Language and Statistics II

Lecture 15: Going Discriminative

Noah Smith

Lecture Outline

- Perceptron for training structured models
- Loss functions for structures
- Boosting
- Maximum margin training: intuition and the big idea

Beware

- In this lecture, I won't say much about the form of the model.
- Assume discrete inputs and discrete outputs.
- If you like, think of parsing or tagging.
- Score = $\exp \mathbf{f}(x, y) \cdot \mathbf{w}$ unless otherwise noted.
- Nitpicky point, for correctness: assume $\forall x, \forall y$, $f_0(x, y) = 1$. (w_0 is a bias weight.)
- Subroutines you already know and love:
  - Sum scores over all *y*'s for a given *x*
  - Find maximum-scoring *y* for a given *x*

General Idea

- MLE: max $p(x, y) = p(x)p(y|x)$
- MCLE: max $p(y|x)$

(Why model x?)

Indeed, why estimate densities *at all*?

Perceptron for Structured Models (Collins, 2002)

Unlike other training methods we have seen

- (Maximum likelihood
- Maximum condtional likelihood)

the perceptron does not explicitly maximize a function.

Instead, it simply tries to learn a model that separates the **right answer** from the wrong answers.

It's also really simple.

Perceptron for Structured Models (Collins, 2002)

- "Global linear model" over structures.
  - Prediction: $$\hat{y} = \arg\max_{y \in \mathrm{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{w}$$
  - *x* is the input
  - *y* is the output
  - GEN enumerates all possible *y* for a given *x*
  - **f** maps (input, output) to $\mathbb{R}^d$
  - **w** is the weight vector ($\mathbb{R}^d$)
- Learning/training/estimation: pick **w**

Perceptron for Structured Models (Collins, 2002)

- Nothing has changed! Just like log-linear models!
- Examples:
  - x is a sentence, y is a POS tag sequence
  - x is a sentence, y is an NP bracketing
  - x is a sentence, y is a parse tree
  - x is two sentences, y is a word alignment
  - x is a sentence, y is its translation

Perceptron for Structured Models (Collins, 2002)

- Input: (x_i, y_i) for $i = 1 \ldots n$; T
- Output: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$

for $t = 1 \ldots T$

for $i = 1 \ldots n$

$y_{\text{hyp}} \leftarrow \text{argmax}_y\ \mathbf{f}(x_i, y)\cdot\mathbf{w}$

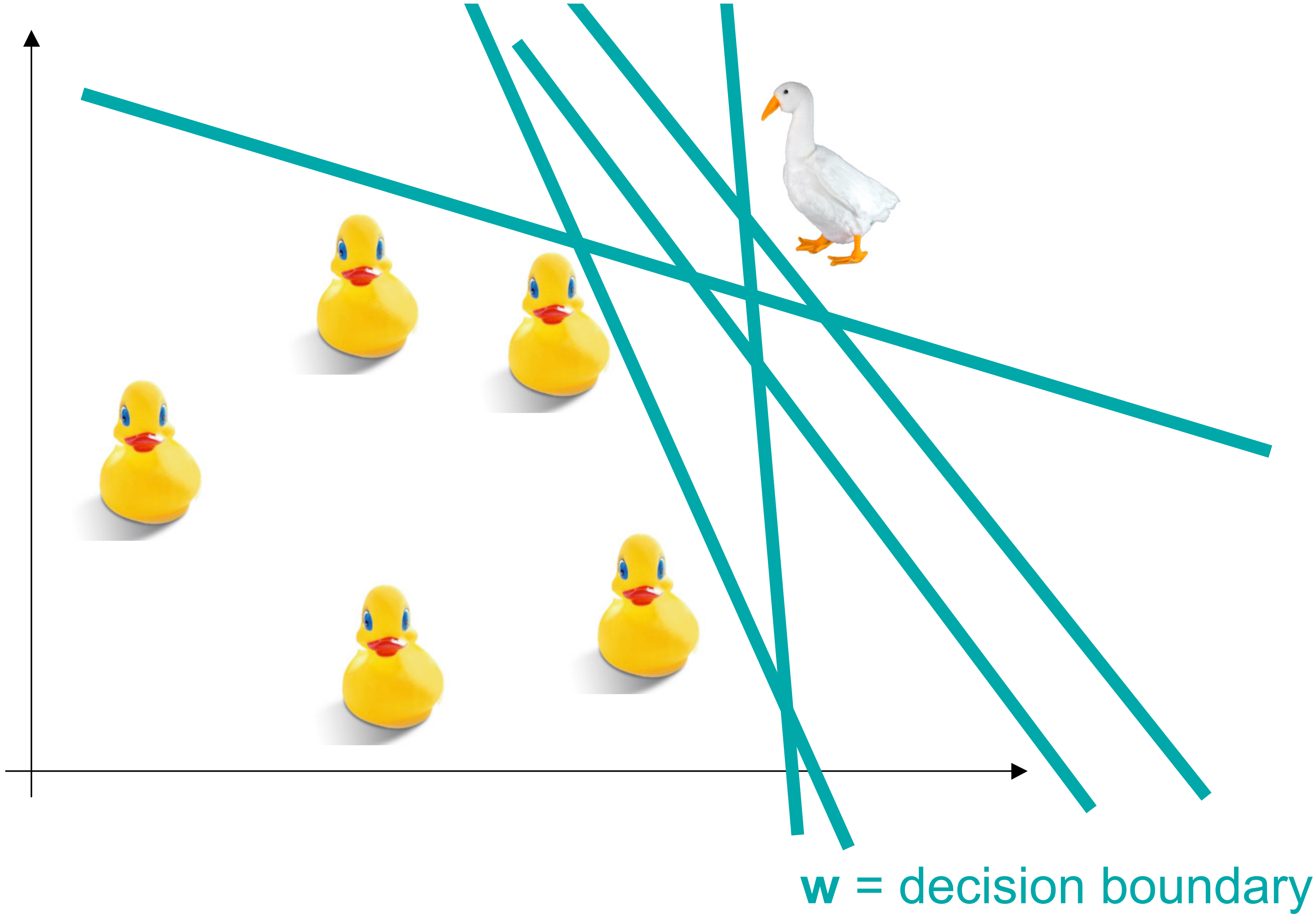
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y_{\text{hyp}})$

return $\mathbf{w}$

Intuition Behind Perceptron Updates

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y_{\text{hyp}})$$

- If $y_i = y_{\text{hyp}}$, no change.
- Otherwise, for each f_j:
  - If $f_j(x_i, y_i) > f_j(x_i, y_{\text{hyp}})$, **increase** w_j
  - Else if $f_j(x_i, y_i) < f_j(x_i, y_{\text{hyp}})$, **decrease** w_j
  - Else f_j makes no difference on this example, so don't change w_j



Duck, Duck, Goose

Theorems

- If the training data (x_i, y_i) for i = 1 … n are **separable** with margin m, and

  $R \geq || \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y) ||$ for all i = 1 … n, y in GEN(x_i) \ {y_i}

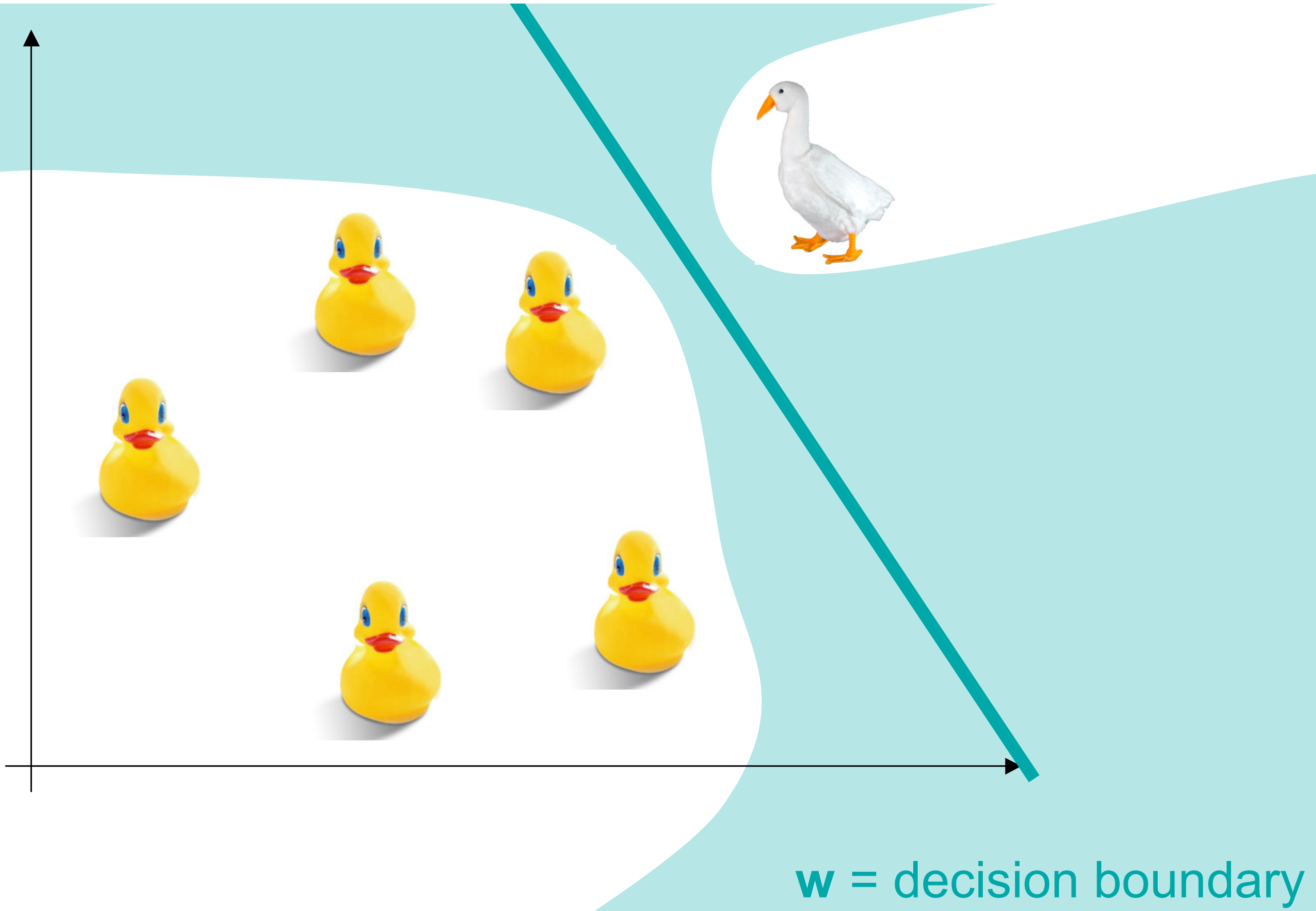
  then

  $$\#\text{mistakes} \leq R^2 / m^2$$

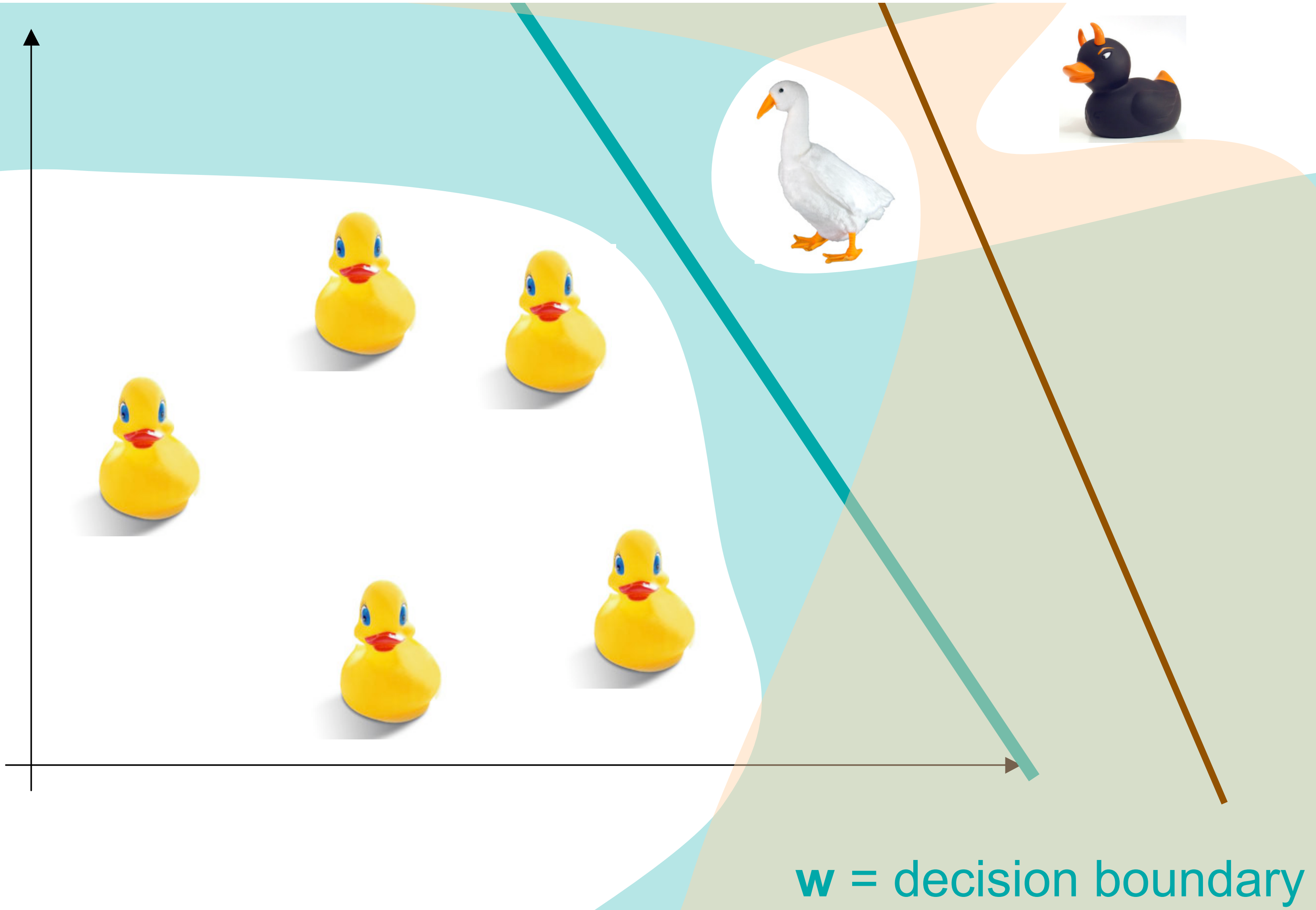
  This extends a classification theorem by Freund & Schapire (1999).

Comments

- What if the training data are **not** separable?
  - See Collins (2002) for the bound. Not as tight!
  - Does it matter? Are NLP data separable?
- How long does it take?
  - You decide: *T* (finite convergence time for separable data guaranteed)
  - Remember: no function to optimize!
- Dealing with oscillation:
  - Averaging: average each iterate of **w**
  - Voting: keep each iterate of **w**, let them all vote



Voted Perceptron



Voted Perceptron

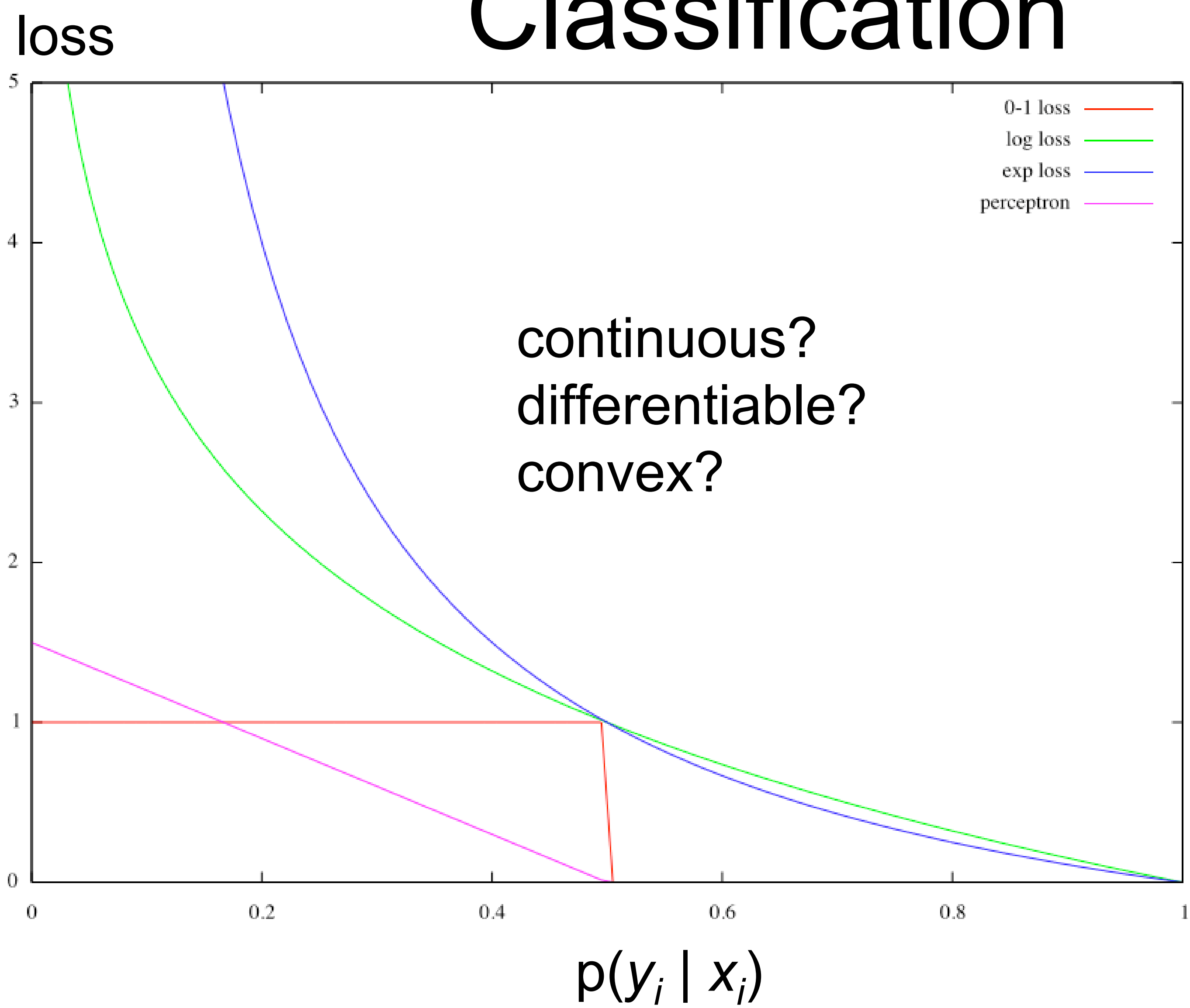
Estimation Methods: A Guide

Name	Features?	Training?	Decoding
Maximum likelihood	Must fit stochastic "story"	Count & Normalize®	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Maximum Conditional likelihood	Relatively local	Convex optimization; $\sum_y e^{\mathbf{f}(x,y)\cdot\mathbf{w}}$ (sum over y)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Perceptron	Relatively local	Perceptron; $\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$

Loss Functions

- At this point it behooves us to talk about loss functions.
- Maximum conditional likelihood can be said to **minimize** -log p(y_i | x_i).
  - This is sometimes called the **log loss**.
- There are many other loss functions!
  - Some are easier to minimize than others, and some have loftier goals than others.

Loss Functions for Binary Classification



Boosting and Exp Loss

- Usually refers to "AdaBoost," another learning algorithm (Freund & Schapire, 1995).
  - The short version: aims* to minimize **exp loss**:

$$\sum_i \sum_{y \in \mathrm{GEN}(x_i)} \exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y) - \mathbf{w} \cdot \mathbf{f}(x_i, y_i)\right)$$

$$= \frac{1}{p_{\mathbf{w}}(y_i \mid x_i)} - 1$$

*Actually minimizes a bound.

- Exp loss is an upper bound on **ranking loss** (the number of alternative *y* that beat y_i).

Estimation Methods: A Guide

Name	Features?	Training?	Decoding
Maximum likelihood	Must fit stochastic "story"	Count & Normalize®	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Maximum Conditional likelihood	Relatively local	Convex optimization; $\sum_y e^{\mathbf{f}(x,y)\cdot\mathbf{w}}$ (sum over y)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Perceptron	Relatively local	Perceptron; $\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Minimum exp-loss	Relatively local	Convex optimization or boosting; $\sum_y e^{\mathbf{f}(x,y)\cdot\mathbf{w}}$ (sum over y)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$

See Also

- Altun, Johnson, & Hoffman (EMNLP 2003)
  - Comparison among {log loss, exp loss} × {sequence loss, pointwise loss}.
  - Sequence loss: pay for getting the whole tag sequence or tree wrong
  - Pointwise loss: pay for each word you got wrong

From Loss to Margin

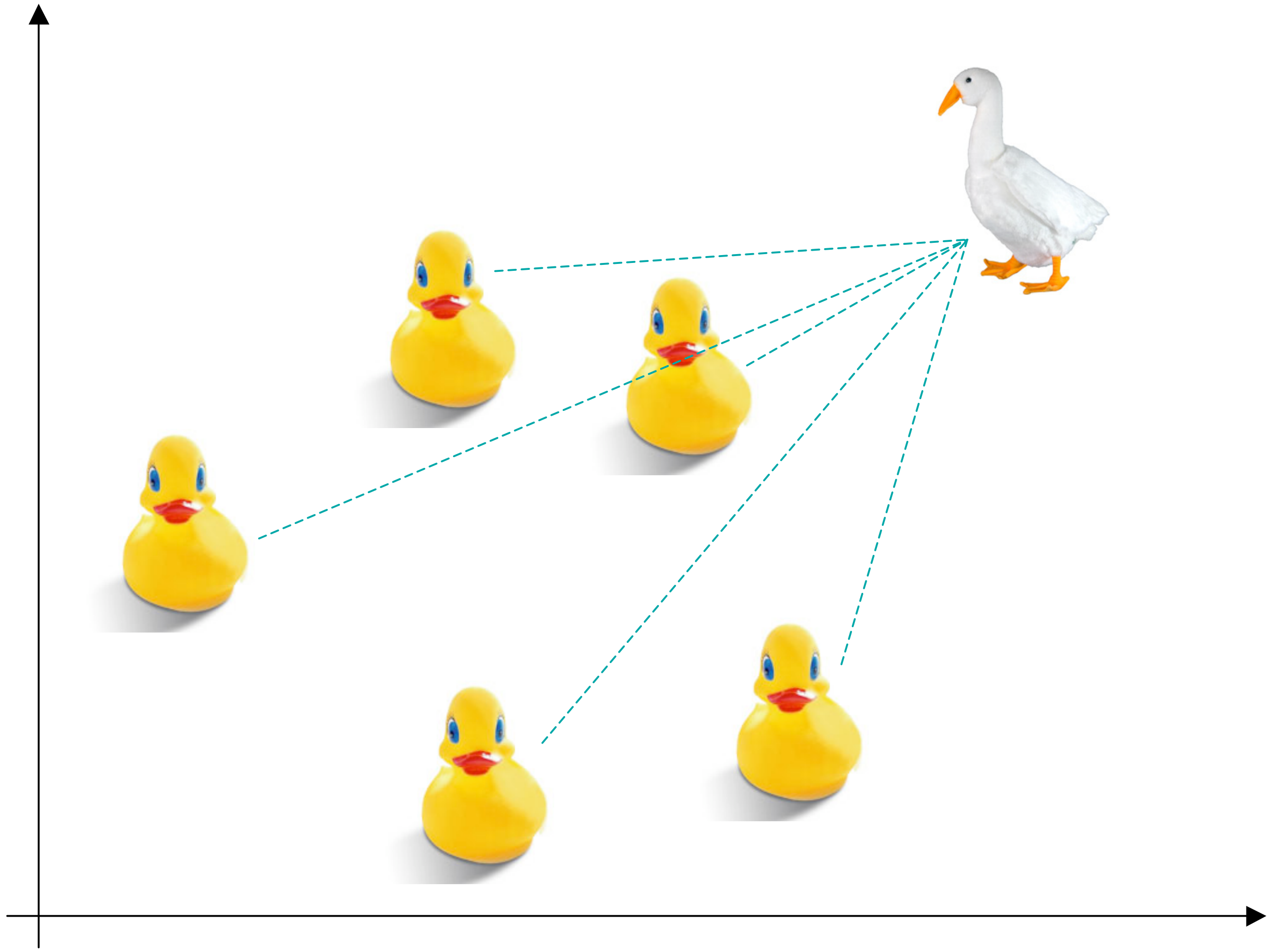
- You can think of loss functions as trying to improve the score of (x_i, y_i) as compared to scores of alternative outputs with x_i.

exp loss

$$\max_{\mathbf{w}} - \sum_i \sum_{y \in \mathrm{GEN}(x_i)} \exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y) - \mathbf{w} \cdot \mathbf{f}(x_i, y_i)\right)$$

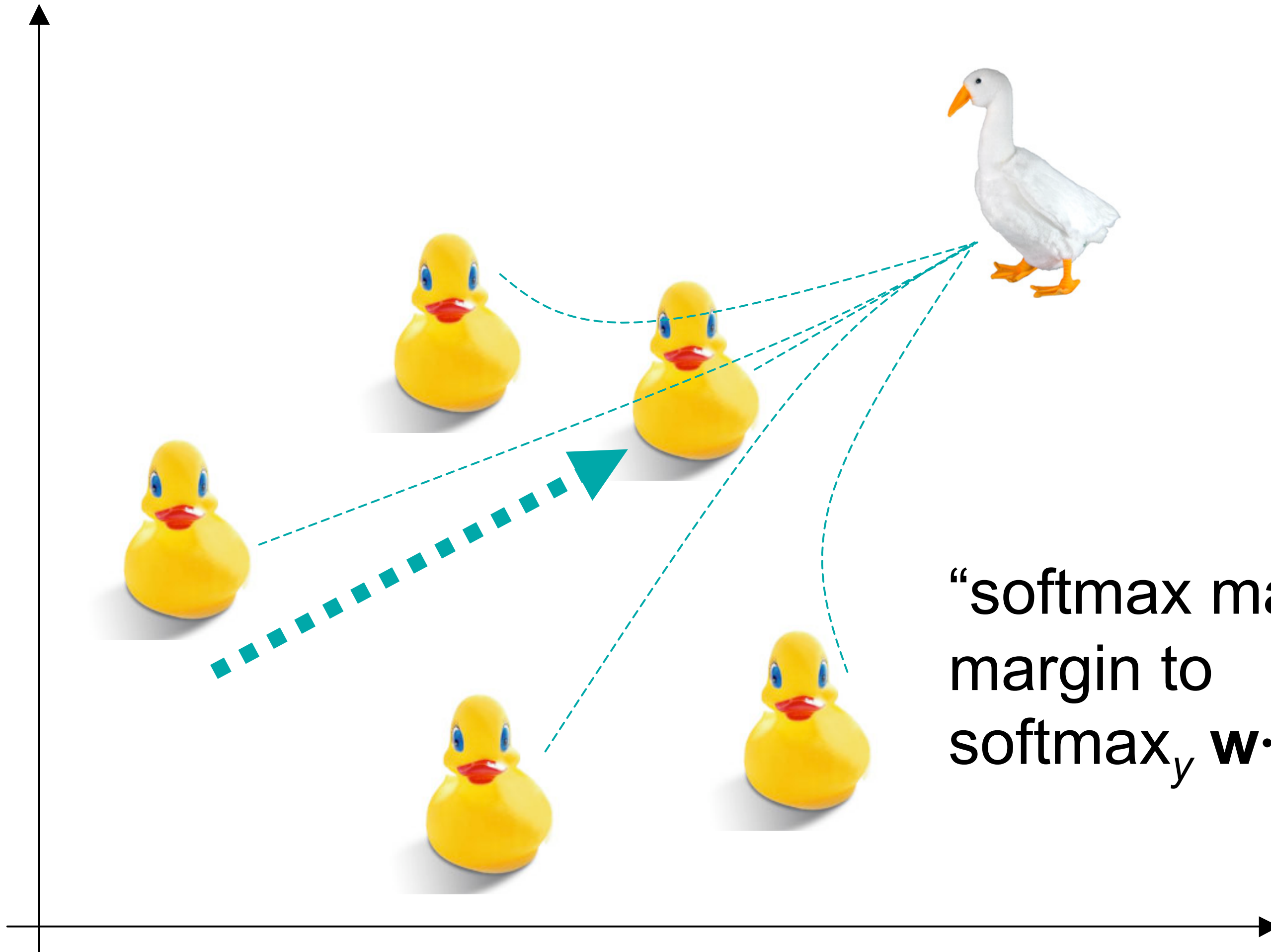
log loss

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w} \cdot \mathbf{f}(x_i, y_i) - \log \sum_{y \in \mathrm{GEN}(x_i)} \exp \mathbf{w} \cdot \mathbf{f}(x_i, y) \right)$$



log loss

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w} \cdot \mathbf{f}(x_i, y_i) - \log \sum_{y \in \mathrm{GEN}(x_i)} \exp \mathbf{w} \cdot \mathbf{f}(x_i, y) \right)$$

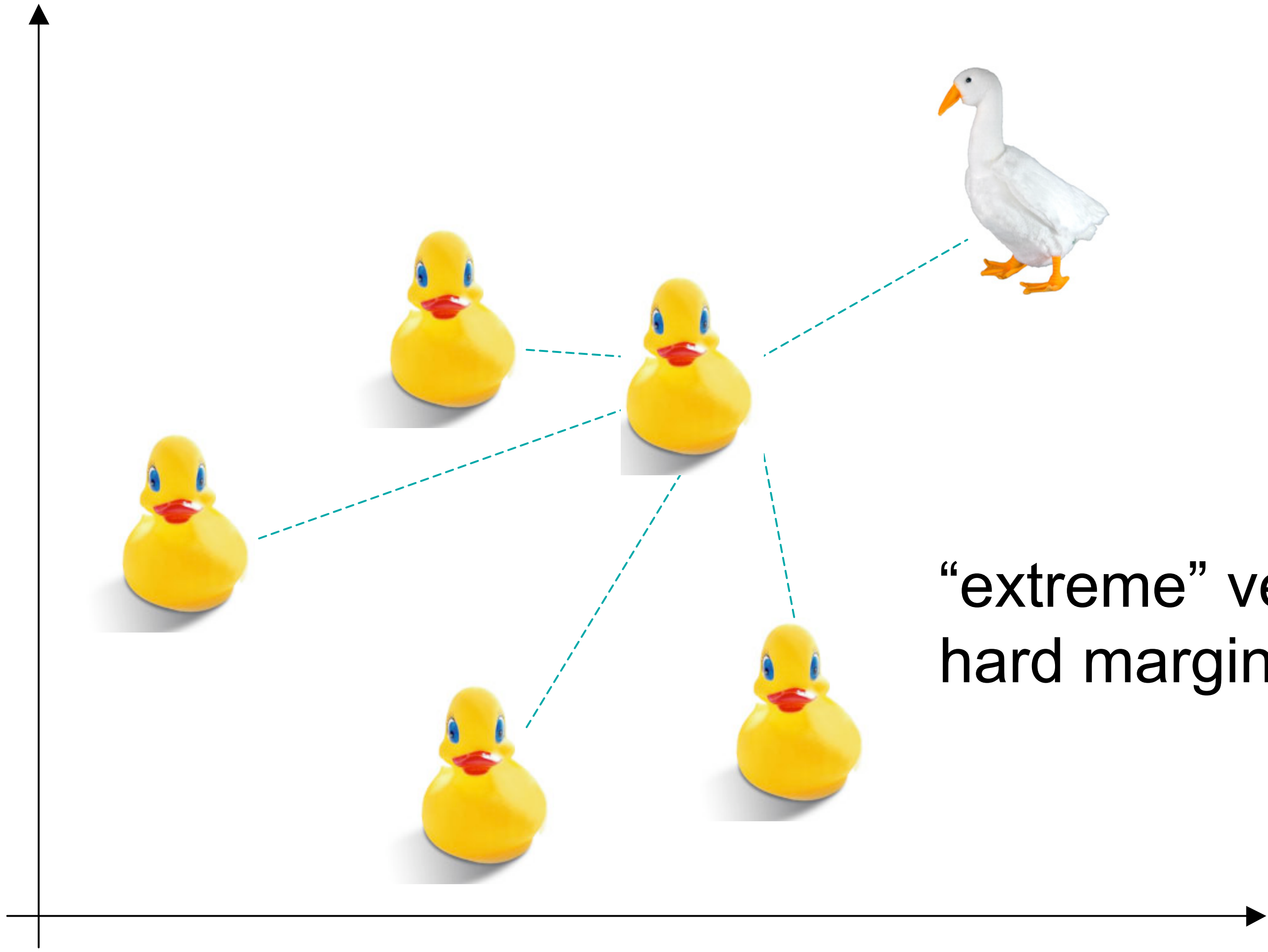


"softmax margin" = margin to softmax$_y$ $\mathbf{w}\cdot\mathbf{f}(x_i, y)$

$$\text{softmax}(a,b) = \log\left(e^a + e^b\right) \xrightarrow{|a-b|\to\infty} \max(a,b)$$

log loss

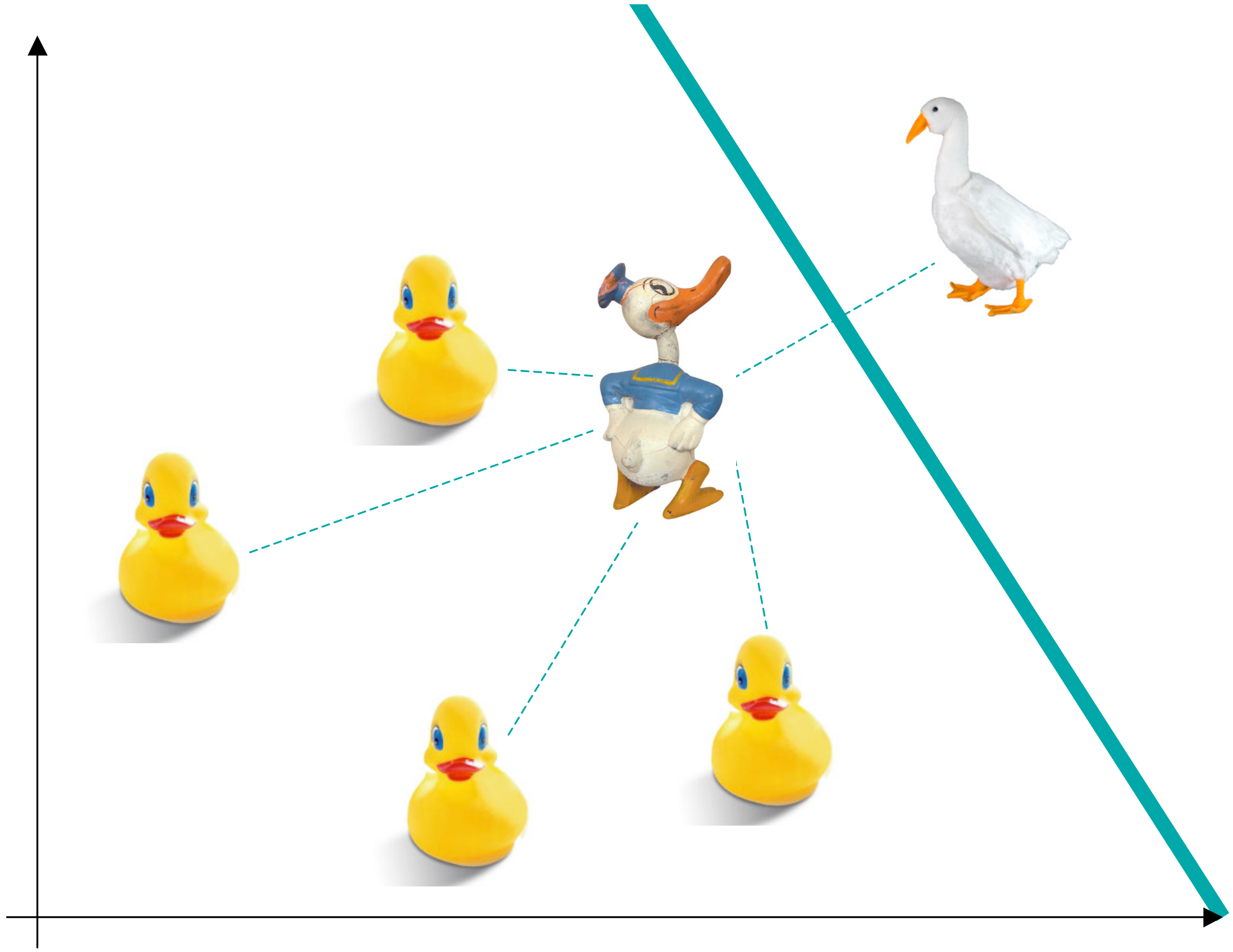
$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w}\cdot\mathbf{f}(x_i,y_i) - \log \sum_{y\in \text{GEN}(x_i)} \exp \mathbf{w}\cdot\mathbf{f}(x_i,y) \right)$$



"extreme" version:
hard margin

0-1 loss

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w} \cdot \mathbf{f}(x_i, y_i) - \max_{y \in \mathrm{GEN}(x_i)} \mathbf{w} \cdot \mathbf{f}(x_i, y) \right)$$



0-1 loss

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w} \cdot \mathbf{f}(x_i, y_i) - \max_{y \in \mathrm{GEN}(x_i)} \mathbf{w} \cdot \mathbf{f}(x_i, y) \right)$$

Desiderata

- Don’t really want 0-1 loss
  - Tagging accuracy
  - Parseval accuracy
  - MT evaluation scores
- Core idea of maximum margin methods:

Maximize the (hard) margin under a particular **loss** function.

(Multiclass) Support Vector Machines

First form:

Note constraint on **w**. This prevents us from cheating by using really big weights. (Can think of it as built-in regularization.)

$$\max_{\mathbf{w}: \frac{1}{2}\mathbf{w}\cdot\mathbf{w} \leq 1} \gamma$$

$$s.t.\ \forall i, \forall y \in \mathrm{GEN}(x_i),$$

$$\mathbf{w}\cdot\mathbf{f}(x_i, y_i) - \mathbf{w}\cdot\mathbf{f}(x_i, y) \geq \gamma\ell(y, y_i; x_i)$$

Second form: change of variable.

Note that the objective is quadratic (indeed, psd!), and the constraints are linear.

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}\cdot\mathbf{w}$$

$$s.t.\ \forall i, \forall y \in \mathrm{GEN}(x_i),$$

$$\mathbf{w}\cdot\mathbf{f}(x_i, y_i) - \mathbf{w}\cdot\mathbf{f}(x_i, y) \geq \ell(y, y_i; x_i)$$

Estimation Methods: A Guide

Name	Features?	Training?	Decoding
Maximum likelihood	Must fit stochastic "story"	Count & Normalize®	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Maximum Conditional likelihood	Relatively local	Convex optimization; $\sum_y e^{\mathbf{f}(x,y)\cdot\mathbf{w}}$ (sum over y)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Perceptron	Relatively local	Perceptron; $\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Minimum exp-loss	Relatively local	Convex optimization or boosting; $\sum_y e^{\mathbf{f}(x,y)\cdot\mathbf{w}}$ (sum over y)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$
Maximum margin (I)	Relatively local	Quadratic program (exponentially many constraints)	$\max_y \mathbf{f}(x,y)\cdot\mathbf{w}$

Coming Soon

- Maximum margin training:
  - Allowing for nonseparable data
  - Hinge loss
  - Making maximum margin training tractable
    - Dual form
    - Factored dual form
  - Sparsity and support vectors
  - Examples on NL tasks
  - Kernels
- Discriminative methods in general:
  - Bringing in “global” features
  - Reranking