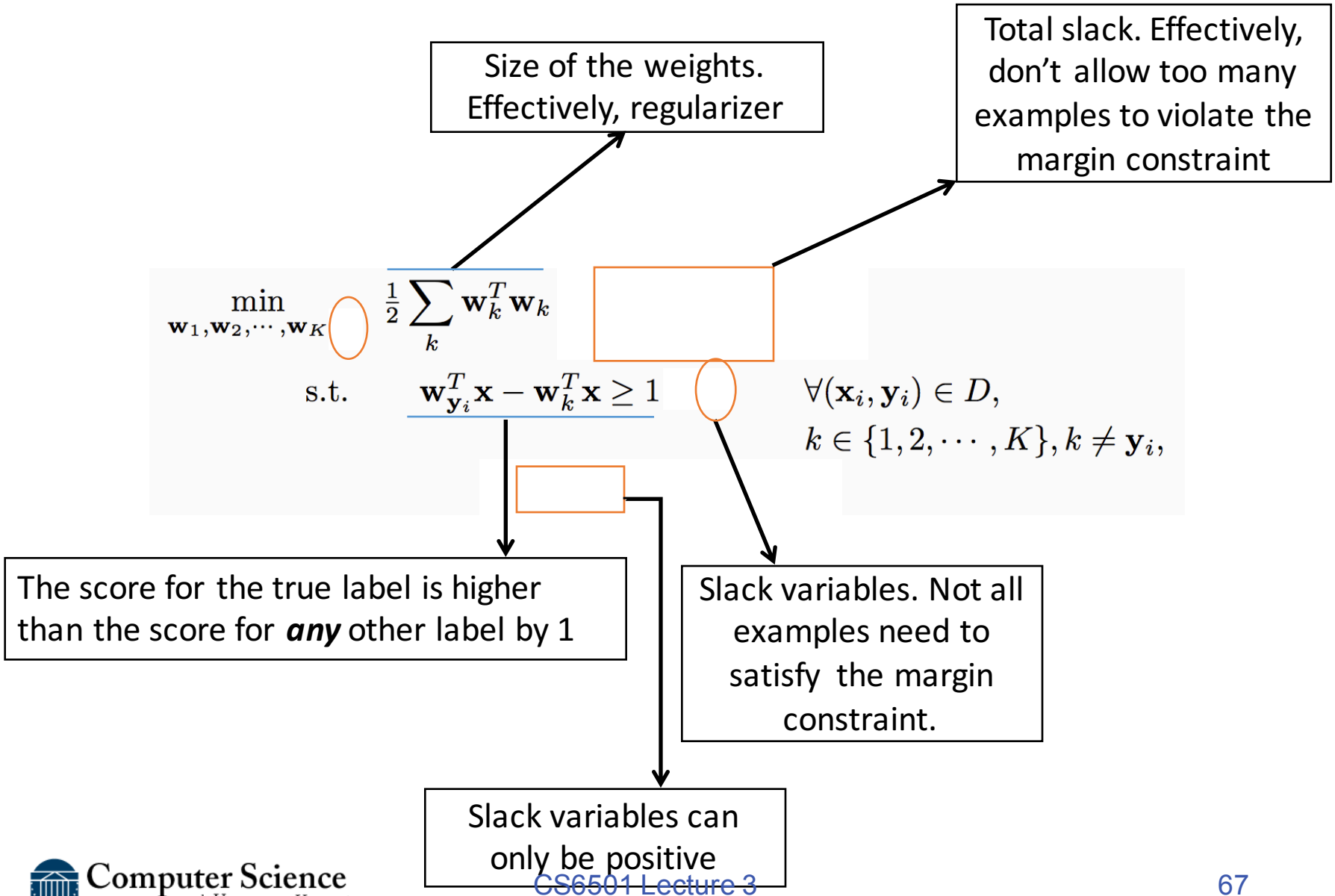
$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

Size of the weights. Effectively, regularizer

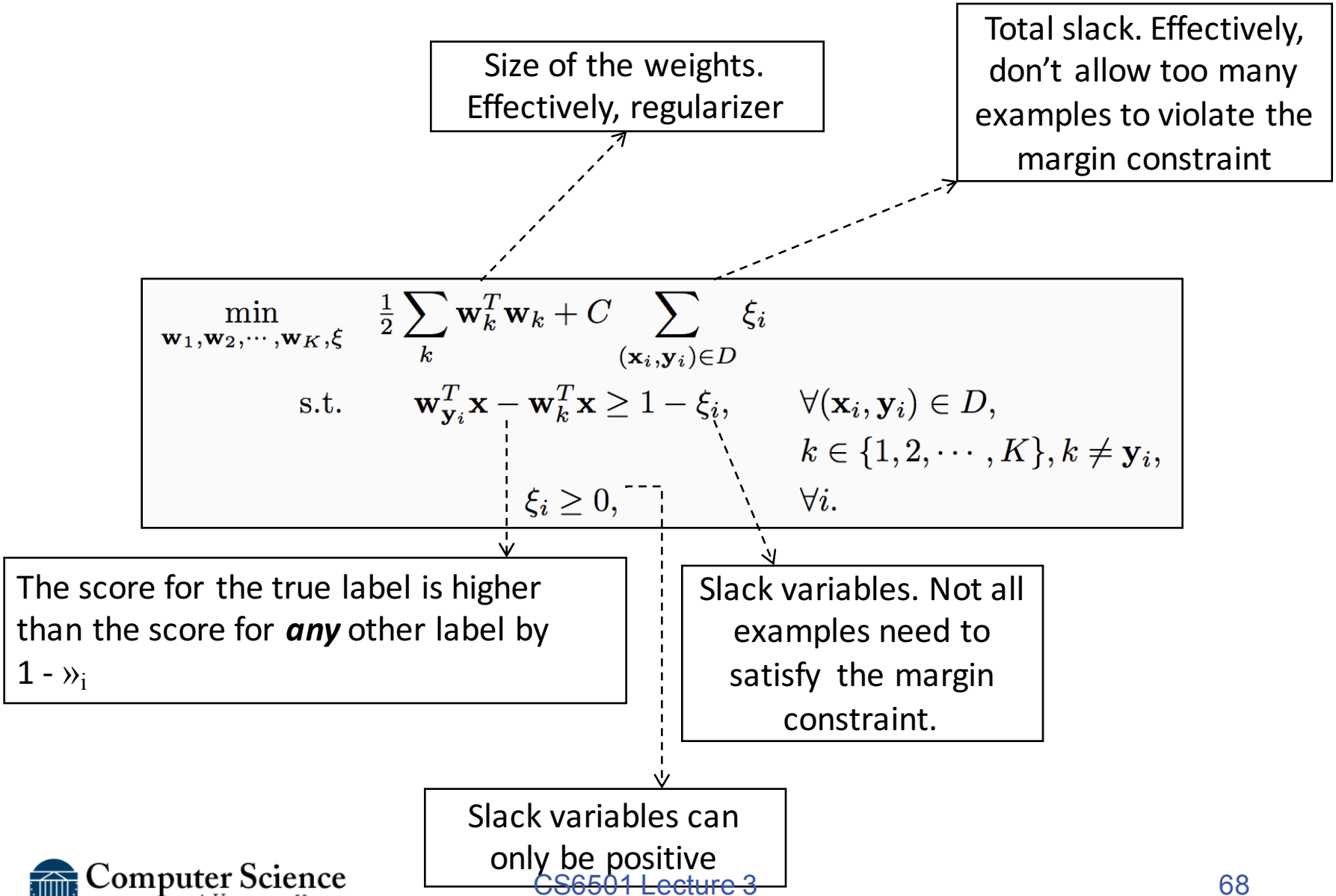
$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k$$
$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 \quad , \quad \forall(\mathbf{x}_i,\mathbf{y}_i) \in D,$$
$$k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i,$$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

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

Multiclass SVM: General case



Multiclass SVM: General case



Recap: An alternative SVM formulation

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

$$\text{s.t} \quad \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) \geq 1 - \xi_i;\ \xi_i \geq 0 \qquad \forall i$$

- Rewrite the constraints:

$$\xi_i \geq 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b);\ \xi_i \geq 0 \qquad \forall i$$

- In the optimum, $\xi_i = \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b))$
- Soft SVM can be rewritten as:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boxed{\boldsymbol{w}^T\boldsymbol{w}} + C\boxed{\sum_i \max(0,\ 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b))}$$

Regularization term

Empirical loss

Rewrite it as unconstraint problem

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

Let's define:

$$\Delta(y,y') = \begin{cases} \delta & \text{if } y \neq y' \\ 0 & if\ y = y' \end{cases}$$

$$\min_w \frac{1}{2}\sum_k w_k^T w_k + C\ \sum_i (\max_k(\Delta(y_i,k) + w_k^T x) - w_{y_i}^T x\ )$$

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

Exercise!

- Write down SGD for multiclass SVM

- Write down multiclas SVM with Kesler construction

This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler's construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression

Recall: (binary) logistic regression

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \log(1 + e^{-\mathrm{y_i}(\mathbf{w}^T\mathbf{x_i})})$$

Assume labels are generated using the following probability distribution:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

$$P(y = -1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}}$$

(multi-class) log-linear model

- Assumption:

Partition function

$$P(y|x,w) = \frac{\exp\left(w_y^T x\right)}{\sum_{y' \in \{1,2,\ldots K\}} \exp\left(w_{y'}^T x\right)}$$

- This is a valid probability assumption. Why?
- Another way to write this (with Kesler construction) is

This often called soft-max

$$P(y|x,w) = \frac{\exp\left(w^T \phi(x,y)\right)}{\sum_{y' \in \{1,2,\ldots K\}} \exp\left(w^T \phi(x,y')\right)}$$

Softmax

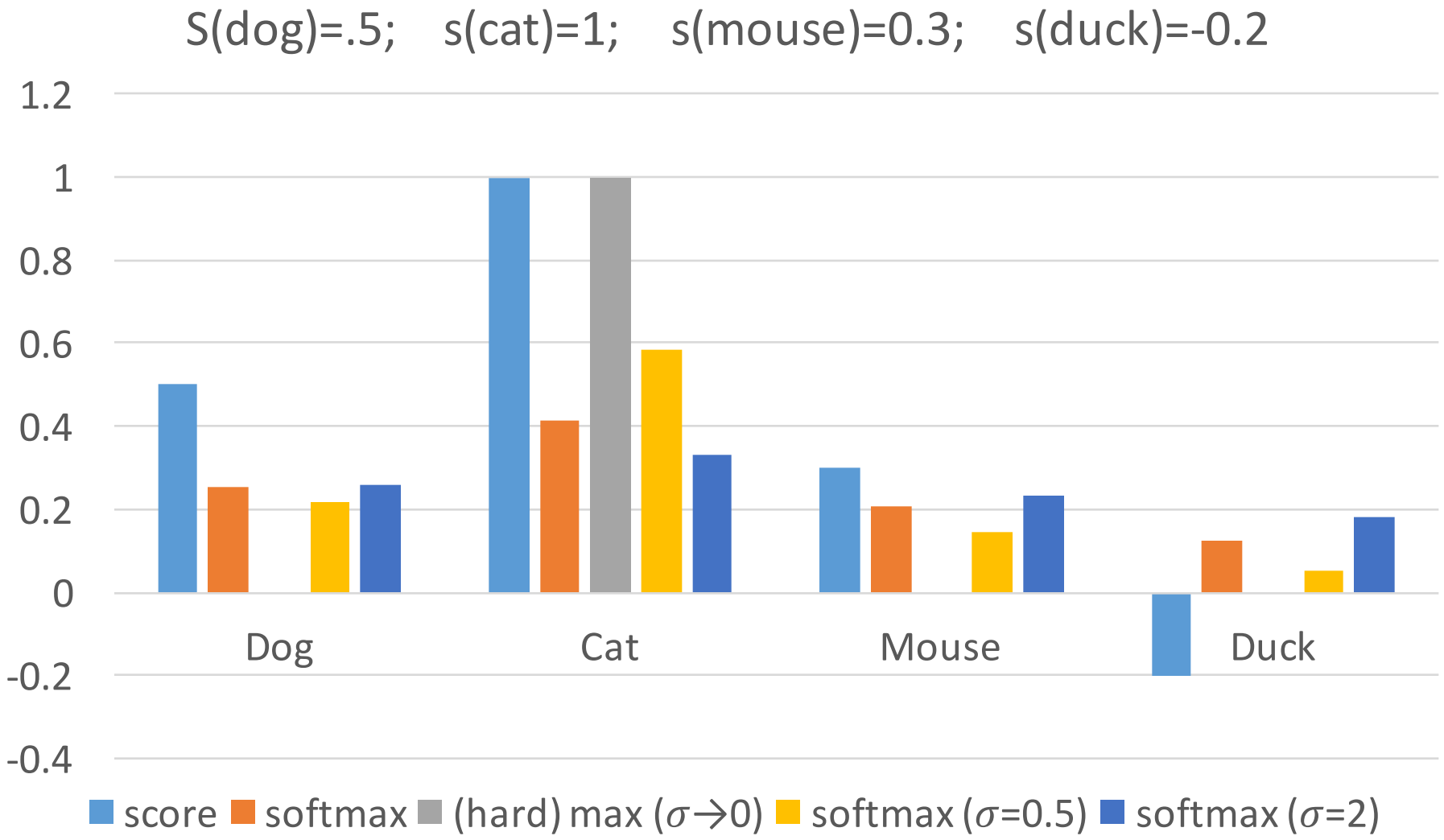
- Softmax: let s(y) be the score for output y
  here s(y)=$w^T \phi(x, y)$ (or $w_y^T x$) but it can be computed by other metric.

$$P(y) = \frac{\exp(s(y))}{\sum_{y' \in \{1,2,\ldots K\}} \exp(s(y))}$$

- We can control the peakedness of the distribution

$$P(y|\sigma) = \frac{\exp(s(y)/\sigma)}{\sum_{y' \in \{1,2,\ldots K\}} \exp(s(y/\sigma))}$$

Example


S(dog)=.5; s(cat)=1; s(mouse)=0.3; s(duck)=-0.2
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
Dog
Cat
Mouse
Duck
score
softmax
(hard) max (σ→0)
softmax (σ=0.5)
softmax (σ=2)

Log linear model

$$P(y|x,w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1,2,\ldots K\}} \exp(w_{y'}^T x)}$$

$$\log P(y|x,w) = \log(\exp(w_y^T x)) - \log(\sum_{y' \in \{1,2,\ldots K\}} \exp(w_{y'}^T x))$$

$$= w_y^T x - \log(\sum_{y' \in \{1,2,\ldots K\}} \exp(w_{y'}^T x))$$

Linear function

Except this term

Note:

$$p(y) \propto \exp(\theta^{\mathsf{T}} \mathbf{f}(y))$$

Maximum log-likelihood estimation

- Training can be done by maximum log-likelihood estimation i.e. $\max_{w} \log P(D|w)$

$$D=\{(x_i, y_i)\}$$

$$P(D|w) = \Pi_i \frac{\exp\left(w_{y_i}^T x_i\right)}{\sum_{y' \in \{1,2,\ldots K\}} \exp\left(w_{y'}^T x_i\right)}$$

$$\log P(D|w) = \sum_i [w_{y_i}^T x_i - \log \sum_{y' \in \{1,2,\ldots K\}} \exp(w_{y'}^T x_i)]$$

Maximum a posteriori

$D=\{(x_i, y_i)\}$

Can you use Kesler construction to rewrite this formulation?

$$P(w|D) \propto P(w)P(D|w)$$

$$max_w \; -\frac{1}{2}\sum_y w_y^T w_y + C\sum_i [w_{y_i}^T x_i - \log \sum_{y' \in \{1,2,\ldots K\}} \exp(w_{y'}^T x_i)]$$

or

$$\min_w \; \frac{1}{2}\sum_y w_y^T w_y + C\sum_i [\log \sum_{y' \in \{1,2,\ldots K\}} \exp\left(w_{y'}^T x_i\right) - w_{y_i}^T x_i]$$

Comparisons

- Multi-class SVM:
  $\min_w \frac{1}{2} \sum_k w_k^T w_k + C \sum_i (\max_k (\Delta(y_i, k) + w_k^T x - w_{y_i}^T x))$

- Log-linear model w/ MAP (multi-class)
  $\min_w \frac{1}{2} \sum_k w_k^T w_k + C \sum_i [\log \sum_{k \in \{1,2,\ldots K\}} \exp(w_k^T x_i) - w_{y_i}^T x_i]$

- Binary SVM:
  $\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C \sum_i \max(0, 1 - \mathrm{y_i}(\mathbf{w}^{\mathrm{T}} \mathbf{x_i}))$
- Log-linear mode (logistic regression)
  $\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C \sum_i \log(1 + e^{-\mathrm{y_i}(\mathbf{w}^{\mathrm{T}} \mathbf{x_i})})$