Midterms

Count	151
Minimum Value	21.00
Maximum Value	96.00
Range	75.00
Average	72.33
Median	75.50
Standard Deviation	13.43
Variance	180.30

PAC Learning

Count	151
Minimum Value	3.00
Maximum Value	25.00
Range	22.00
Average	19.74
Median	21.00
Standard Deviation	4.90
Variance	24.04

SVM

Count	151
Minimum Value	2.50
Maximum Value	25.00
Range	22.50
Average	16.65
Median	18.00
Standard Deviation	5.92
Variance	35.07

Kernels+Boost

Count	151
Minimum Value	5.00
Maximum Value	25.00
Range	20.00
Average	16.82
Median	17.50
Standard Deviation	3.63
Variance	13.19

Decision Trees

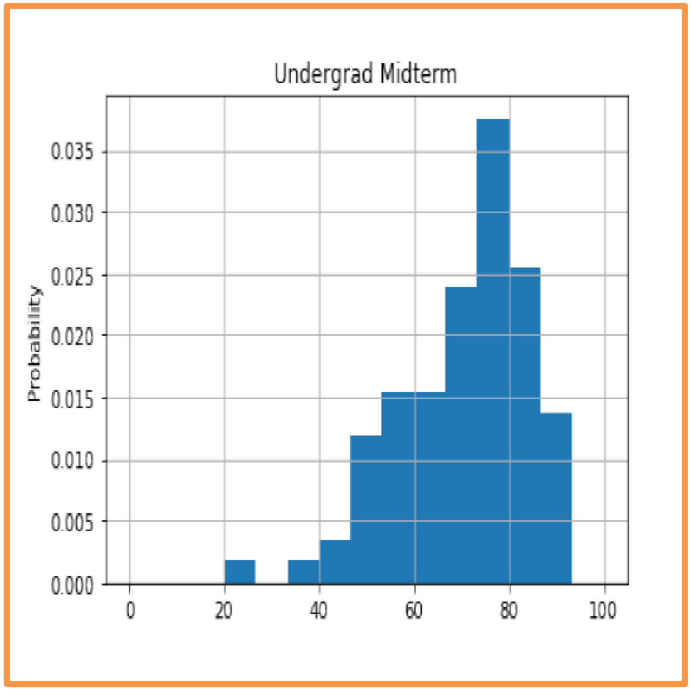
Count	151
Minimum Value	5.00
Maximum Value	24.00
Range	19.00
Average	18.13
Median	18.50
Standard Deviation	4.07
Variance	16.54

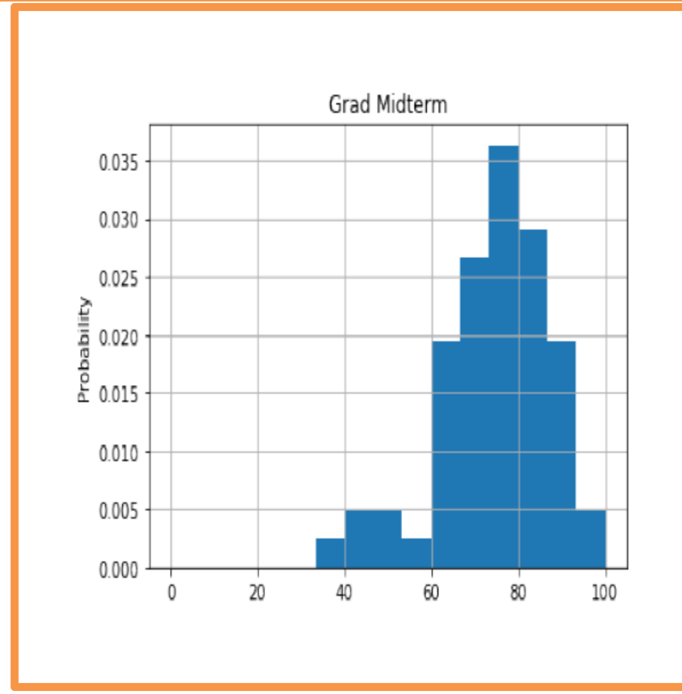
Midterms

Grades are on a curve

Will be available at the TA sessions this week

Projects feedback has been sent. Recall that this is 25% of your grade!





Midterm Level = U	
count	88.000000
mean	70.539773
std	13.822400
min	22.000000
25%	61.375000
50%	74.000000
75%	80.250000
max	92.000000

Midterm Level = ALL	
count	150.000000
mean	72.173333
std	13.442643
min	22.000000
25%	63.750000
50%	75.000000
75%	82.000000
max	96.000000

Midterm Level = G	
count	62.000000
mean	74.491935
std	12.632735
min	37.500000
25%	68.125000
50%	75.750000
75%	83.000000
max	96.000000

Classification

- So far we focused on Binary Classification
- For linear models:
  - Perceptron, Winnow, SVM, GD, SGD
- The prediction is simple:
  - Given an example x,
  - Prediction = $\text{sgn}(w^T x)$
  - Where w is the learned model
- The output is a single bit

Multi-Categorical Output Tasks

- Multi-class Classification ($y \in \{1,\ldots,K\}$)
  - character recognition (‘6’)
  - document classification (‘homepage’)
- Multi-label Classification ($y \subseteq \{1,\ldots,K\}$)
  - document classification (‘(homepage,facultypage)’)
- Category Ranking ($y \in \pi(K)$)
  - user preference (‘(love > like > hate)’)
  - document classification (‘hompage > facultypage > sports’)
- Hierarchical Classification ($y \subseteq \{1,..,K\}$)
  - cohere with class hierarchy
  - place document into index where ‘soccer’ is-a ‘sport’

Setting

- Learning:
  - Given a data set $D = \{(x_i, y_i)\}_1^m$
  - Where $x_i \in R^n$, $y_i \in \{1,2,\ldots,k\}$.
- Prediction (inference):
  - Given an example **x**, and a learned function (model),
  - Output a single class labels **y**.

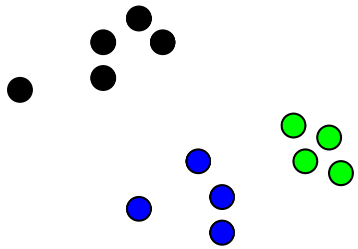
Binary to Multiclass

- Most schemes for multiclass classification work by reducing the problem to that of binary classification.
- There are multiple ways to decompose the multiclass prediction into multiple binary decisions
  - One-vs-all
  - All-vs-all
  - Error correcting codes
- We will then talk about a more general scheme:
  - Constraint Classification
- It can be used to model other non-binary classification schemes and leads to Structured Prediction.

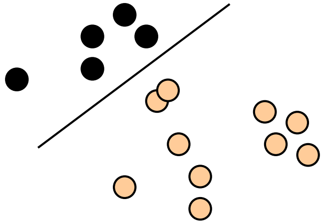
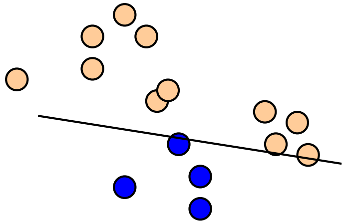
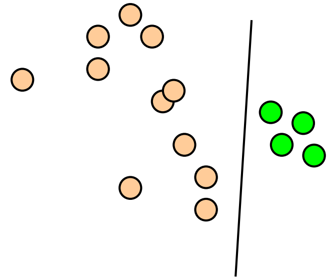
One-Vs-All

- Assumption: Each class can be separated from all the rest using a binary classifier in the hypothesis space.
- Learning: Decomposed to learning k independent binary classifiers, one for each class label.
- Learning:
  - Let D be the set of training examples.
  - $\forall$ label l, construct a binary classification problem as follows:
    - Positive examples: Elements of D with label l
    - Negative examples: All other elements of D
  - This is a binary learning problem that we can solve, producing k binary classifiers $w_1, w_2, \ldots w_k$
- Decision: Winner Takes All (WTA):
  - $f(x) = \text{argmax}_i\ w_i^T x$

Solving MultiClass with 1vs All learning



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

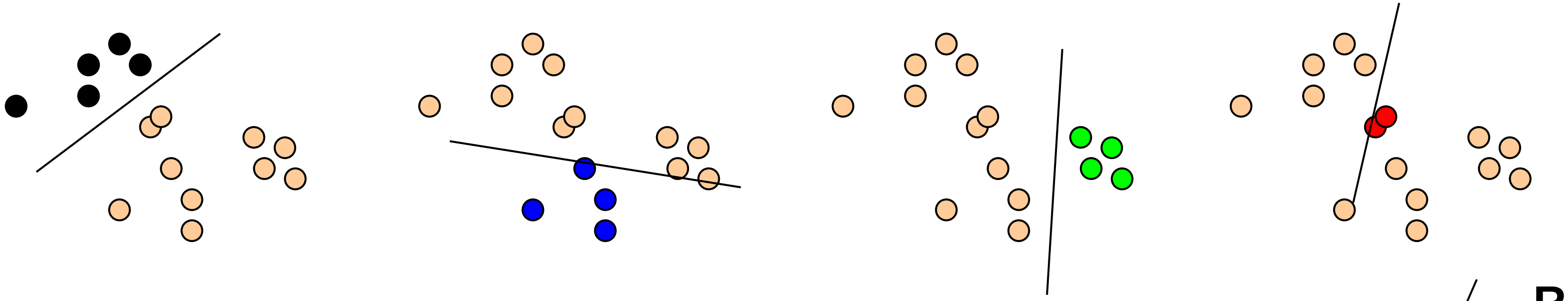




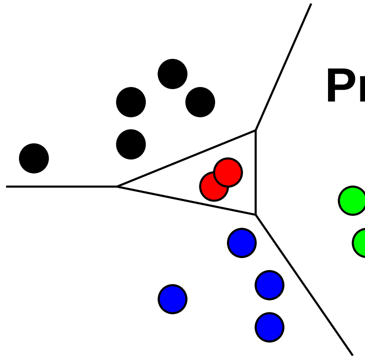
- Not always possible to learn
- No theoretical justification
  - Need to make sure the range of all classifiers is the same
- (unless the problem is easy)

Learning via One-Versus-All (OvA) Assumption

- Find $v_r, v_b, v_g, v_y \in \mathbf{R}^n$ such that
  - $v_r.x > 0$ iff y = red ⊗
  - $v_b.x > 0$ iff y = blue √
  - $v_g.x > 0$ iff y = green √
  - $v_y.x > 0$ iff y = yellow √
- **Classification: $f(x) = \text{argmax}_i\ v_i\ x$**

$\mathbf{H} = \mathbf{R}^{nk}$



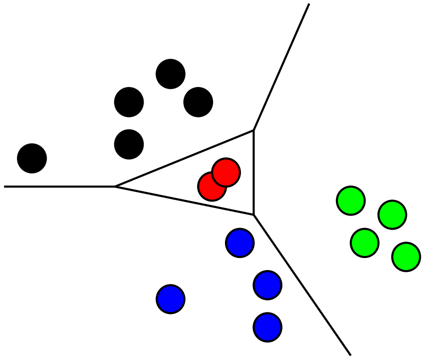


All-Vs-All

- Assumption: There is a separation between every pair of classes using a binary classifier in the hypothesis space.
- Learning: Decomposed to learning [k choose 2] ~ k^2 independent binary classifiers, one corresponding to each pair of class labels. For the pair (i, j):
  - Positive example: all exampels with label i
  - Negative examples: all examples with label j
- Decision: More involved, since output of binary classifier may not cohere. Each label gets k-1 votes.
- Decision Options:
  - 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 .

Learning via All-Verses-All (AvA) Assumption

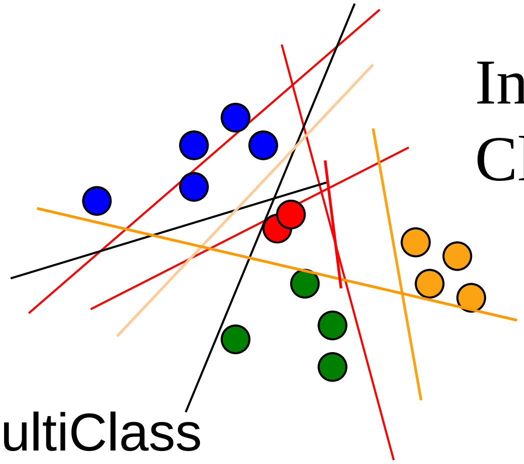
- Find $v_{rb}, v_{rg}, v_{ry}, v_{bg}, v_{by}, v_{gy} \in \mathbf{R}^d$ such that
  - $v_{rb}.x > 0$ if y = red
    < 0 if y = blue
  - $v_{rg}.x > 0$ if y = red
    < 0 if y = green
  - ... (for all pairs)



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

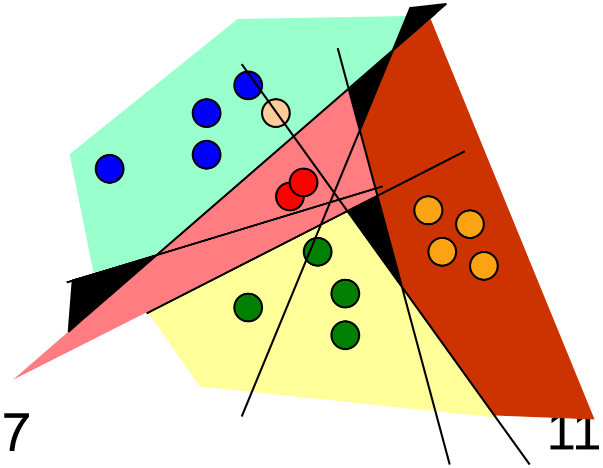
$\mathbf{H} = \mathbf{R}^{kkn}$

How to classify?



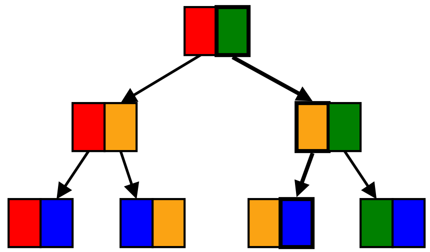
Individual Classifiers

Decision Regions

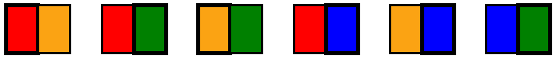


Classifying with AvA

Tournament



Majority Vote



1 red, 2 yellow, 2 green

→ ?

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

One-vs-All vs. All vs. All

- Assume m examples, k class labels.
  - For simplicity, say, m/k in each.
- One vs. All:
  - classifier f_i: m/k (+) and (k-1)m/k (-)
  - Decision:
  - Evaluate k linear classifiers and do Winner Takes All (WTA):
  - $$f(x) = \text{argmax}_i\, f_i(x) = \text{argmax}_i\, w_i^T x$$
- All vs. All:
  - Classifier f_{ij}: m/k (+) and m/k (-)
  - More expressivity, but less examples to learn from.
  - Decision:
  - Evaluate k^2 linear classifiers; decision sometimes unstable.
- What type of learning methods would prefer All vs. All (efficiency-wise)?

(Think about Dual/Primal)

Error Correcting Codes Decomposition

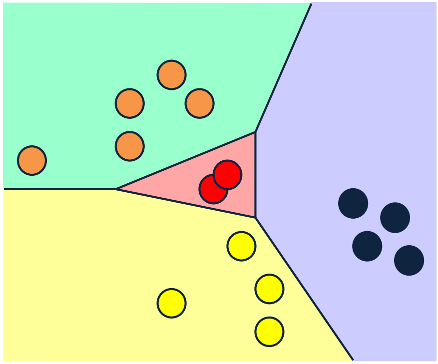
- 1-vs-all uses k classifiers for k labels; can you use only $\log_2 k$?
- Reduce the multi-class classification to random binary problems.
  - Choose a "code word" for each label.
  - K=8: all we need is 3 bits, three classifiers
- Rows: An encoding of each class (k rows)
- Columns: L dichotomies of the data, each corresponds to a new classification problem
- Extreme cases:
  - 1-vs-all: k rows, k columns
  - k rows $\log_2 k$ columns
- Each training example is mapped to one example per column
  - (x,3) → {(x,P1), +; (x,P2), -; (x,P3), -; (x,P4), +}
- To classify a new example x:
  - Evaluate hypothesis on the 4 binary problems {(x,P1) , (x,P2), (x,P3), (x,P4),}
  - Choose label that is most consistent with the results.
    - Use Hamming distance (bit-wise distance)
- Nice theoretical results as a function of the performance of the P_i s (depending on code & size)
- Potential Problems?

Label	P1	P2	P3	P4
1	-	+	-	+
2	-	+	+	-
3	+	-	-	+
4	+	-	+	+
k	-	+	-	-

Can you separate any dichotomy?

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 $\Rightarrow$ poor performance
  - Difficult local problems
  - Irrelevant local problems

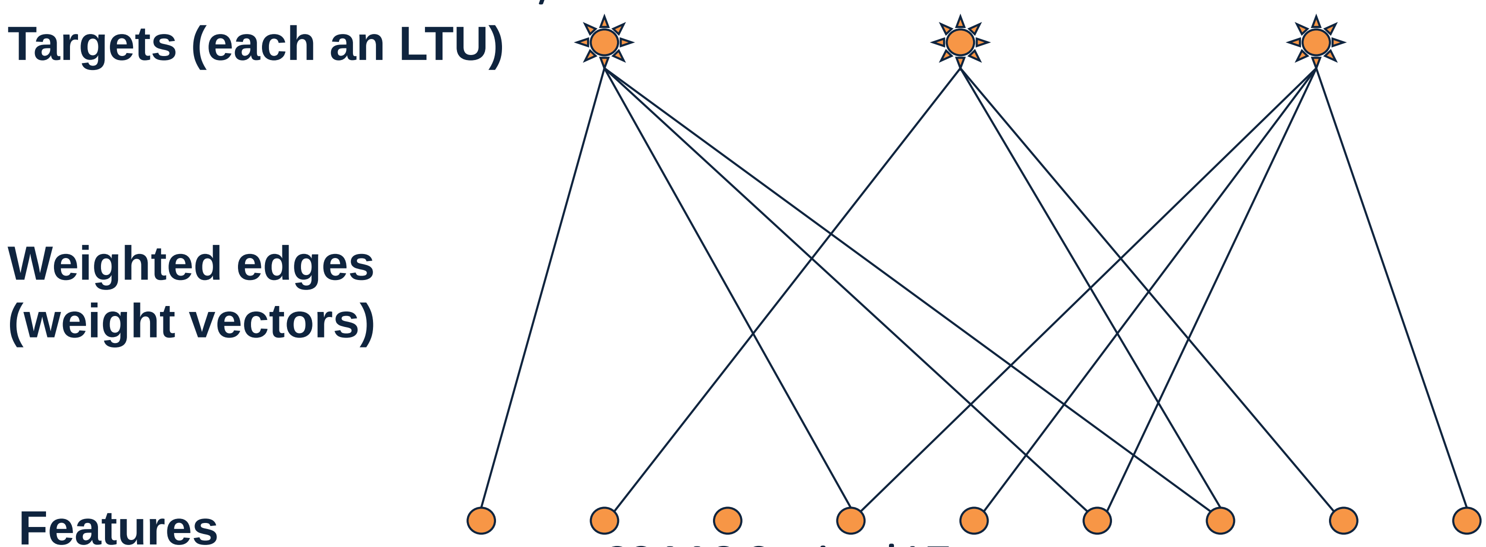
- Especially true for Error Correcting Output Codes
  - Another (class of) decomposition
  - Difficulty: how to make sure that the resulting problems are separable.

- Efficiency: e.g., All vs. All vs. One vs. All
- Former has advantage when working with the dual space.

- Not clear how to generalize multi-class to problems with a very large # of output variables.

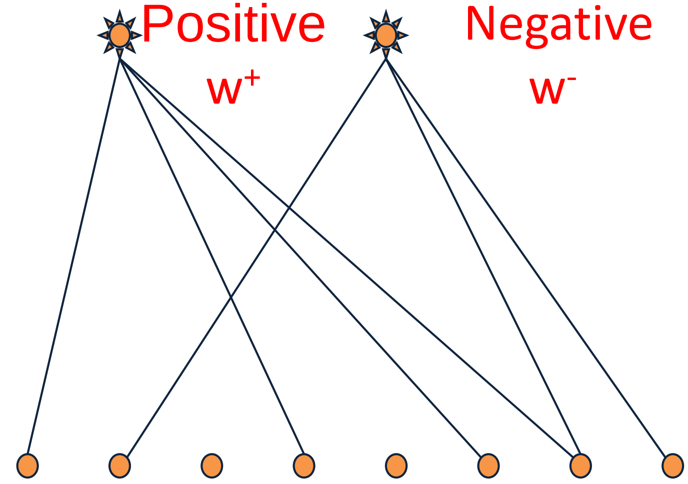
1 Vs All: Learning Architecture

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



Recall: Winnow's Extensions

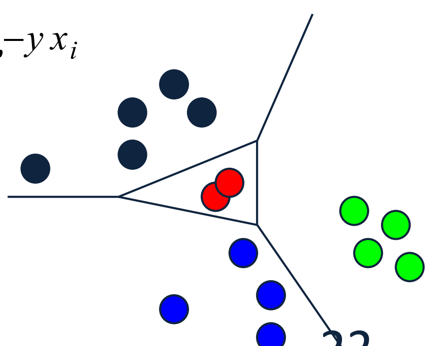
- Winnow learns monotone Boolean functions
- We extended to general Boolean functions via
- "Balanced Winnow"
  - 2 weights per variable;
  - Decision: using the "effective weight",
    the difference between w^+ and w^-
  - This is equivalent to the Winner take all decision
  - Learning: In principle,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



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

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

We need a "global" learning approach



Extending Balanced

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

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

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

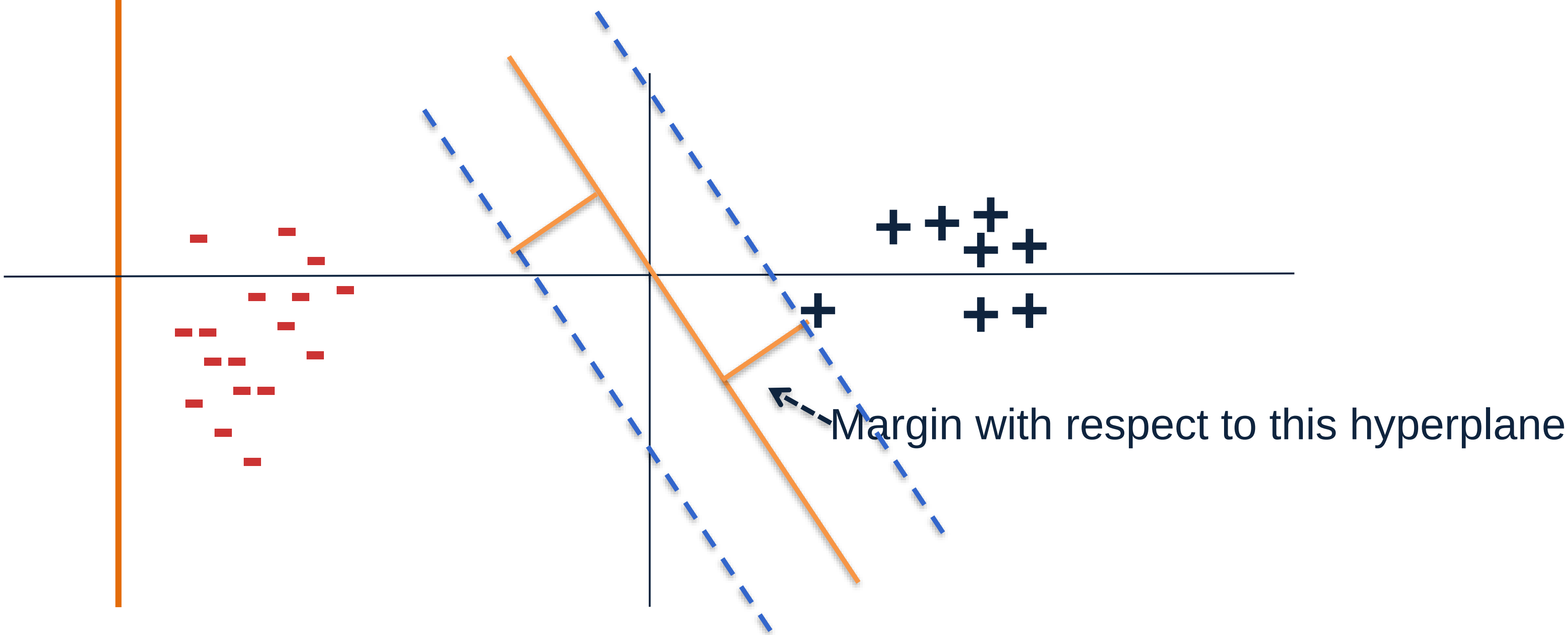
- Can this be generalized to the case of k labels, k >2?

Where are we?

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

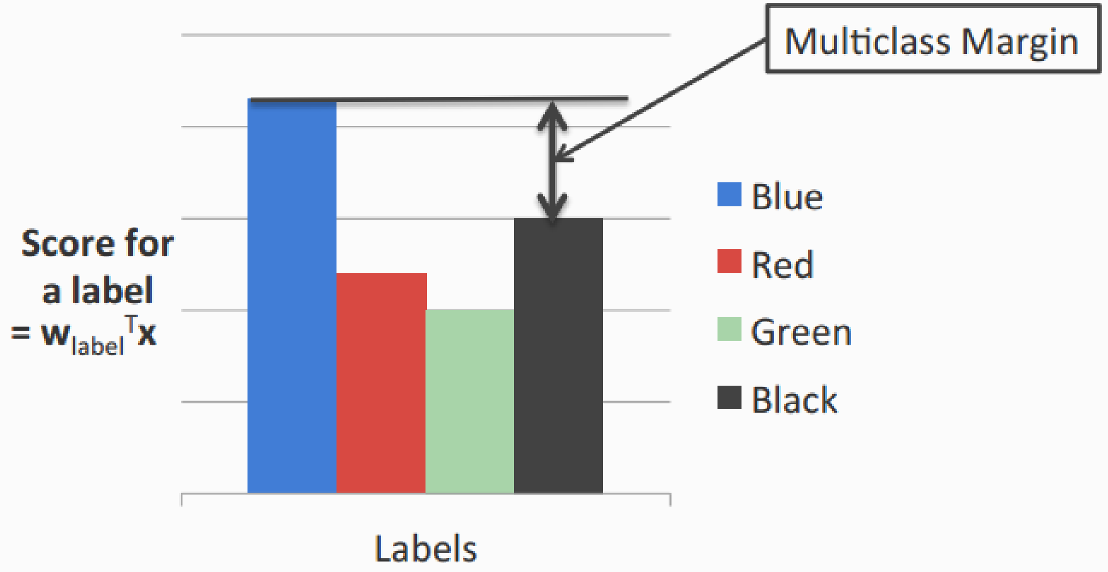
Recall: Margin for binary classifiers

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



Multiclass Margin

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



Multiclass SVM (Intuition)

- Recall: Binary SVM
  - Maximize margin
  - Equivalently,

    Minimize norm of weight vector, while keeping the closest points to the hyperplane with 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 weight vectors while making sure that the true label scores 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

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

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

Multiclass SVM: General case

Size of the weights. Effectively, regularizer

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

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

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

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

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

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. Effectively, don't allow too many examples to violate the margin constraint

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

$$\begin{aligned} \text{s.t.} \quad & \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 - \xi_i, \quad && \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \\ & && k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i, \\ & \xi_i \geq 0, && \forall i. \end{aligned}$$

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

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

Slack variables can only be positive

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 "global" 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
- Why does it work?
  - Why is this the "right" definition of multiclass margin?
- A theoretical justification, along with extensions to other algorithms beyond SVM is given by "Constraint Classification"
  - Applies also to multi-label problems, ranking problems, etc.
  - [Dav Zimak; with D. Roth and S. Har-Peled]

Constraint Classification

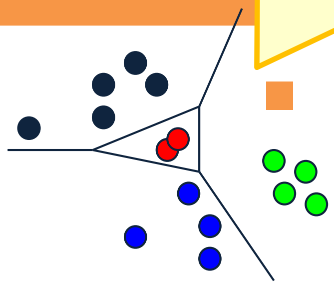
- The examples we give the learner are pairs (x,y), $y \in \{1,\ldots k\}$
- The "black box learner" (1 vs. all) we described might be thought of as a function of x only but, actually, we made use of the labels y
- How is y being used?
  - y decides what to do with the example x; that is, which of the k classifiers should take the example as a positive example (making it a negative to all the others).
- How do we predict?
  - Let: $f_y(x) = w_y^T \cdot x$
  - Then, we predict using: $y^* = \text{argmax}_{y=1,\ldots k} f_y(x)$

Is it better in any well defined way?

- Equivalently, we can say that we predict as follows:
  - Predict y iff
  - $\forall y' \in \{1,\ldots k\}, y' \neg= y \quad (w_y^T - w_{y'}^T) \cdot x \geq 0 \quad (**)$
- So far, we did not say how we learn the k weight vectors w_y $(y = 1,\ldots k)$
  - Can we train in a way that better fits the way we predict?
  - What does it mean?

We showed: if pairs of labels are separable (a reasonable assumption) than in some higher dimensional space, the problem is linearly separable.

Linear Separability for Multiclass



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

  $w = (w_1, w_2, \ldots w_k) \in$

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

- Key Construction: (Kesler Construction; Zimak's Constraint Classification)
  - We will represent each example (x,y), as an nk-dimensional vector, x_y, with x embedded in the y-th part of it (y=1,2,…k) and the other coordinates are 0.
- **E.g.,** $\mathbf{x_y} = (\mathbf{0}, x, \mathbf{0}, \mathbf{0}) \in \mathbf{R}^{kn}$ (here k=4, y=2)
- Now we can understand the n-dimensional decision rule:
- Predict y iff $\forall y' \in \{1,\ldots k\}, y' \neg= y \quad (w_y^T - w_{y'}^T) \cdot x \geq 0$ (**)
- Equivalently, in the nk-dimensional space.
- Predict y iff $\forall y' \in \{1,\ldots k\}, y' \neg= y \quad w^T \cdot (x_y - x_{y'}) \equiv w^T \cdot x_{yy'} \geq 0$

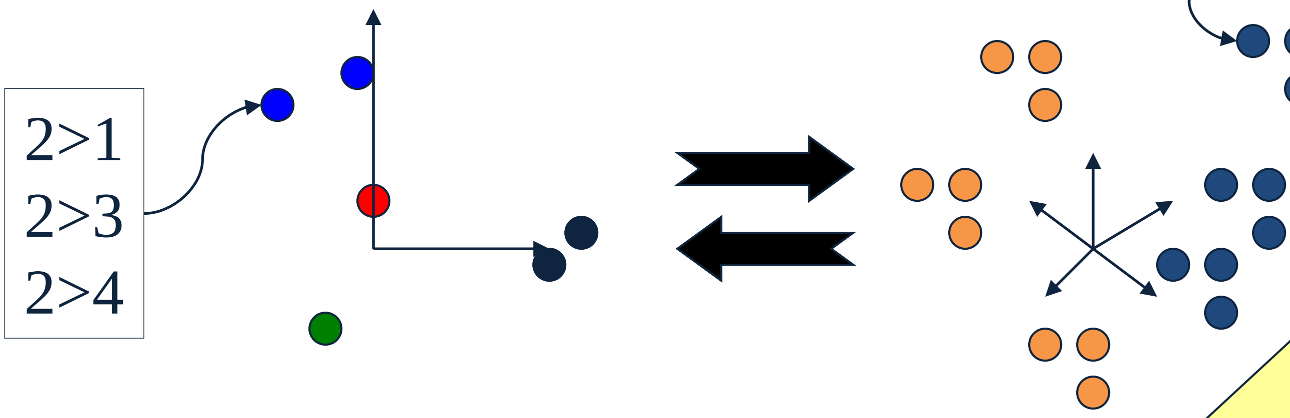
- Conclusion: The set $(x_{yy'}, +) \equiv (x_y - x_{y'}, +)$ is linearly separable from the set $(-x_{yy'}, -)$ using the linear separator $w \in R^{kn}$,
  - We solved the voroni diagram challenge.

Constraint Classification

- Training:
  - [We first explain via Kesler's construction; then show we don't need it]
  - Given a data set {(x,y)}, (m examples) with $x \in R^n$, $y \in \{1,2,\ldots k\}$ create a binary classification task (in R^{kn}):
    $(x_y - x_{y'}, +)$, $(x_{y'} - x_y, -)$, for all $y' \neg = y$ (2m(k-1) examples)
    Here $x_y \in R^{kn}$
  - Use your favorite linear learning algorithm to train a binary classifier.
- Prediction:
  - Given an nk dimensional weight vector w and a new example x, predict: $\text{argmax}_y\, w^T x_y$

Details: Kesler Construction & Multi-Class Separability

- Transform Examples



If (x,i) was a given n-dimensional example (that is, x has is labeled i, then x_{ij}, $\forall$ j=1,…k, j¬= i, are positive examples in the nk-dimensional space. $-x_{ij}$ are negative examples.

$$
\begin{array}{lll}
i>j & f_i(x) - f_j(x) & > 0 \\
 & w_i \cdot x - w_j \cdot x & > 0 \\
 & \mathbf{W} \cdot \mathbf{X}_i - \mathbf{W} \cdot \mathbf{X}_j & > 0 \\
 & \mathbf{W} \cdot (\mathbf{X}_i - \mathbf{X}_j) & > 0 \\
 & \mathbf{W} \cdot \mathbf{X}_{ij} & > 0
\end{array}
$$

$$\mathbf{X}_i = (\mathbf{0}, x, \mathbf{0}, \mathbf{0}) \in \mathbf{R}^{kd}$$

$$\mathbf{X}_j = (\mathbf{0}, \mathbf{0}, \mathbf{0}, x) \in \mathbf{R}^{kd}$$

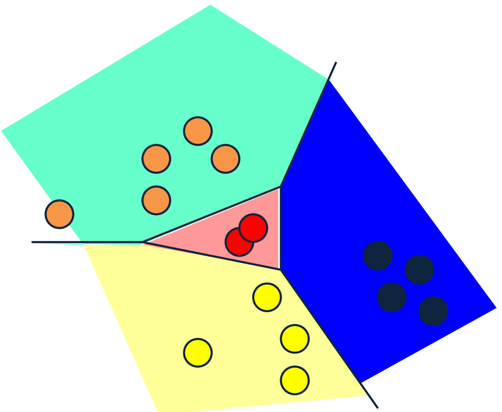
$$\mathbf{X}_{ij} = \mathbf{X}_i - \mathbf{X}_j = (\mathbf{0}, x, \mathbf{0}, -x)$$

$$\mathbf{W} = (w_1, w_2, w_3, w_4) \in \mathbf{R}^{kd}$$

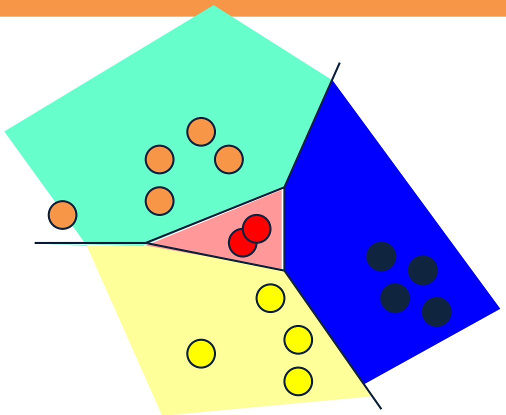
Kesler's Construction (1)

- $y = \text{argmax}_{i=(r,b,g,y)}\ w_i.x$
  - $w_i,\ x \in \mathbf{R}^n$
- Find $w_r, w_b, w_g, w_y \in \mathbf{R}^n$ such that
  - $w_r.x > w_b.x$
  - $w_r.x > w_g.x$
  - $w_r.x > w_y.x$

$\mathbf{H} = \mathbf{R}^{kn}$



Kesler's Construction (2)



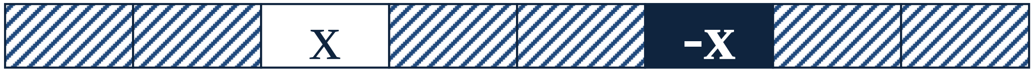
- Let $\mathbf{w} = (w_r, w_b, w_g, w_y) \in \mathbf{R}^{kn}$
- Let $\mathbf{0}^n$, be the n-dim zero vector

- $w_r.x > w_b.x \Leftrightarrow \mathbf{w}.(x,-x,\mathbf{0}^n,\mathbf{0}^n) > 0 \Leftrightarrow \mathbf{w}.(-x,x,\mathbf{0}^n,\mathbf{0}^n) < 0$
- $w_r.x > w_g.x \Leftrightarrow \mathbf{w}.(x,\mathbf{0}^n,-x,\mathbf{0}^n) > 0 \Leftrightarrow \mathbf{w}.(-x,\mathbf{0}^n,x,\mathbf{0}^n) < 0$
- $w_r.x > w_y.x \Leftrightarrow \mathbf{w}.(x,\mathbf{0}^n,\mathbf{0}^n,-x) > 0 \Leftrightarrow \mathbf{w}.(-x,\mathbf{0}^n,\mathbf{0}^n,x) < 0$

+ −

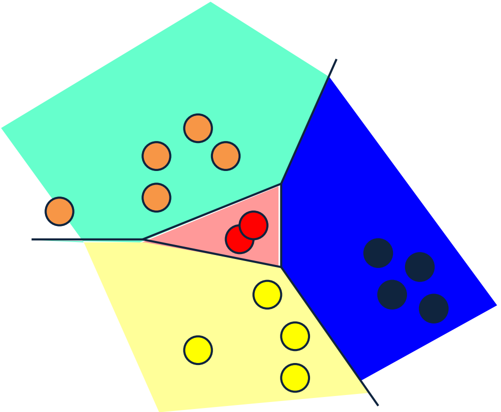
Kesler's Construction (3)

- Let
  - $\mathbf{w} = (w_1, \ldots, w_k) \in \mathbf{R}^n \times \ldots \times \mathbf{R}^n = \mathbf{R}^{kn}$
  - $\mathbf{x}_{ij} = (\mathbf{0}^{(i-1)n}, x, \mathbf{0}^{(k-i)n}) - (\mathbf{0}^{(j-1)n}, -x, \mathbf{0}^{(k-j)n}) \in \mathbf{R}^{kn}$

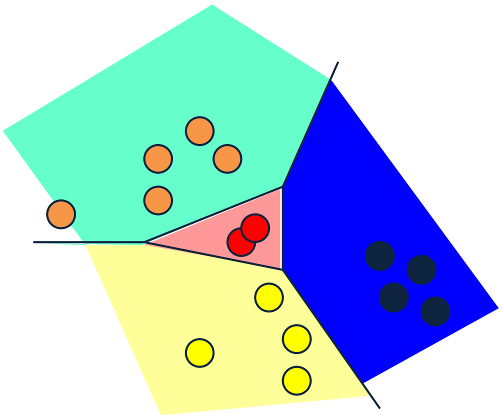


- Given $(x, y) \in \mathbf{R}^n \times \{1,\ldots,k\}$
  - For all $j \neq y$ (all other labels)
    - Add to $\mathbf{P}^+(x,y)$, $(\mathbf{x}_{yj}, 1)$
    - Add to $\mathbf{P}^-(x,y)$, $(-\mathbf{x}_{yj}, -1)$

- $\mathbf{P}^+(x,y)$ has k-1 positive examples ($\in \mathbf{R}^{kn}$)
- $\mathbf{P}^-(x,y)$ has k-1 negative examples ($\in \mathbf{R}^{kn}$)



Learning via Kesler's Construction



- Given $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbf{R}^n \times \{1,\ldots,k\}$
- Create
  - $\mathbf{P}^+ = \cup\, \mathbf{P}^+(x_i,y_i)$
  - $\mathbf{P}^- = \cup\, \mathbf{P}^-(x_i,y_i)$
- Find $\mathbf{w} = (w_1, \ldots, w_k) \in \mathbf{R}^{kn}$, such that
  - $\mathbf{w.x}$ separates $\mathbf{P}^+$ from $\mathbf{P}^-$
- One can use any algorithm in this space: Perceptron, Winnow, SVM, etc.
- To understand how to update the weight vector in the n-dimensional space, we note that
- $w^T \cdot x_{yy'} \geq 0$ (in the nk-dimensional space)
- is equivalent to:
- $(w_y^T - w_{y'}^T) \cdot x \geq 0$ (in the n-dimensional space)

Perceptron in Kesler Construction

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

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

- Note that this is a generalization of balanced Winnow rule.

- Note that we promote w_i and demote k-1 weight vectors w_j

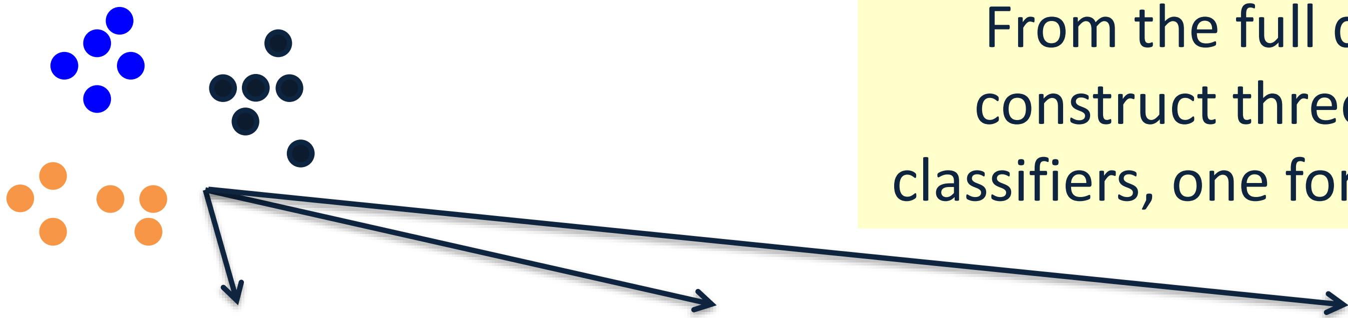
Conservative update

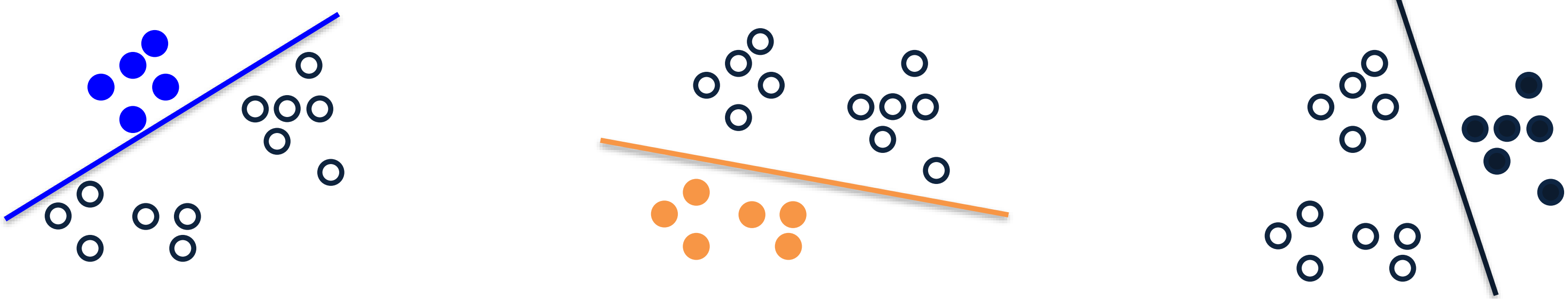
- The general scheme suggests:
- Given example (x,i) (example $x \in R^n$, labeled i)
  - $\forall (i,j)$, $i,j = 1,\ldots k$, $i \neg= j$ (***)
  - If $(w_i^T - w_j^T) \cdot x < 0$ (mistaken prediction; equivalent to $w^T \cdot x_{ij} < 0$)
  - $w_i \leftarrow w_i + x$ (promotion) and $w_j \leftarrow w_j - x$ (demotion)
- Promote w_i and demote $k-1$ weight vectors w_j
- A conservative update: (SNoW and LBJava's implementation):
  - In case of a mistake: only the weights corresponding to the target node i and that closest node j are updated.
  - Let: $j^* = \text{argmax}_{j=1,\ldots k}\, w_j^T \cdot x$ (highest activation among competing labels)
  - If $(w_i^T - w_{j^*}^T) \cdot x < 0$ (mistaken prediction)
  - $w_i \leftarrow w_i + x$ (promotion) and $w_{j^*} \leftarrow w_{j^*} - x$ (demotion)
  - Other weight vectors are not being updated.

Multiclass Classification Summary 1:

Multiclass Classification

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





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

$\mathbf{w}_{org}^T\mathbf{x}$ > 0 for **orange inputs**

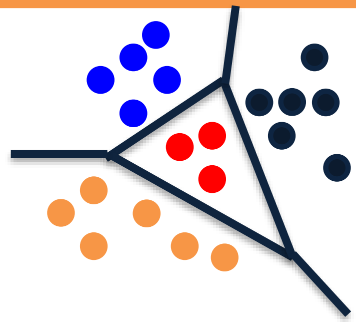
$\mathbf{w}_{black}^T\mathbf{x}$ > 0 for **black inputs**

Notation: Score for blue label

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

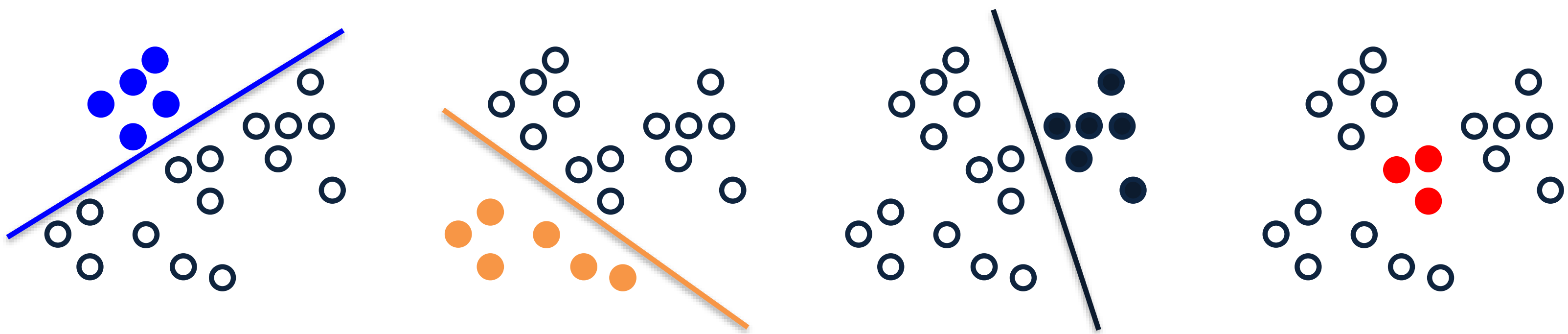
Multiclass Classification Summary 2:

One-vs-all may not always work



Red points are not separable with a single binary classifier

The decomposition is not expressive enough!



$\mathbf{w}_{blue}^T\mathbf{x} > 0$ for **blue** **inputs**	$\mathbf{w}_{org}^T\mathbf{x} > 0$ for **orange** **inputs**	$\mathbf{w}_{black}^T\mathbf{x} > 0$ for **black** **inputs**	**???**

Summary 3:
Local Learning: One-vs-all classification

- Easy to learn
  - Use any binary classifier learning algorithm
- Potential Problems
  - Calibration issues
    - We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  - Train vs. Train
    - Does not account for how the final predictor will be used
    - Does not optimize any **global** measure of correctness
  - Yet, works fairly well
    - In most cases, especially in high dimensional problems (everything is already linearly separable).

Summary 4:
Global Multiclass Approach [Constraint Classification, Har-Peled et. al '02]

- Create K classifiers $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$. ;
- Predict with WTA: $\mathrm{argmax}_i\ \mathbf{w}_i^T\mathbf{x}$

- **But**, train differently:
  - **For examples with label i, we want**
    $w_i^Tx > w_j^Tx$ for all j
- **Training:** For each training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$:

$$\hat{y} \leftarrow \ arg \max_{\boldsymbol{j}} \boldsymbol{w}_{\boldsymbol{j}}^T \phi(\boldsymbol{x}_i, y_i)$$

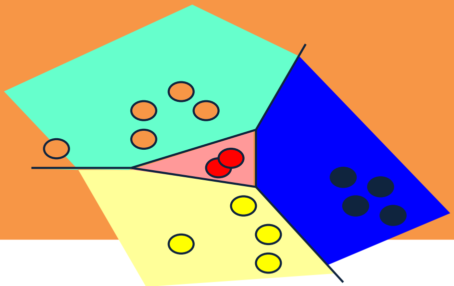
if $\hat{y} \neq y_i$

$$\boldsymbol{w}_{y_i} \leftarrow \boldsymbol{w}_{y_i} + \eta \boldsymbol{x}_i \qquad \textbf{(promote)}$$

$$\boldsymbol{w}_{\hat{y}} \leftarrow \boldsymbol{w}_{\hat{y}} - \eta \boldsymbol{x}_i \qquad \textbf{(demote)}$$

η: learning rate

Significance



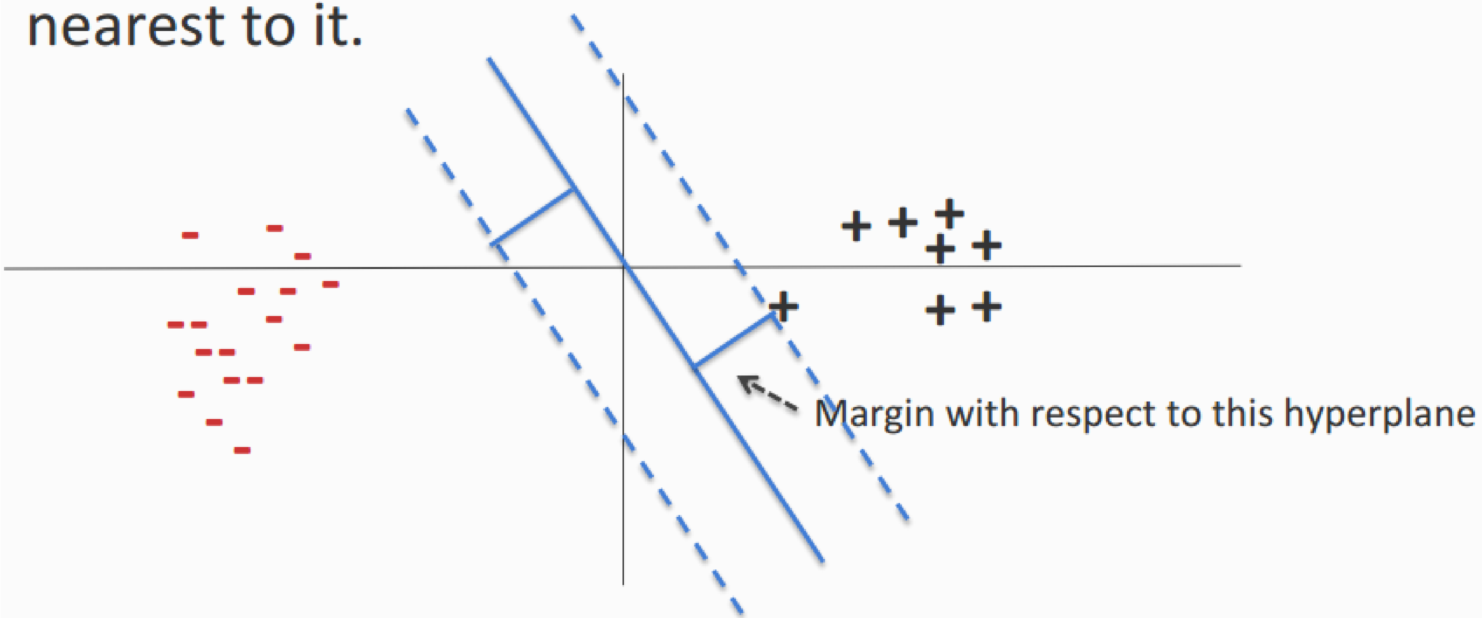
- The hypothesis learned above is more expressive than when the OvA assumption is used.
- Any linear learning algorithm can be used, and algorithmic-specific properties are maintained (e.g., attribute efficiency if using winnow.)
- E.g., the multiclass support vector machine can be implemented by learning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

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

$$\text{margin}(x, w) = \min_j (w_i^T - w_j^T) \cdot x$$

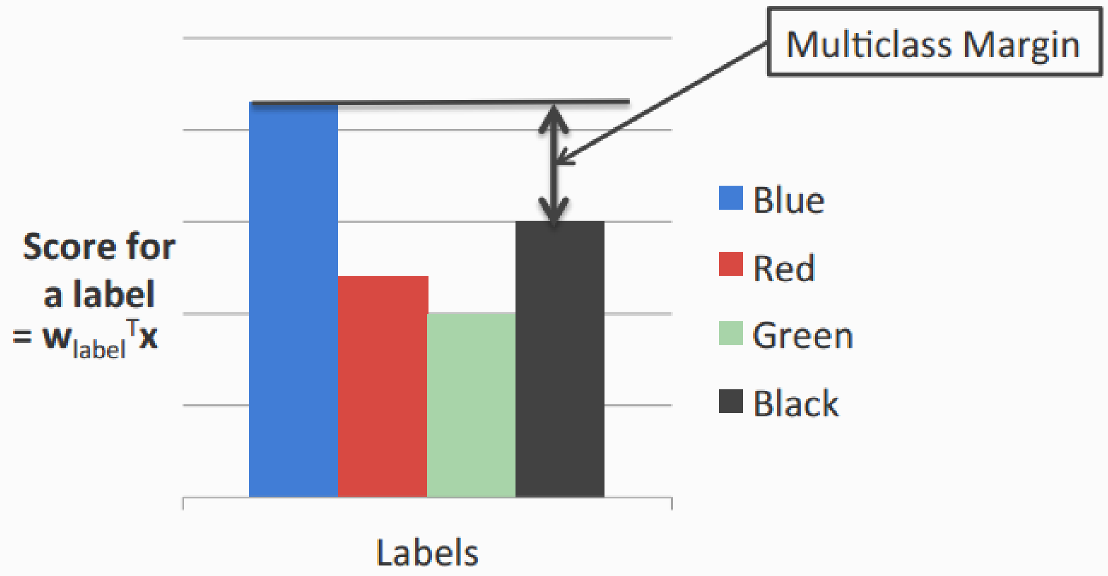
Margin

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



Multiclass Margin

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



Constraint Classification

- The scheme presented can be generalized to provide a uniform view for multiple types of problems: multi-class, multi-label, category-ranking
- Reduces learning to a *single* binary learning task
- Captures theoretical properties of binary algorithm
- Experimentally verified
- Naturally extends Perceptron, SVM, etc...

- *It is called "constraint classification" since it does it all by representing labels as a set of constraints or preferences among output labels.*

Multi-category to Constraint Classification

- The unified formulation is clear from the following examples:
- 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))

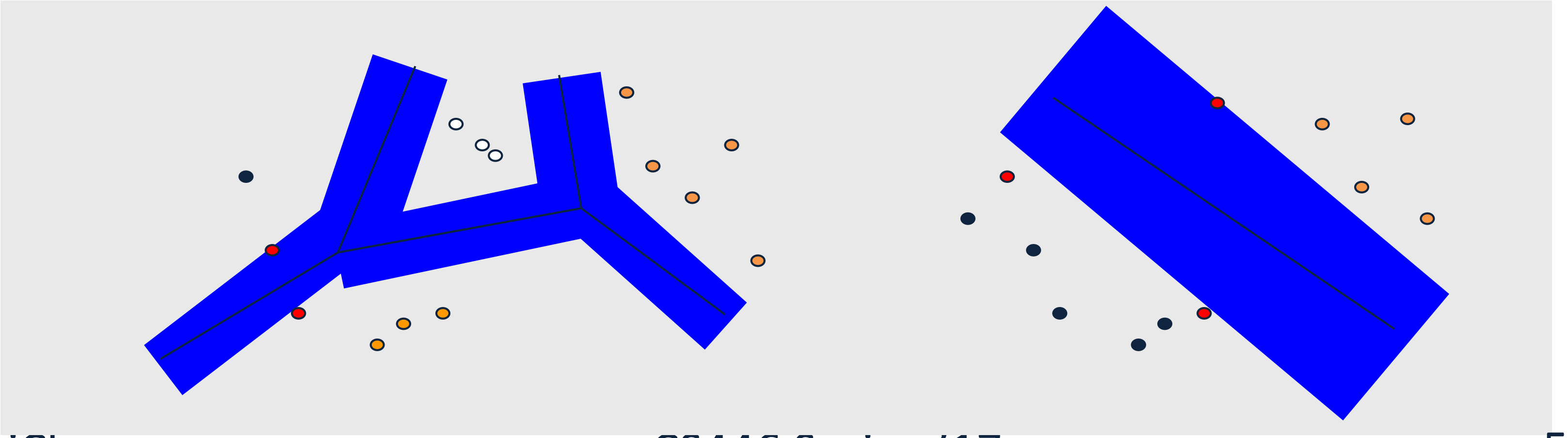
Just like in the multiclass we learn one $w_i \in R^n$ for each label, the same is done for multi-label and ranking. The weight vectors are updated according with the requirements from $y \in S_k$

(Consult the Perceptron in Kesler construction slide)

- 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: $\mathbf{X} \longrightarrow \mathbf{S_k}$
  - h(x) is a partial order
  - h(x) is *consistent* with y if $(i<j) \in y \rightarrow (i<j) \in h(x)$

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

- Can learn *any* argmax $v_i.x$ 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^{kd}
  - multiclass SVM [Crammer, Singer 2001]



Margin Generalization Bounds

- Linear Hypothesis space:
  - $h(x) = \text{argsort } v_i.x$
    - $v_i, x \in \mathbf{R}^d$
    - argsort returns *permutation* of {1,…,k}
- CC margin-based bound
  - $\gamma = \min_{(x,y)\in \mathbf{S}} \min_{(i<j)\in \mathbf{y}} v_i.x - v_j.x$

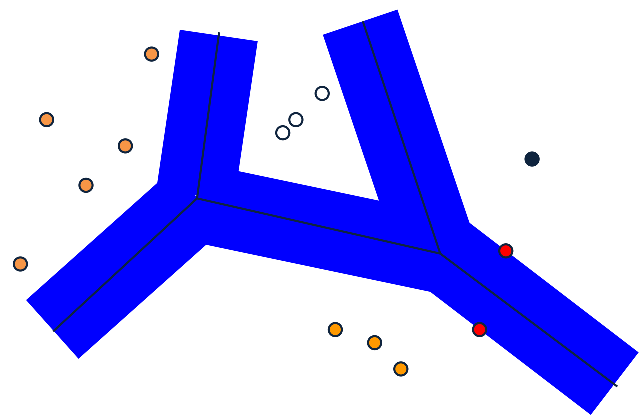
$$err_D(h) \le \Theta\left(\frac{C}{m}\left(\frac{R^2}{\gamma^2} - \ln(\delta)\right)\right)$$

m - number of examples

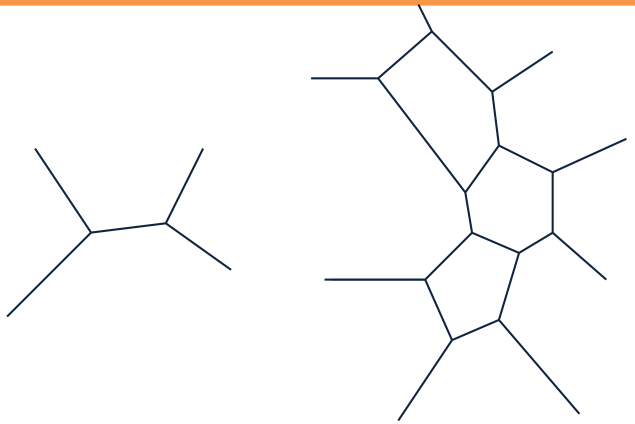
R - $\max_x ||x||$

δ - confidence

C - average # constraints



VC-style Generalization Bounds



- Linear Hypothesis space:
  - h(x) = argsort $v_i.x$
    - v_i, $x \in \mathbf{R}^d$
    - argsort returns *permutation* of {1,...,k}
- CC VC-based bound

$$err_D(h) \leq err(S,h) + \theta\left(\sqrt{\frac{kd\log(mk/d) - \ln\delta}{}}\right)$$

m - number of examples

d - dimension of input space

delta - confidence

k - number of classes

Performance: even though this is the right thing to do, and differences can be observed in low dimensional cases, in high dimensional cases, the impact is not always significant.

Beyond MultiClass Classification

- Ranking
  - category ranking (over classes)
  - ordinal regression (over examples)

- Multilabel
  - **x** is both red and blue
- Complex relationships
  - **x** is more red than blue, but not green

- Millions of classes
  - sequence labeling (e.g. POS tagging)
  - The same algorithms can be applied to these problems, namely, to Structured Prediction
  - This observation is the starting point for CS546.

(more) Multi-Categorical Output Tasks

- Sequential Prediction ($y \in \{1,...,K\}^+$)
  - *e.g. POS tagging ('(NVNNA)')*
    - "This is a sentence." $\Rightarrow$ D V D N
  - *e.g. phrase identification*
  - *Many* labels: K^L for length L sentence
- Structured Output Prediction ($y \in C(\{1,...,K\}^+)$)
  - *e.g. parse tree, multi-level phrase identification*
  - *e.g. sequential prediction*
  - Constrained by
    - domain, problem, data, background knowledge, etc...

Semantic Role Labeling: A Structured-Output Problem

- For each verb in a sentence
  1. Identify all constituents that fill a semantic role
  2. Determine their roles
     - Core Arguments, e.g., Agent, Patient or Instrument
     - Their adjuncts, e.g., Locative, Temporal or Manner

A0 : leaver A2 : benefactor

I **left** my pearls to my daughter-in-law in my will.

A1 : thing left AM-LOC

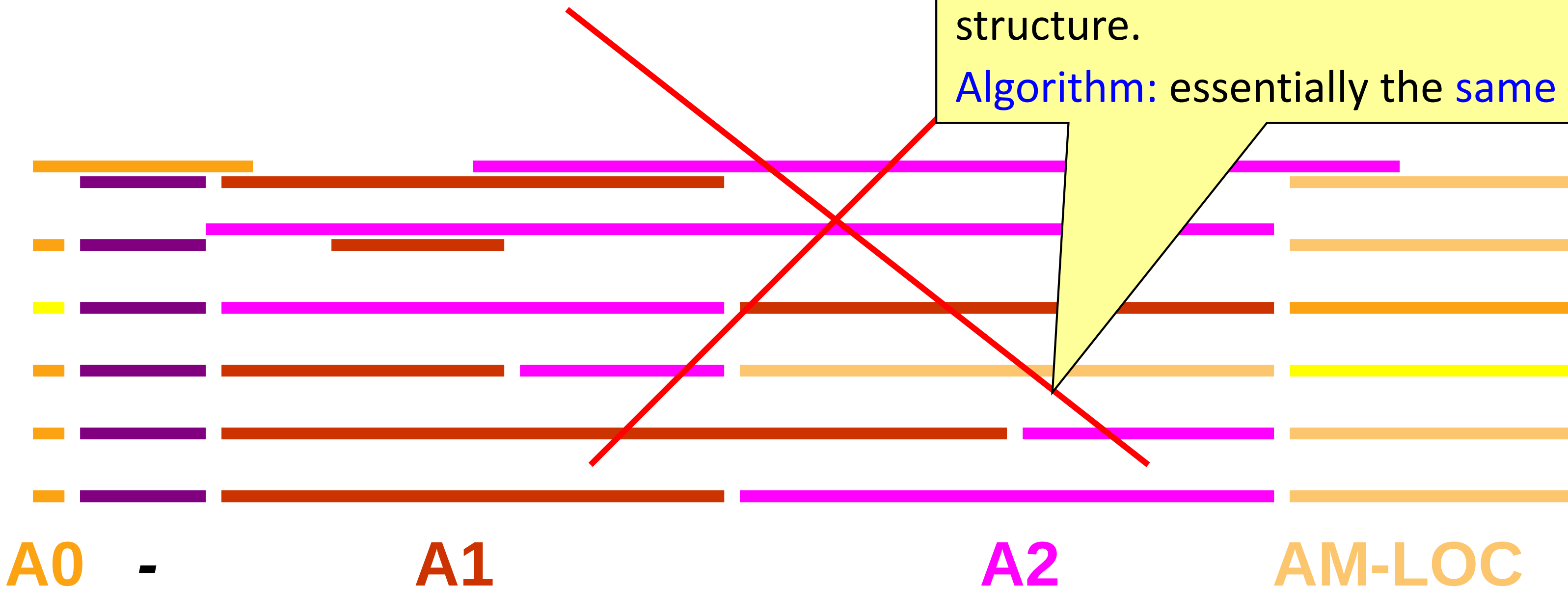
Semantic Role Labeling

Just like in the multiclass case we can think about local vs. global predictions.

Local: each component learned separately, w/o thinking about other components.

Global: learn to predicting the whole structure.

Algorithm: essentially the same as CC



I ***left*** my pearls to my daughter-in-law in my will.

- Many possible *valid* outputs
- Many possible *invalid* outputs
- Typically, one *correct* output (per input)