Multiclass Classification

David Rosenberg

New York University

March 30, 2016

Introduction

Multiclass Setting

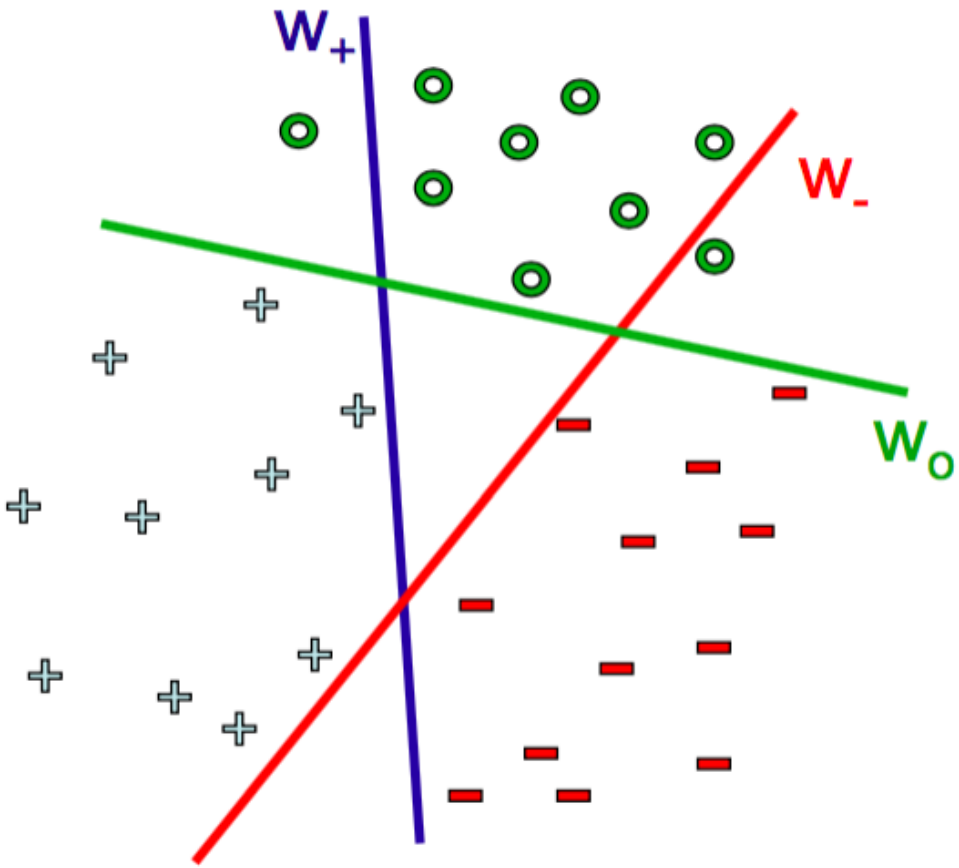
- Input space: $\mathcal{X}$
- Ouput space: $\mathcal{Y} = \{1, \ldots, k\}$

- So far, our only approach to multiclass problems is to use trees.
- (And by extension, random forests.)

- Today we consider linear methods specifically designed for multiclass.

Reduction to Binary Classification

One-vs-All / One-vs-Rest



Plot courtesy of David Sontag.

One-vs-All / One-vs-Rest

- Train k binary classifiers, one for each class.
- Train ith classifier to distinguish class i from rest
- Suppose $h_1, \ldots, h_k : \mathcal{X} \to \mathbf{R}$ are our binary classifiers.
  - Can output hard classifications in $\{-1, 1\}$ or scores in $\mathbf{R}$.
- Final prediction is

$$h(x) = \underset{i \in \{1,\ldots,k\}}{\arg\max}\, h_i(x)$$

- Ties can be broken arbitrarily.

Linear Classifers: Binary and Multiclass

Linear Binary Classifier Review

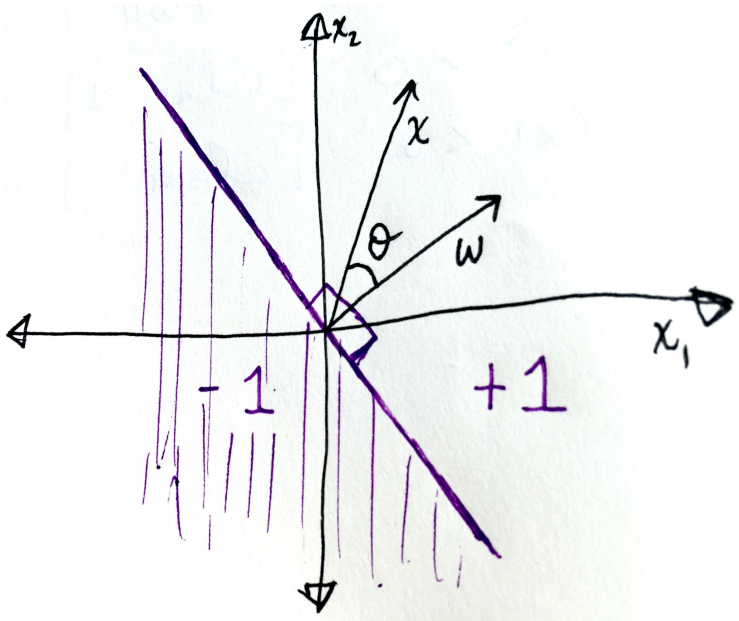
- Input Space: $\mathcal{X} = \mathbf{R}^d$
- Output Space: $\mathcal{Y} = \{-1, 1\}$

- Linear classifier score function:

$$f(x) \quad = \quad \langle w, x \rangle = w^T x$$

- Final classification prediction: $\text{sign}(f(x))$
- Geometrically, when are $\text{sign}(f(x)) = +1$ and $\text{sign}(f(x)) = -1$?

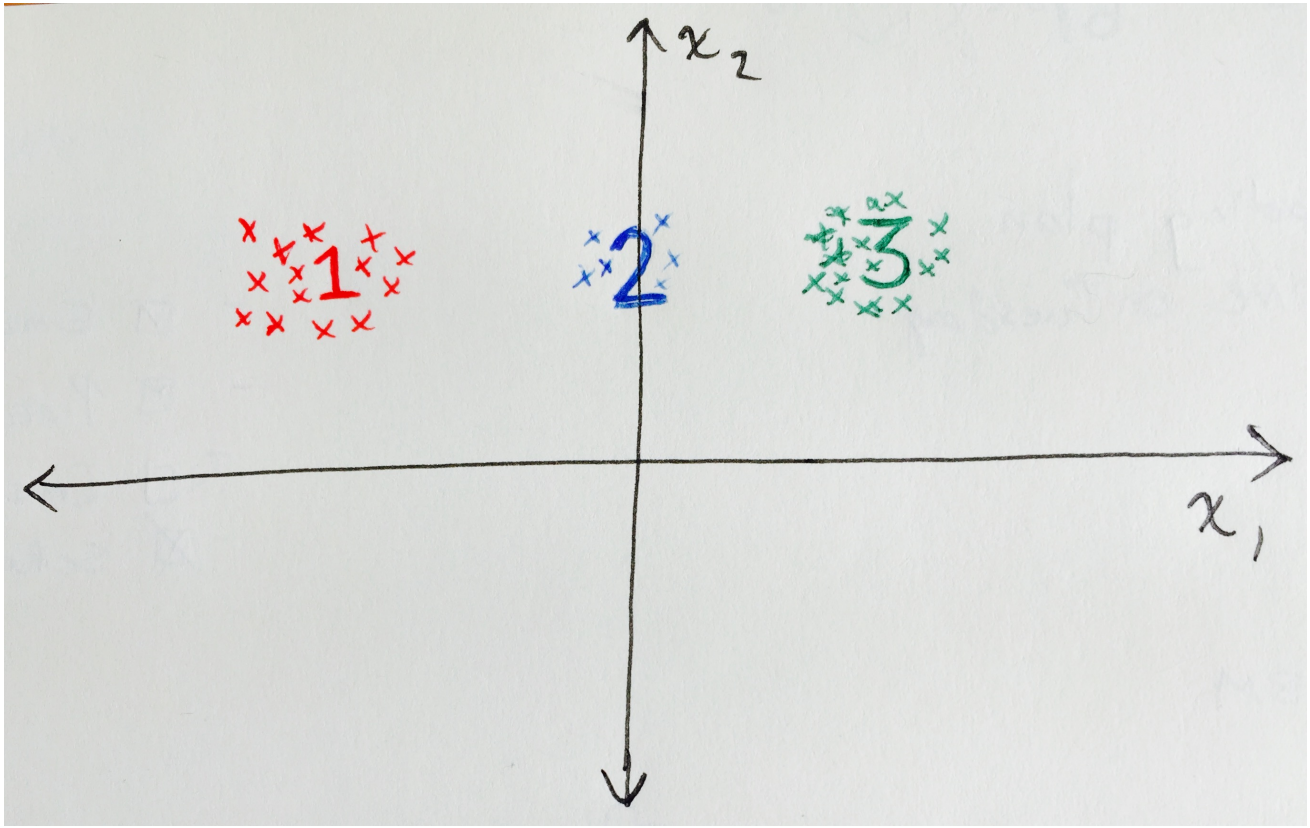
Linear Binary Classifier Review



Suppose $\|w\| > 0$ and $\|x\| > 0$:

$$\begin{aligned} f(x) &= \langle w, x \rangle = \|w\|\|x\| \cos\theta \\ f(x) > 0 &\iff \cos\theta > 0 \iff \theta \in (-90°, 90°) \\ f(x) < 0 &\iff \cos\theta < 0 \iff \theta \notin [-90°, 90°] \end{aligned}$$

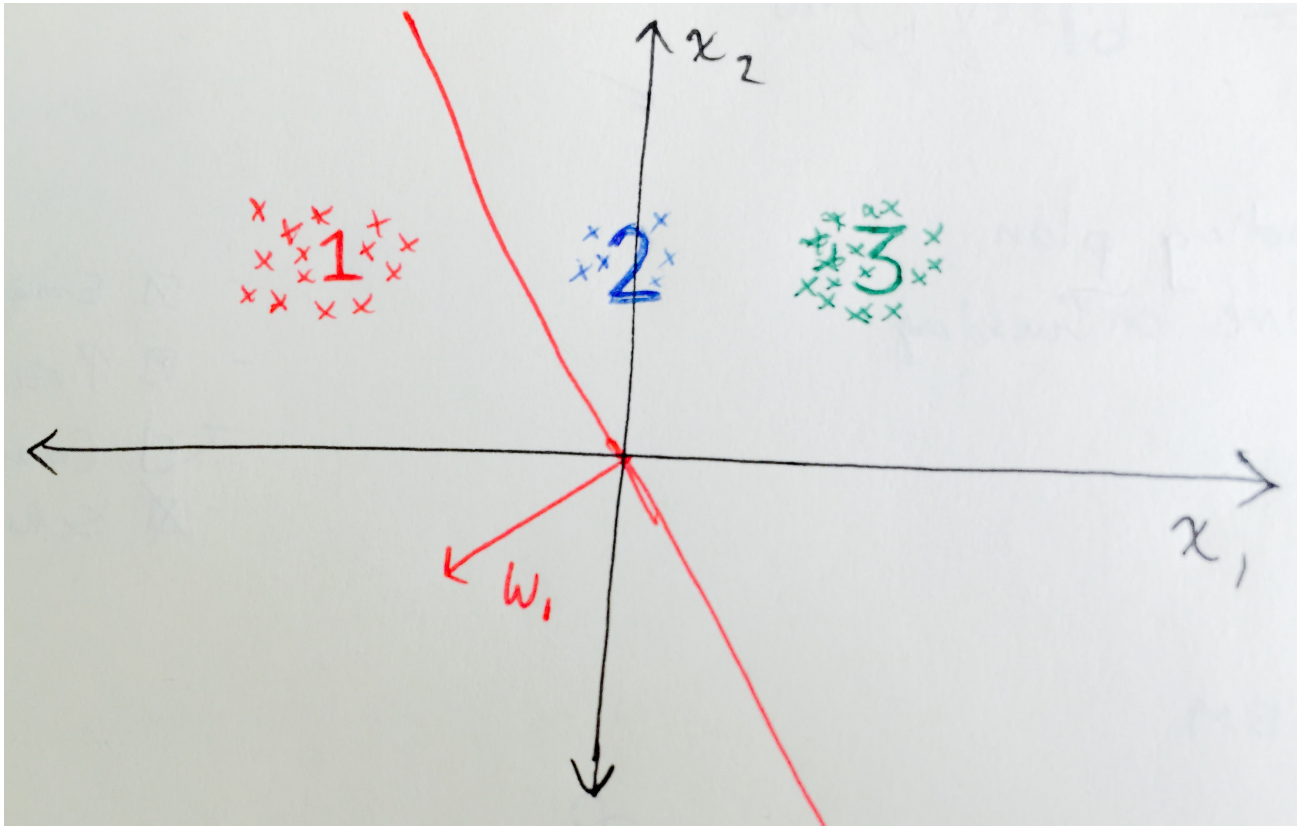
Three Class Example



- Base hypothesis space $\mathcal{H} = \left\{ f(x) = w^T x \mid x \in \mathbf{R}^2 \right\}$.
- Note: Separating boundary always contains the origin.

Example based on Shalev-Schwartz and Ben-David's *Understanding Machine Learning*, Section 17.1

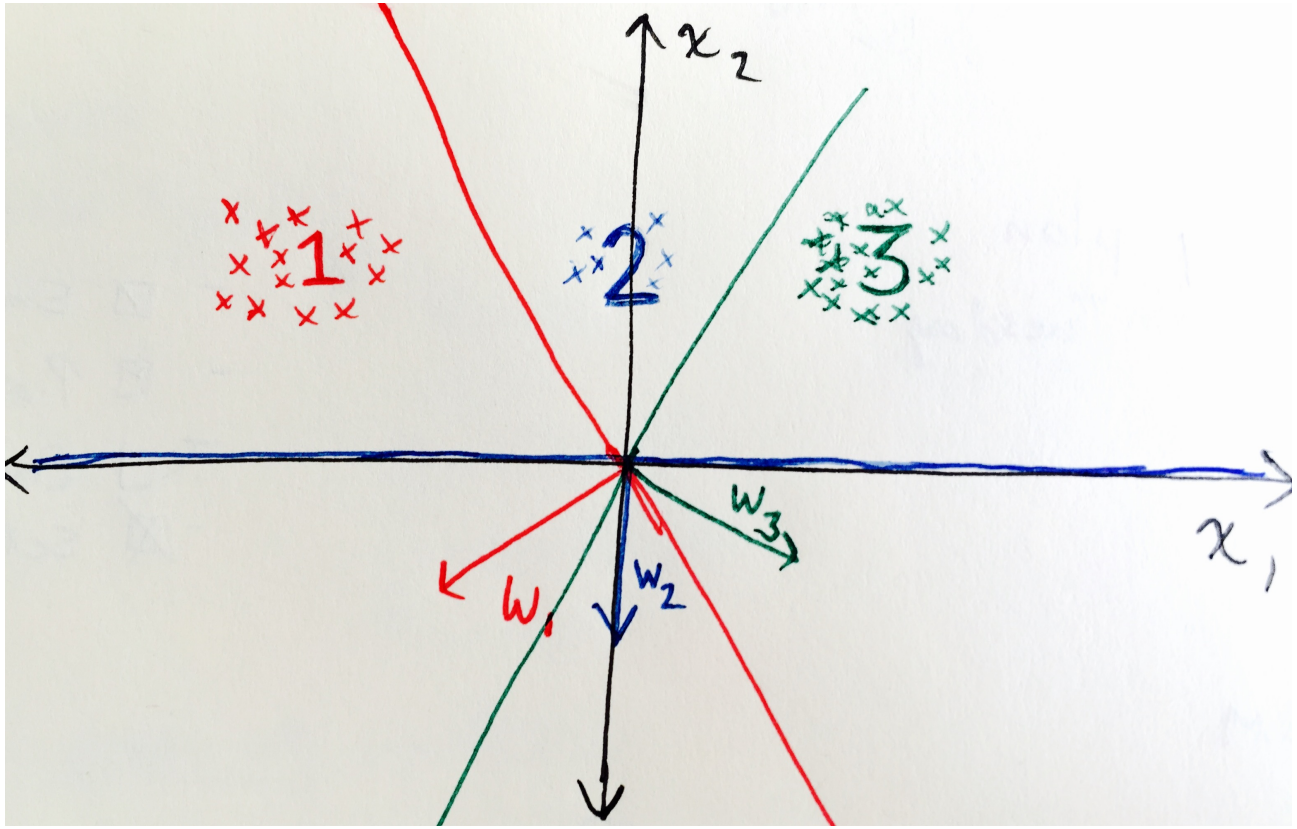
Three Class Example: One-vs-Rest



- Class 1 vs Rest:

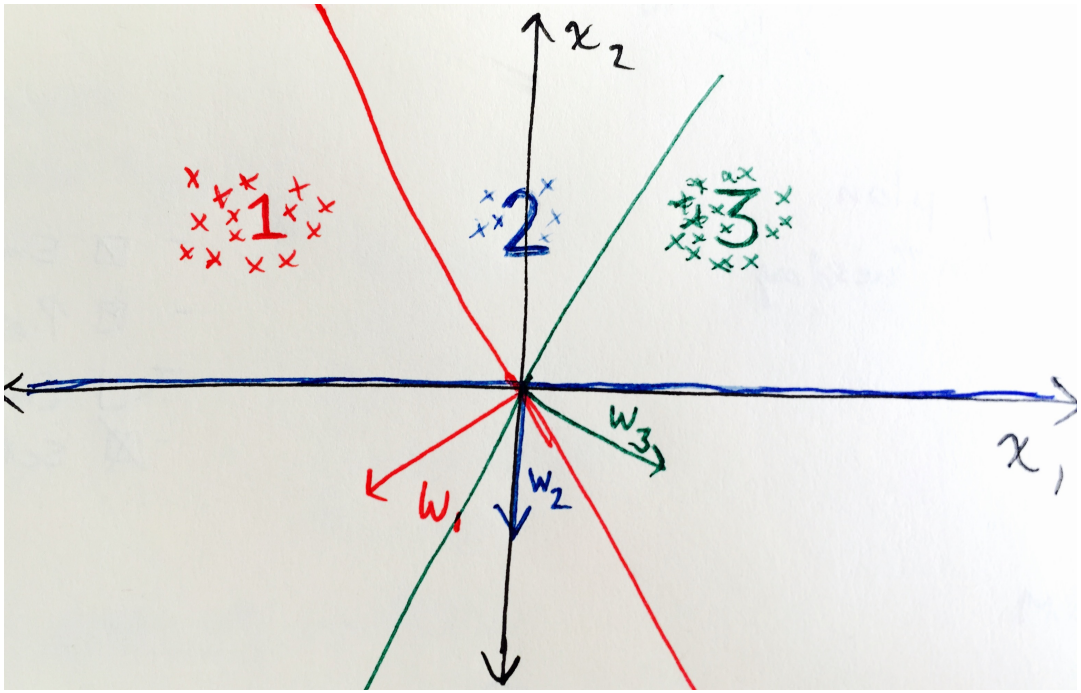
$$f_1(x) = w_1^T x$$

Three Class Example: One-vs-Rest



- Examine "Class 2 vs Rest"
  - Predicts everything to be "Not 2".
  - If it predicted some "2", then it would get many more "Not 2" incorrect.

One-vs-Rest: Predictions



- Score for class i is

$$f_i(x) = \langle w_i, x \rangle = \|w_i\| \|x\| \cos\theta_i,$$

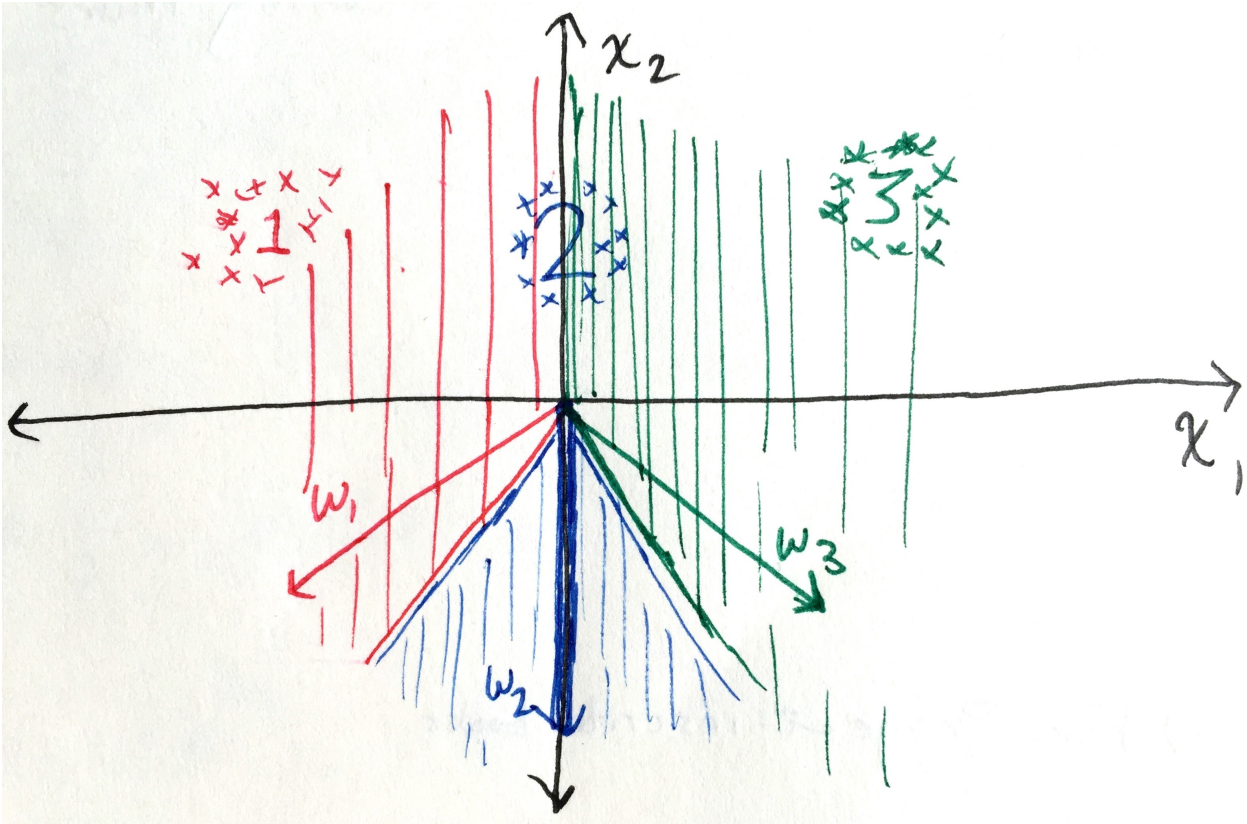
  where θ_i is the angle between x and w_i.

- Predict class i that has highest $f_i(x)$.

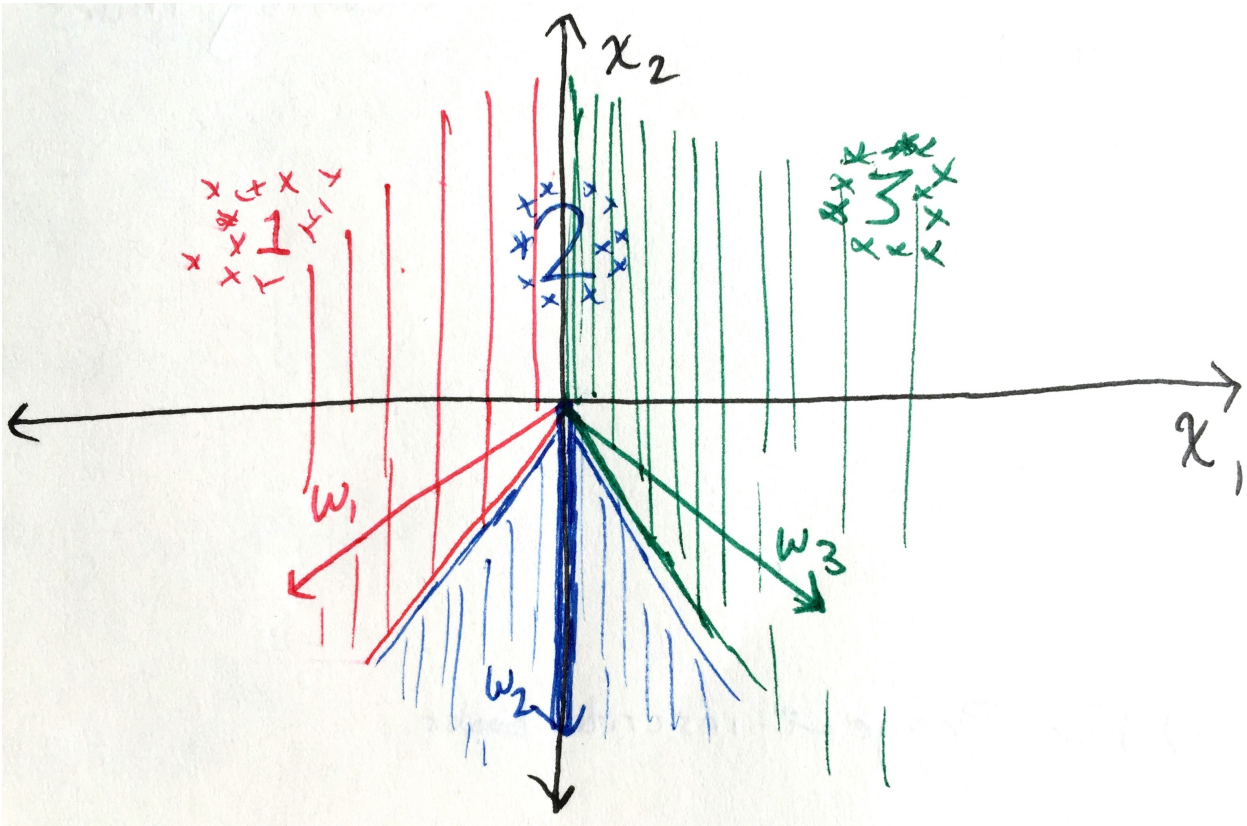
One-vs-Rest: Class Boundaries

- For simplicity, we've assumed $\|w_1\| = \|w_2\| = \|w_3\|$.
- Then $\|w_i\|$ and $\|x\|$ are equal for all scores.

$\implies$ x is classified by whichever has smallest $\cos\theta_i$.



One-vs-Rest: Class Boundaries



- This approach doesn't work well in this instance.
- Can we fix it by changing our base hypothesis space?

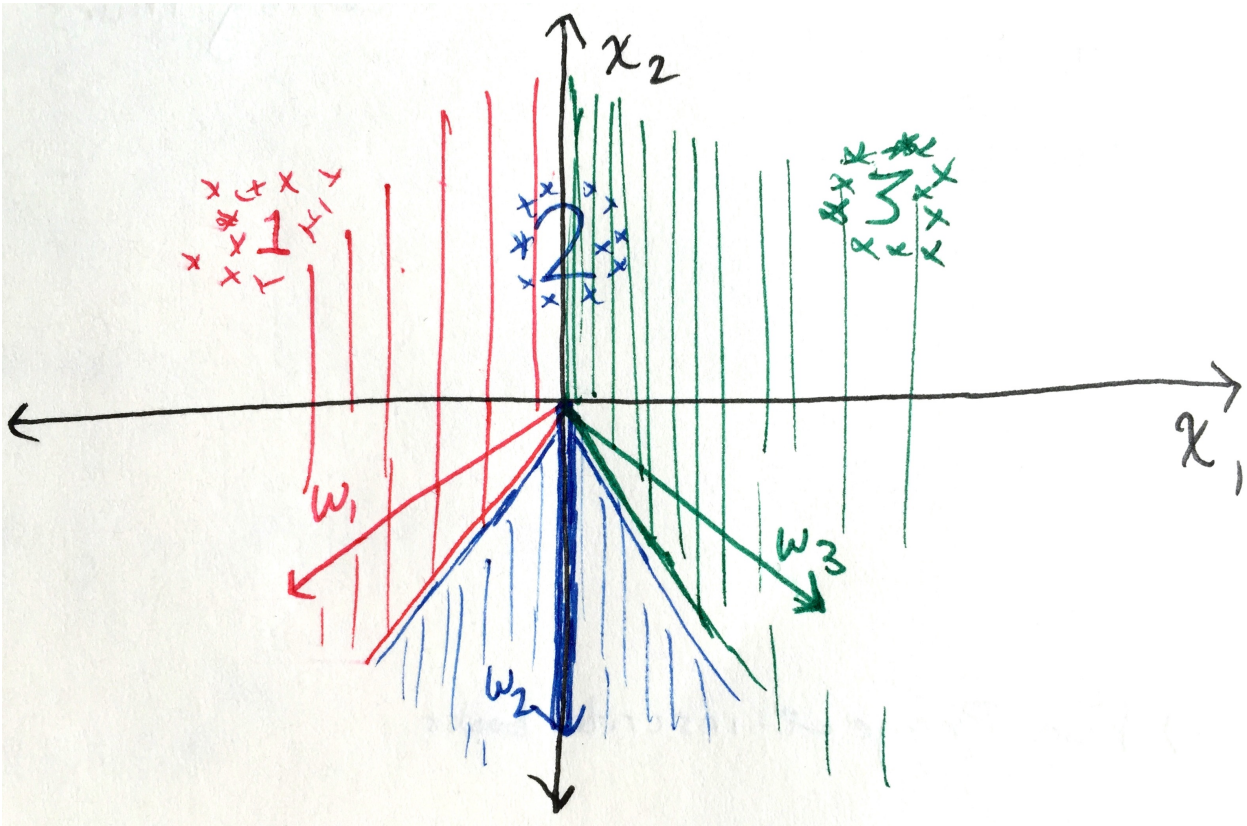
The Linear Multiclass Hypothesis Space

- **Base Hypothesis Space**: $\mathcal{H} = \left\{ x \mapsto w^T x \mid w \in \mathbf{R}^d \right\}$.
- **Linear Multiclass Hypothesis Space** (for k classes):

$$\mathcal{F} = \left\{ x \mapsto \arg\max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$$

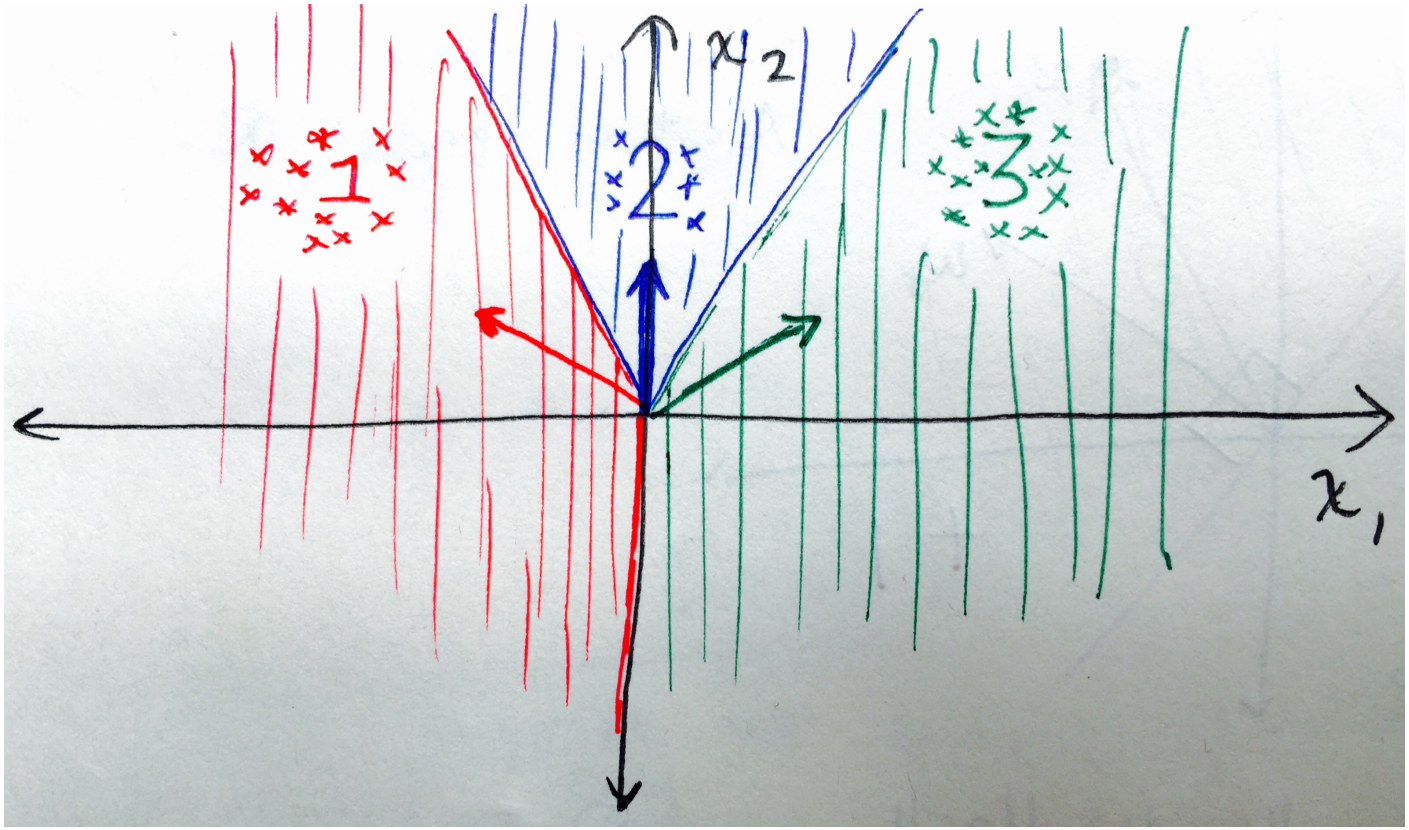
- What's the action space here?

One-vs-Rest: Class Boundaries



- Is this a failure of the hypothesis space or the learning algorithm?
- (A **learning algorithm** chooses the hypothesis from the hypothesis space.)

A Solution with Linear Functions



- This works... so the problem is not with the hypothesis space.
- How can we get a solution like this?

Multiclass Predictors

Multiclass Hypothesis Space

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathbf{R}\}$ ("score functions").
- **Multiclass Hypothesis Space** (for k classes):

$$\mathcal{F} = \left\{ x \mapsto \arg\max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$$

- $h_i(x)$ **scores** how likely x is to be from class i.

Multiclass Hypothesis Space: Reframed

- A slight reframing turns out to be more convenient down the line
- **General Output Space:** $\mathcal{Y}$
  - e.g $\mathcal{Y} = \{1, \ldots, k\}$for multiclass
- **Base Hypothesis Space:** $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}\}$
  - gives **compatibility score** between input x and output y
- **Multiclass Hypothesis Space**

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} h(x, y) \mid h \in \mathcal{H} \right\}$$

- Now we're back to a single score function.
- Takes x **and** y and evalutes their compatibility.

Learning in a Multiclass Hypothesis Space: In Words

- **Base Hypothesis Space:** $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}\}$
- Training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learning process chooses $h \in \mathcal{H}$.
- What type of h do we want?
- Want $h(x, y)$ to be large when x has label y, small otherwise.

Learning in a Multiclass Hypothesis Space: In Math

- $h(x, y)$ classifies(x_i, y_i) correctly iff

$$h(x_i, y_i) > h(x_i, y) \, \forall y \neq y_i$$

- h should give higher score for correct y than for all other $y \in \mathcal{Y}$.
- An equivalent condition is the following:

$$h(x_i, y_i) > \max_{y \neq y_i} h(x_i, y)$$

- First idea for objective function:

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{n} \ell \left[h(x_i, y_i) - \max_{y \neq y_i} h(x_i, y) \right]$$

Linear Hypothesis Space

Linear Multiclass Prediction Function

- A **linear class-sensitive score function** is given by

$$h(x, y) = \langle w, \Psi(x, y) \rangle ,$$

where $\Psi(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}^d$ is a **class-sensitive feature map**.

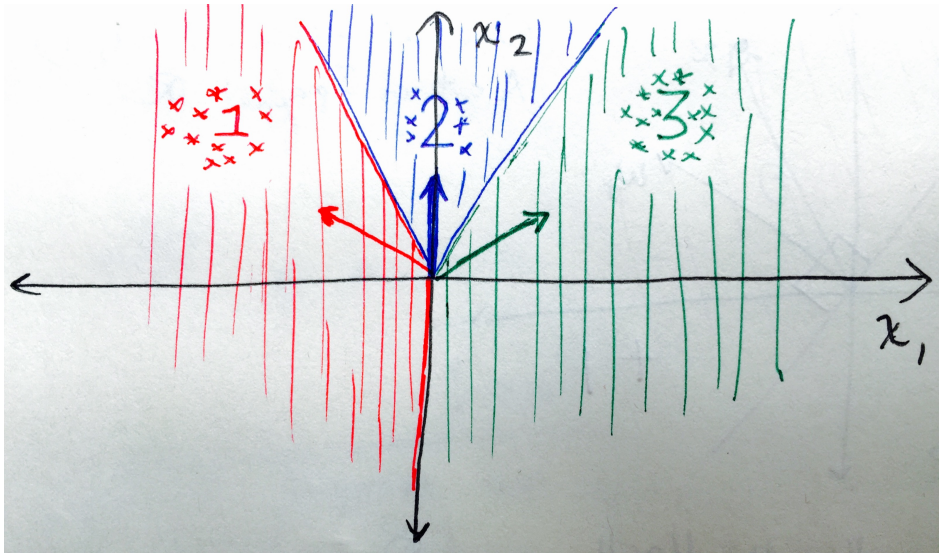
- **Linear Multiclass Hypothesis Space**

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle \mid w \in \mathbf{R}^d \right\}$$

- $\Psi(x, y)$ is a feature vector representing how much of a match y is for x.

- Final compatibility score must be extracted **linearly** from $\Psi(x, y)$.

Example: $\mathcal{X} = \mathbf{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$



- $w_1 = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$, $w_2 = (0, 1)$, $w_3 = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$
- Prediction function: $(x_1, x_2) \mapsto \arg\max_{i \in \{1,2,3\}} \langle w_i, (x_1, x_2) \rangle$.
- How can we get this into the form $x \mapsto \arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle$

The Multivector Construction

- What if we stack w_i's together:

$$w = \left(\underbrace{-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_1}, \underbrace{0, 1}_{w_2}, \underbrace{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_3} \right)$$

- And then do the following:

$$\begin{aligned} \Psi(x, 1) &:= (x_1, x_2, 0, 0, 0, 0) \\ \Psi(x, 2) &:= (0, 0, x_1, x_2, 0, 0) \\ \Psi(x, 1) &:= (0, 0, 0, 0, x_1, x_2) \end{aligned}$$

- Then $\langle w, \Psi(x, y) \rangle = \langle w_y, x \rangle$, which is what we want.

Natural Language Processing Example

- $\mathcal{X} = \{\text{All possible words}\}$.
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,ADVERB,ARTICLE,PREPOSITION}\}$.
- Features of $x \in \mathcal{X}$:
  - $\mathcal{F}$=[The word itself], ENDS_IN_ly, ENDS_IN_ness, ...}
- $\Psi(x, y) = (\psi_1(x, y), \psi_2(x, y), \psi_3(x, y), \ldots, \psi_d(x, y))$:

$$\begin{aligned}
\psi_1(x, y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x, y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x, y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x, y) &= 1(x \text{ ENDS_IN_ly AND } y = \text{ADVERB}) \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}$$

- After training, what would corresponding w_1, w_2, w_3, w_4 would look?

NLP Example: How does it work? (To be totally clear)

- $\Psi(x,y) = (\psi_1(x,y), \psi_2(x,y), \psi_3(x,y), \ldots, \psi_d(x,y)) \in \mathbf{R}^d$:

$$\begin{aligned} \psi_1(x,y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\ \psi_2(x,y) &= 1(x = \text{run AND } y = \text{NOUN}) \\ &\vdots \quad \vdots \quad \vdots \end{aligned}$$

- After training, we've learned $w \in \mathbf{R}^d$. Say $w = (5, 3, 1, 4, \ldots)$
- To predict label for $x = \text{apple}$, we compute scores for each $y \in \mathcal{Y}$:

$$\begin{gathered} \langle w, \Psi(\text{apple}, \text{NOUN}) \rangle \\ \langle w, \Psi(\text{apple}, \text{VERB}) \rangle \\ \langle w, \Psi(\text{apple}, \text{ADVERB}) \rangle \\ \vdots \end{gathered}$$

- Predict class that gives highest score.

TF-IDF Features for News Article Classification

- $\mathcal{X} = \{\text{news articles}\}$
- $\mathcal{Y} = \{\text{politics, sports, entertainment, world news, local news}\}$ [TOPICS]
- Want to use the words in article x to predict topic y.
- The **Term-Frequency** of word w in document x, denoted

$$TF(w, x),$$

is the numer of times word w occurs in document x.

TF-IDF Features for News Article Classification

- The **Document-Frequency** of word w in class y, denoted

$$DF(w, y),$$

  is the numer of documents contain word w NOT in topic y.
- The TF-IDF feature for word w is then defined as

$$\psi_w(x, y) = TF(w, x) \log\left(\frac{m}{DF(w, y)}\right),$$

  where m is the total number of documents in training set.
- (NOTE: There are **many** other variations of TF-IDF).

TF-IDF: Things to note

- Suppose we d words in our vocabulary and k topic classes.
- Suppose we have a TF-IDF feature for each word (and no other features).
- What's the dimension of $\Psi(x, y)$?
- We have one TF-IDF for each word.
- Recall our multivector-style NLP features:

$$\begin{aligned} \psi_2(x, y) &= 1(x = \text{run AND } y = \text{NOUN}) \\ \psi_3(x, y) &= 1(x = \text{run AND } y = \text{VERB}) \end{aligned}$$

- If made this "TF-IDF" style, it would like

$$\psi_{x=\text{run}}(x, y) = 1(x = \text{run}) \times (\text{compatibility of "run" with class } y)$$

Another Approach: Use Label Features

- What if we have a very large number of classes?
- Make features for the classes.
- Common in advertising
  - $\mathcal{X}$: User and user context
  - $\mathcal{Y}$: A large set of banner ads
- Suppose user x is shown many banner ads.
- We want to predict which one the user will click on.
- Possible features:

$$\begin{aligned}
\psi_1(x,y) &= 1(x \text{ interested in sports AND } y \text{ relevant to sports}) \\
\psi_2(x,y) &= 1(x \text{ is in target demographic group of } y) \\
\psi_3(x,y) &= 1(x \text{ previously clicked on ad from company sponsoring } y)
\end{aligned}$$

Linear Multiclass SVM

The Margin for Multiclass

- Training set: $(x_1, y_1), \ldots, (x_n, y_n)$
- Define a "**margin**" between correct class and each other class:

Definition

The **margin** of prediction function h when predicting y on the ith example (x_i, y_i) is

$$m_{i,y}(h) = h(x_i, y_i) - h(x_i, y).$$

- Want $m_{i,y}(h)$ to be large and positive for all $y \neq y_i$.
- For our linear hypothesis space, margin is

$$m_{i,y}(w) = \langle w, \Psi(x_i, y_i) \rangle - \langle w, \Psi(x_i, y) \rangle$$

Multiclass SVM with Hinge Loss

- Recall binary SVM (without bias term):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \left(1 - \underbrace{y_i w^T x_i}_{\text{margin}}\right)_+ .$$

- [Recall $(x)_+ = \max(0, x)$.]
- Multiclass SVM (Version 1):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max_y \left(1 - m_{i,y}(w)\right)_+$$

where $m_{i,y}(w) = \langle w, \Psi(x_i, y_i)\rangle - \langle w, \Psi(x_i, y)\rangle$.

Class-Sensitive Loss

- In multiclass, some misclassifications may be worse than others.
- Rather than 0/1 Loss, we may be interested in a more general loss

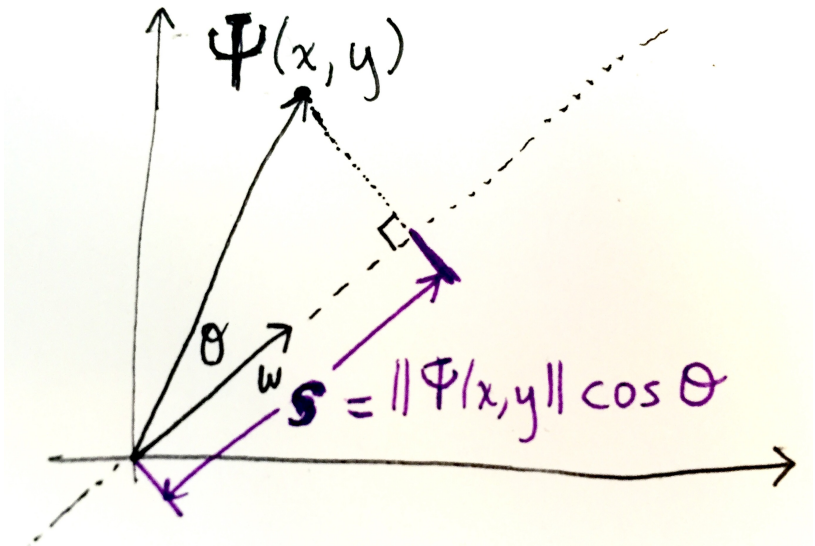
$$\Delta : \mathcal{Y} \times \mathcal{A} \to \mathbf{R}^{\geqslant 0}$$

- We can use this loss as our **target margin** SVM.
- Multiclass SVM (Version 2):

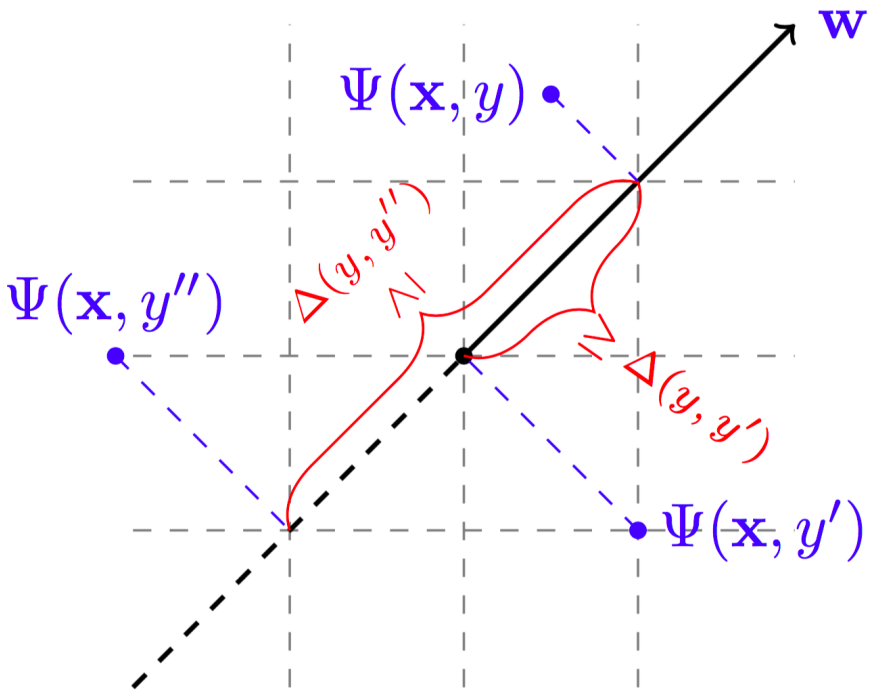
$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^{n} \max_{y} \left(\Delta(y_i, y) - m_{i,y}(w)\right)_+$$

Geometric Interpretation

- Prediction is given by $\arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle$.
- Note it's unchanged if we replace w by $w/\|w\|$.
- For simplicity, let's assume $\|w\| = 1$.
- Then score function $\langle w, \Psi(x, y) \rangle = \|\Psi(x, y)\| \cos\theta = \text{Proj}_w \Psi(x, y)$.



Geometric Interpretation



- Ψ maps each $x \in \mathcal{X}$ to $|\mathcal{Y}|$ different vectors in $\mathbf{R}^d$.
- For example (x, y), we want margins for all $y' \neq y$ to exceed target margin:

$$\langle w, \Psi(x, y) \rangle - \langle w, \Psi(x, y') \rangle \quad \geqslant \quad \Delta(y, y')$$

Figure from Section 17.2.4 from Shalev-Schwartz and Ben-David's *Understanding Machine Learning*

Introduction to Structured Prediction

Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, \text{"."}\}$
- $\mathcal{P} = \{\text{START}, \text{Pronoun}, \text{Verb}, \text{Noun}, \text{Adjective}\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]
- $\mathcal{Y} = \mathcal{P}^n$, $n = 1, 2, 3, \ldots$[Part of speech sequence of any length]

Structured Prediction

- A **structured prediction** problem is a multiclass problem in which $\mathcal{Y}$ is very large, but has (or we assume it has) a certain structure.
- For POS tagging, $\mathcal{Y}$ grows exponentially in the length of the sentence.
- The **structure** in POS labels is that labels that are far apart in the sentence are independent (we assume).

Local Feature Functions: Type 1

- A "type 1" **local feature** only depends on
  - the label at a single position, say y_i (label of the ith word) and
  - x at any position
- Example:

$$\begin{aligned}
\phi_1(i, x, y_i) &= 1(x_i = \text{He})1(y_i = \text{Pronoun}) \\
\phi_2(i, x, y_i) &= 1(x_i = \text{eats})1(y_i = \text{Noun}) \\
\phi_3(i, x, y_i) &= 1(x_{i-1} = \text{He})1(x_i = \text{eats})1(y_i = \text{Verb})
\end{aligned}$$

Local Feature Functions: Type 2

- A "type 2" **local feature** only depends on
  - the labels at 2 consecutive positions: y_{i-1} and y_i
  - x at any position
- Example:

$$\begin{aligned}\theta_1(i, x, y_{i-1}, y_i) &= 1(y_{i-1} = \text{Pronoun})1(y_i = \text{Verb}) \\ \theta_2(i, x, y_{i-1}, y_i) &= 1(y_{i-1} = \text{Verb})1(y_i = \text{Verb})\end{aligned}$$

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector**:

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(i, x, y_i), \phi_2(i, x, y_i), \ldots, \\ &\quad \theta_1(i, x, y_{i-1}, y_i), \theta_2(i, x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- **Local compatibility score** for (x, y) at position i is $\langle w, \Psi_i(x, y_{i-1}, y_i) \rangle$.

Sequence Compatibility Score

- The **compatibility score** for the pair of sequences (x, y) is the sum o fthe local compatibility scores:

$$\begin{aligned} & \sum_i \langle w, \Psi_i(x, y_{i-1}, y_i) \rangle \\ = & \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i) \right\rangle \\ = & \langle w, \Psi(x, y) \rangle , \end{aligned}$$

where we define the sequence feature vector by

$$\Psi(x, y) = \sum_i \Psi_i(x, y_{i-1}, y_i).$$

- So we see this is a special case of linear multiclass prediction.

Sequence Target Loss

- How do we assess the loss for prediction sequence y' for example (x, y)?
- **Hamming loss** is common:

$$\Delta(y, y') = \frac{1}{|y|} \sum_{i=1}^{|y|} 1(y_i \neq y_i')$$

- Could generalize this as

$$\Delta(y, y') = \frac{1}{|y|} \sum_{i=1}^{|y|} \delta(y_i, y_i')$$

What remains to be done?

- To compute predictions, we need to find

$$\arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle .$$

- This is straightforward for $|\mathcal{Y}|$ small.
- Now $|\mathcal{Y}|$ is exponentially large.
- Because Ψ breaks down into local functions only depending on 2 adjacent labels,
  - we can solve this efficiently using dynamic programming.
  - (Similar to Viterbi decoding.)
- Learning can be done witih SGD and a similar dynamic program.