Lecture #3: Online Learning*

Janardhan Rao (Jana) Doppa

School of EECS, Washington State University

* Slides partly based on Koby Crammer

Formal setting – Classification

- **Instances** $\mathbf{x} \in \mathcal{X}$
  - emails
- **Labels** $y \in \mathcal{Y} = \{-1\ ;\ 1\}$
  - Spam vs. non-spam
- **Prediction rule** $f(\mathbf{x}) = \hat{y}$
  - Linear prediction rule
- **Loss** $\ell(\hat{y}, y) \in \mathbb{R}_+$
  - No. of mistakes

Predictions

- Continuous predictions : $f : \mathcal{X} \rightarrow \mathbb{R}$
  - Label $\text{sign}(f(\mathbf{x}))$
  - Confidence $|f(\mathbf{x})|$
- Linear Classifiers
  - Prediction :

$$\begin{aligned}\hat{y} &= \text{sign}(f(\mathbf{x})) \\ &= \arg\max_{y \in \mathcal{Y}} \mathbf{w} \cdot \Phi(\mathbf{x}, y) \\ &= \text{sign}(\mathbf{w} \cdot \mathbf{x})\end{aligned}$$

$$|f(\mathbf{x})| = |\mathbf{w} \cdot \mathbf{x}|$$

Loss Functions

- **Natural Loss:**
  - Zero-One loss

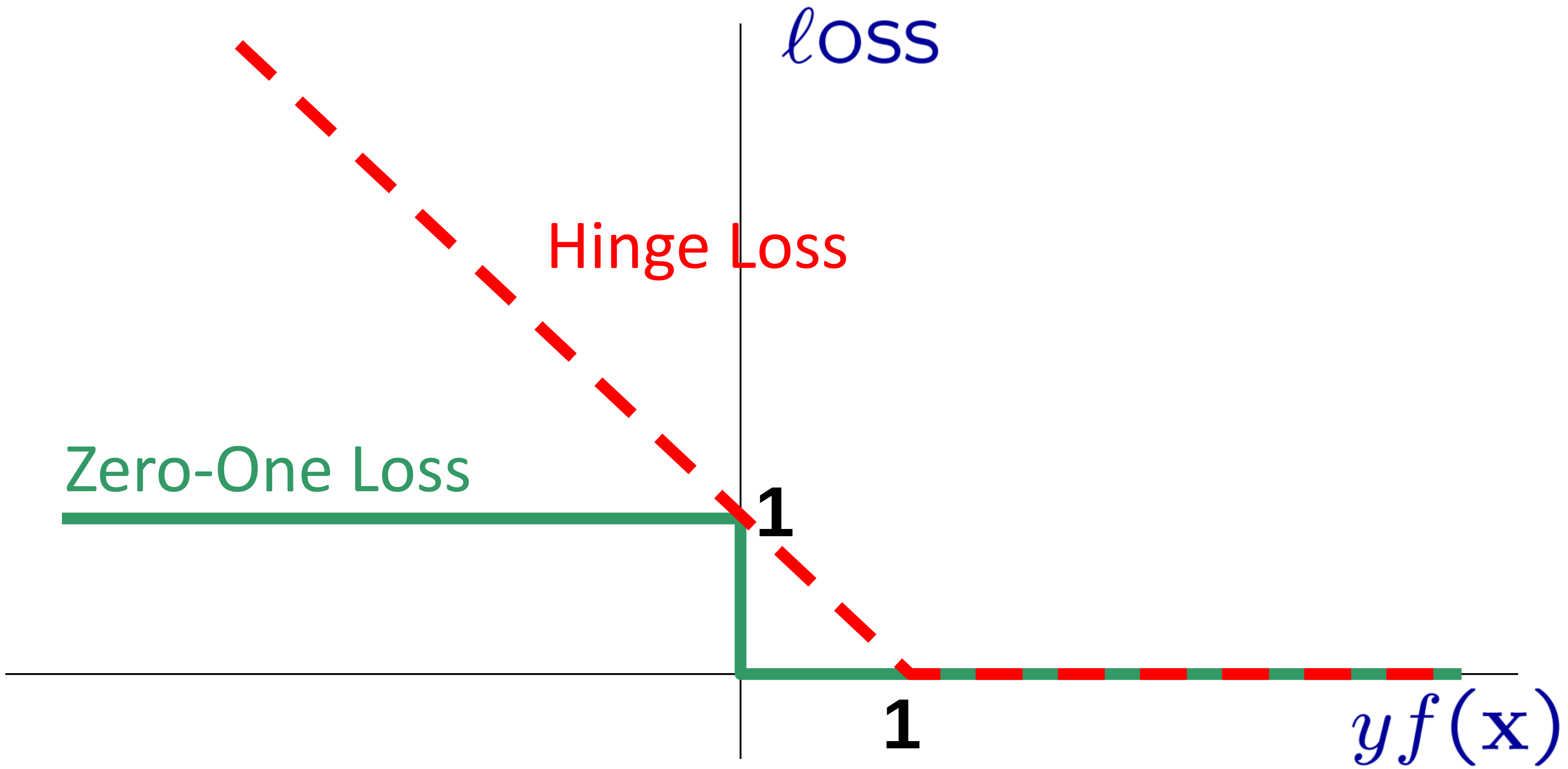
$$\ell(\widehat{y}, y) = \begin{cases} 0 & y = \widehat{y} \\ 1 & y \neq \widehat{y} \end{cases}$$

- **Real-valued-predictions loss:**
  - Hinge loss

$$\ell(\widehat{y}, y) = \max\{0, 1 - yf(\mathbf{x})\}$$

  - Exponential loss (Boosting)

Loss Functions



Online Framework

- Initialize Classifier $f_1(\mathbf{x})$
- Algorithm works in rounds $t = 1 \ldots T \ldots$
- On round t the online algorithm :
  - Receives an input instance $\mathbf{x}_t$
  - Outputs a prediction $f_t(\mathbf{x}_t) = \hat{y}_t$
  - Receives a feedback label y_t
  - Computes loss $\ell(\hat{y}_t, y_t)$
  - Updates the prediction rule $f_t \rightarrow f_{t+1}$
- Goal :
  - Suffer small cumulative loss $\sum_t \ell(\hat{y}_t, y_t)$

Margin

- ***Margin*** of an example $(\mathbf{x}_t, y_t)$ with respect to the classifier $\mathbf{w}_t$:

$$y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$$

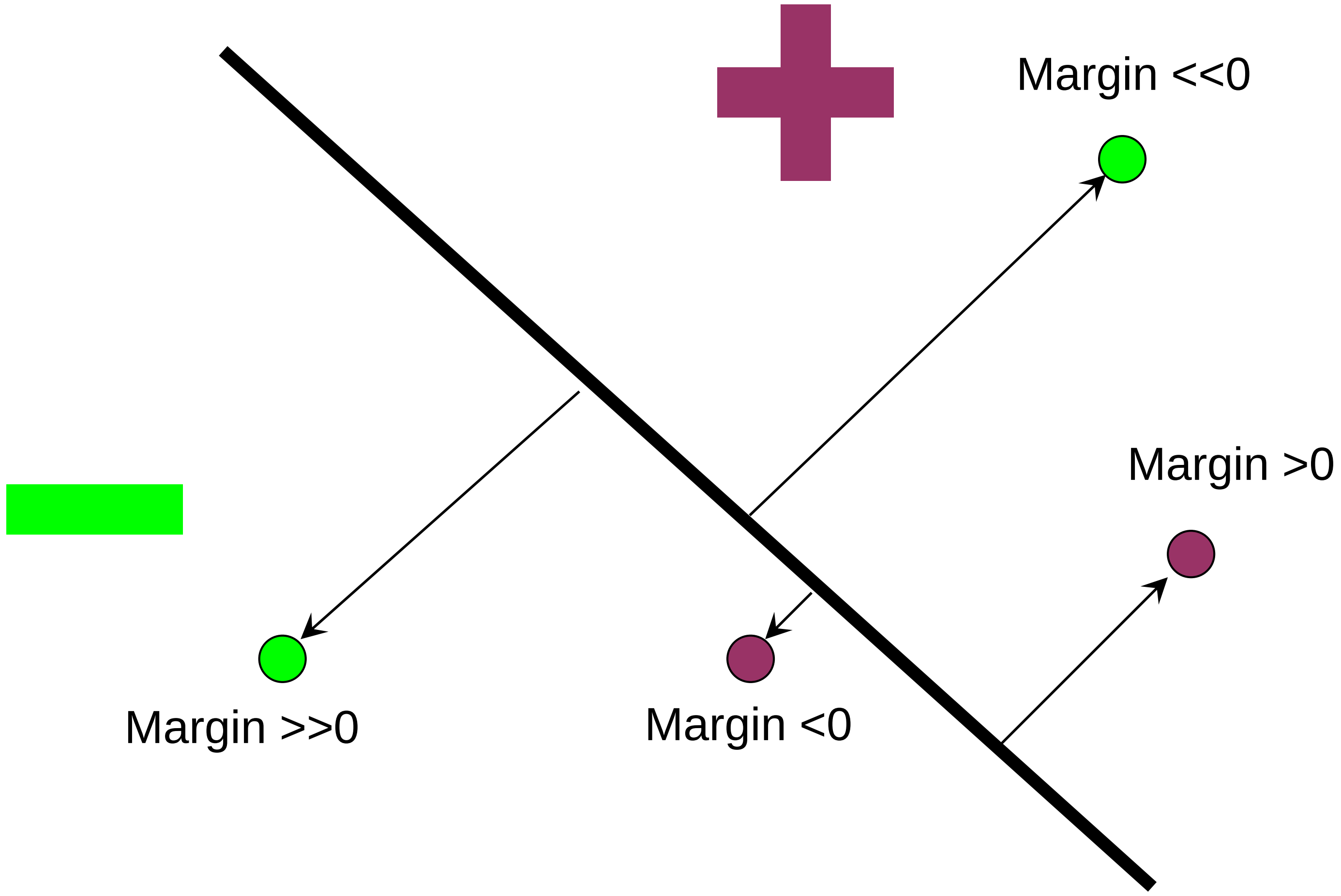
- Note :

$$y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0 \quad \Leftrightarrow \quad \ell_{01}(\hat{y}_t, \mathbf{x}_t) = 0$$

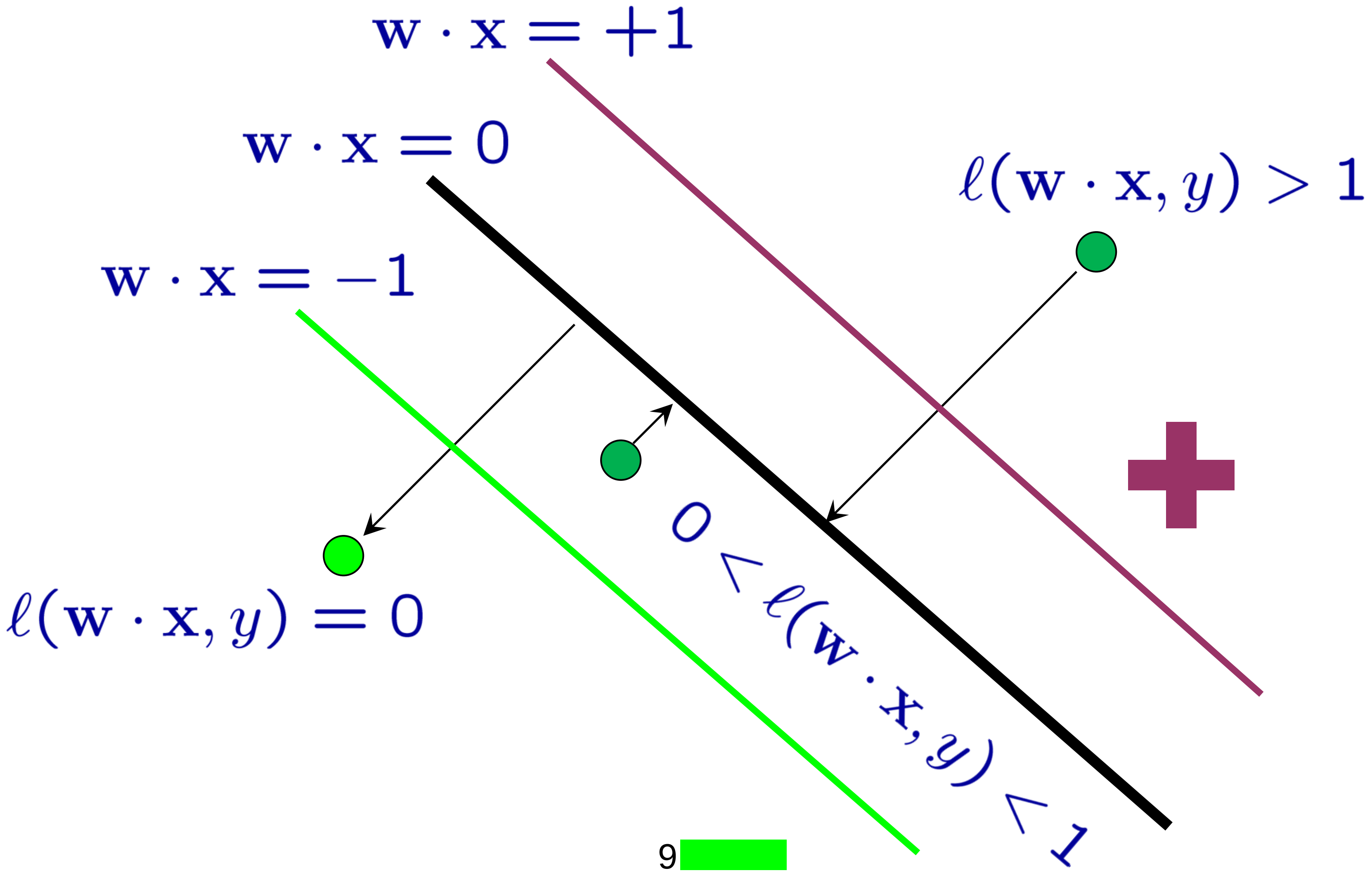
- The set $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_T, y_T)$ is separable iff there exists *u* such that

$$y_t(\mathbf{u} \cdot \mathbf{x}_t) > 0 \qquad \forall t$$

Geometrical Interpretation



Hinge Loss


$\mathbf{w} \cdot \mathbf{x} = +1$
$\mathbf{w} \cdot \mathbf{x} = 0$
$\ell(\mathbf{w} \cdot \mathbf{x}, y) > 1$
$\mathbf{w} \cdot \mathbf{x} = -1$
$0 < \ell(\mathbf{w} \cdot \mathbf{x}, y) < 1$
$\ell(\mathbf{w} \cdot \mathbf{x}, y) = 0$

Why Online Learning?

- Fast
- Memory efficient - process one example at a time
- Simple to implement
- Formal guarantees – Mistake bounds
- Online to Batch conversions
- No statistical assumptions
- Adaptive

Update Rules

- Online algorithms are based on an update rule which defines f_{t+1} from f_t (and possibly other information)
- **Linear Classifiers** : find $\mathbf{w}_{t+1}$ from $\mathbf{w}_t$ based on the input $(\mathbf{x}_t, y_t)$
- **Some Update Rules** :
  - Perceptron (Rosenblat)
  - ALMA (Gentile)
  - ROMMA (Li & Long)
  - NORMA (Kivinen et. al)
  - MIRA (Crammer & Singer)
  - EG (Littlestown and Warmuth)
  - Bregman Based (Warmuth)
  - CWL (Dredze et. al)

Design Principles of Algorithms

- If the learner suffers non-zero loss at any round, then we want to balance two goals:
  - **Corrective:** Change weights so that we don't make this error again
  - **Conservative:** Don't change the weights too much

$$d(w_{t+1}, w_t) + \eta L(y_t, w_{tt})$$

The Perceptron Algorithm

- If the learner suffers non-zero loss at any round, then we want to balance two goals:

$$d(w_{t+1}, w_t) + \eta L(y_t, w_{tt})$$

  - **Euclidean distance**
  - **Hinge loss**
  - Stochastic Gradient Descent (SGD)

The Perceptron Algorithm ($\eta = 1$)

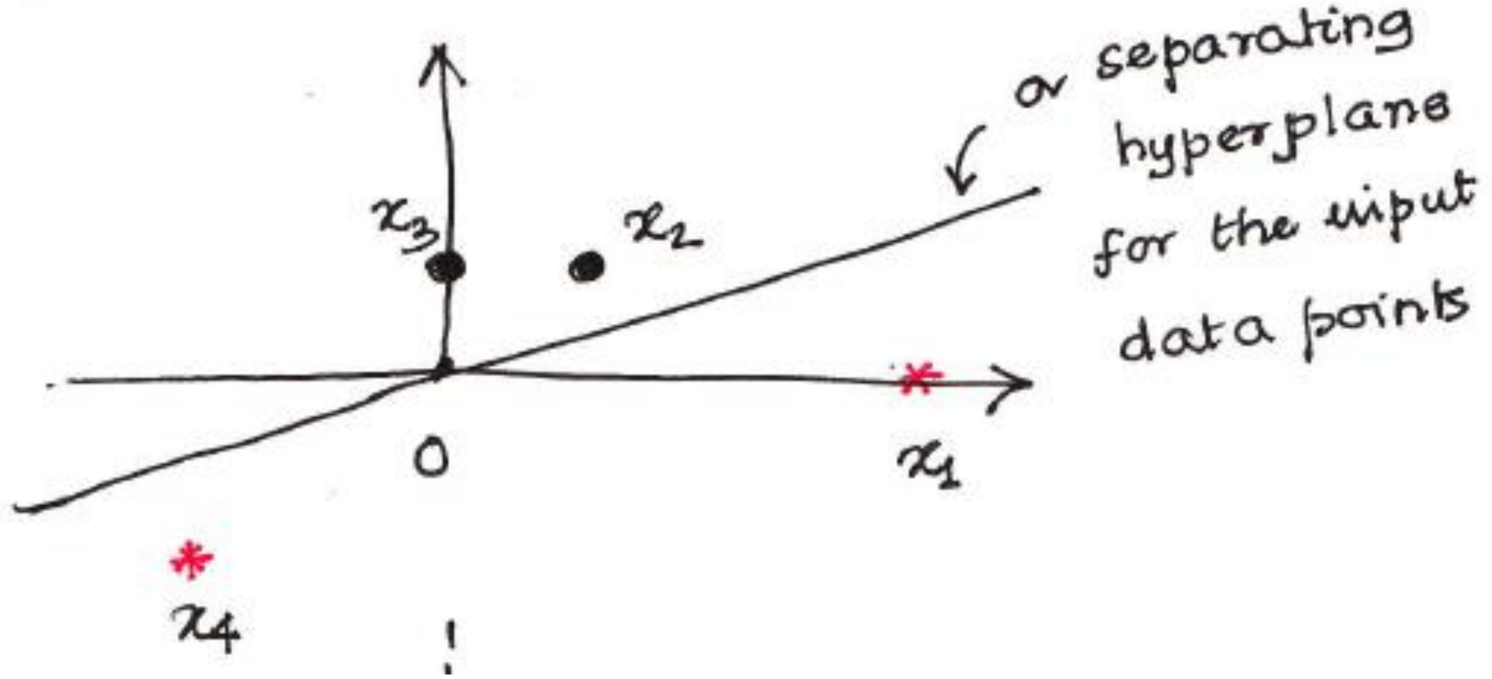
- If No-Mistake $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0$
  - Do nothing $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
- If Mistake $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0$
  - Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t\mathbf{x}_t$

The Perceptron Algorithm ($\eta = 1$)

- When mistake happens, what does the update do?
  - **If $y_t = 1$:** $\quad \boldsymbol{w_{t+1} = w_t + x_t}$
    - ✓ w_{t+1} moves "closer to" x_t OR
    - ✓ x_t moves towards the positive side of the decision boundary
  - **If $y_t = -1$:** $\quad \boldsymbol{w_{t+1} = w_t - x_t}$
    - ✓ w_{t+1} moves "away from" x_t OR
    - ✓ x_t moves towards the negative side of the decision boundary
- In both cases, we are moving towards the "correct solution"

Running Example #1

<u>Training Data:</u> ((4,0), 1), ((1,1), -1), ((0,1), -1), ((-2,-2), 1)



Running Example #1

Round 1:

* $w_1 = 0$

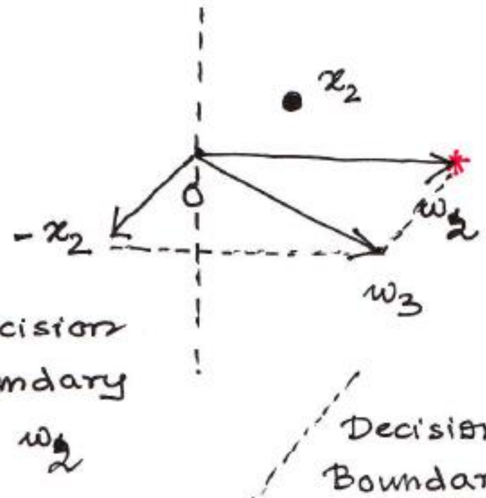
Training Data: $((4,0), 1)$, $((1,1), -1)$, $((0,1), -1)$, $((-2,-2), 1)$

* $y_t \langle w_t, x_t \rangle$ for $t = 1$

  $= 0$ as well.

* $w_2 = w_1 + y_1 x_1$

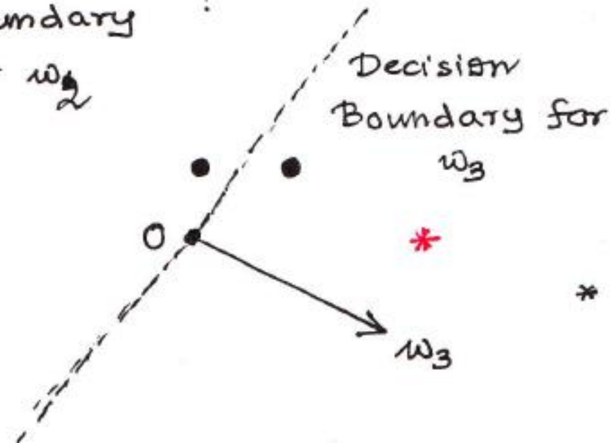
  $= (4,0)$



Round 2:

* $y_2 \langle w_2, x_2 \rangle < 0$
* So $w_3 = w_2 + y_2 x_2$

  $= (4,0) - (1,1) = (3,-1)$



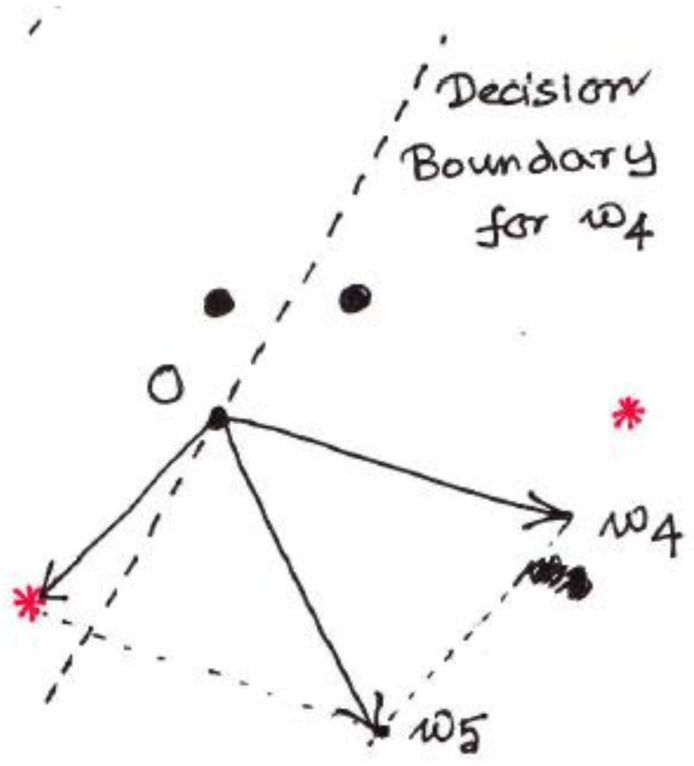
Round 3:

* $y_3 \langle w_3, x_3 \rangle > 0$

  Correct. $w_4 = w_3 = (3,-1)$

Running Example #1

Training Data: ((4, 0), 1), ((1, 1), −1), ((0, 1), −1), ((−2, −2), 1)



Round 4:

* $y_4 \langle w_4, x_4 \rangle < 0$

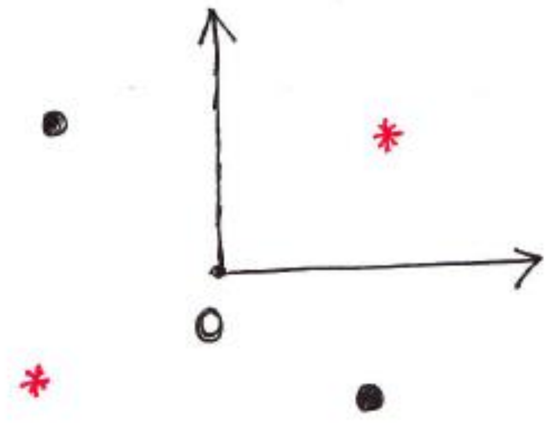
* So $w_5 = w_4 + y_4 x_4$

$= (3, -1) + (-2, -2)$

$= (1, -3)$

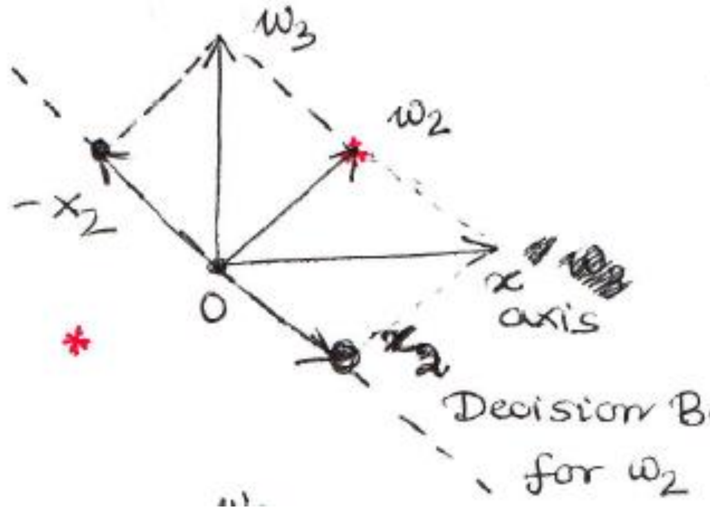
Running Example #2

$((1, 1), 1), \quad ((1, -1), -1), \quad ((-1, 1), -1), \quad ((-1, -1), 1)$



Initially, $w_1 = 0$. In round 1,

* $y_1 \langle w_1, x_1 \rangle \leq 0$, so mistake.
* $w_2 = w_1 + y_1 x_1 = (1, 1)$

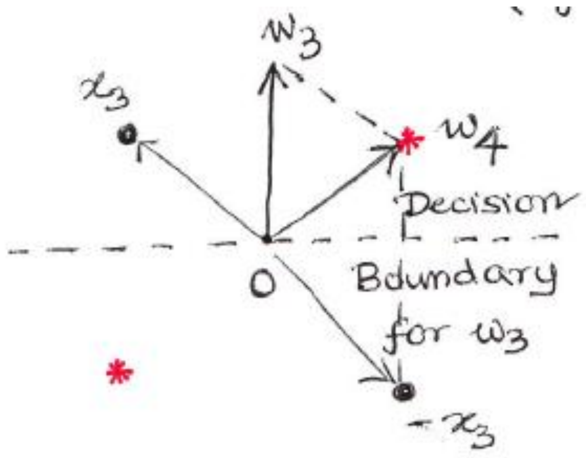


Round 2:

* $y_2 \langle w_2, x_2 \rangle \leq 0$
* $w_3 = w_2 + y_2 x_2 = (1, 1) - (1, -1)$

$= (0, 2)$

Running Example #2



Round 3:

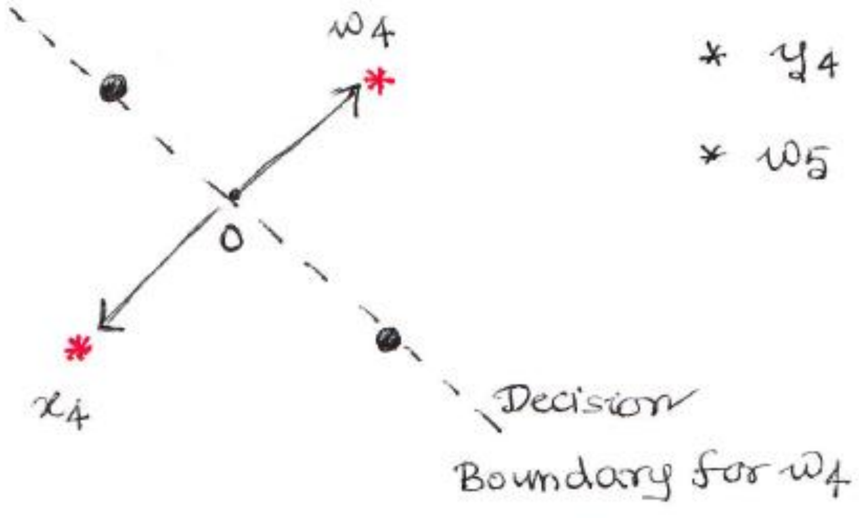
* $y_3 \langle w_3, x_3 \rangle \leq 0$
* $w_4 = w_3 + y_3 x_3 = (0,2) - (-1,1)$

  $= (1, 1)$

Round 4:

* $y_4 \langle w_4, x_4 \rangle \leq 0.$
* $w_5 = w_4 + y_4 x_4 = (1,1) + (-1,-1)$

  $= 0$



The Perceptron Algorithm

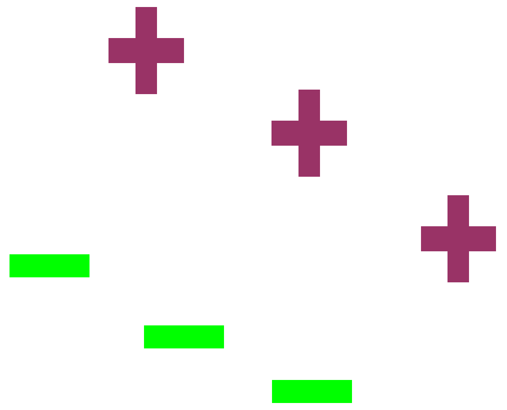
- **Suppose w_t makes a mistake on (x_t, y_t), and we update w_{t+1}as $w_{t+1} = w_t + y_t x_t$. Is it possible for w_{t+1} to also make a mistake on (x_t, y_t) ?**

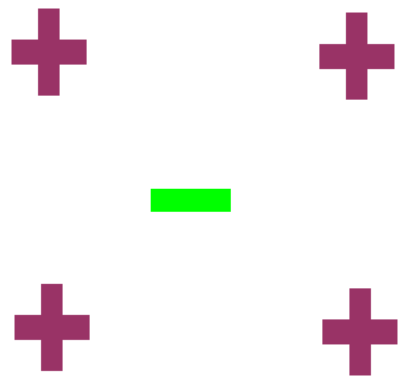
The Perceptron Algorithm ($\eta = 1$)

- **Suppose w_t makes a mistake on (x_t, y_t), and we update w_{t+1} as $w_{t+1} = w_t + y_t x_t$. Is it possible for w_{t+1} to also make a mistake on (x_t, y_t) ?**
  -
  - Yes, depends on the learning rate η

When does Perceptron converge?

- **Linear Separability**
  - There exists a hyper-plane (weight vector) separating the positive and negative points


Linearly separable


Not linearly separable

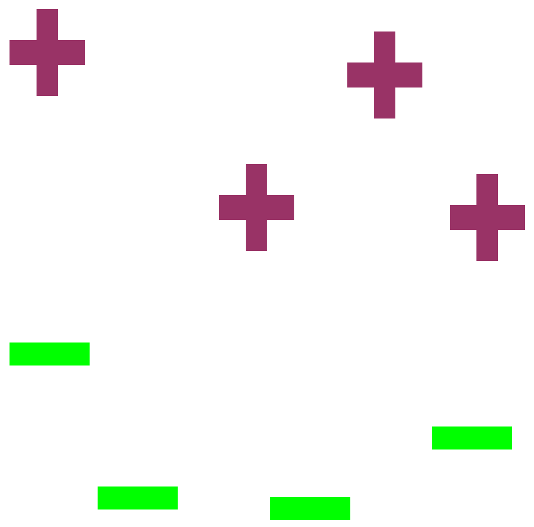
Measure of Separability

- **Margin**
  - For a weight vector $\boldsymbol{w}$, and training set $\boldsymbol{S}$, margin of w with respect to S is defined as follows:

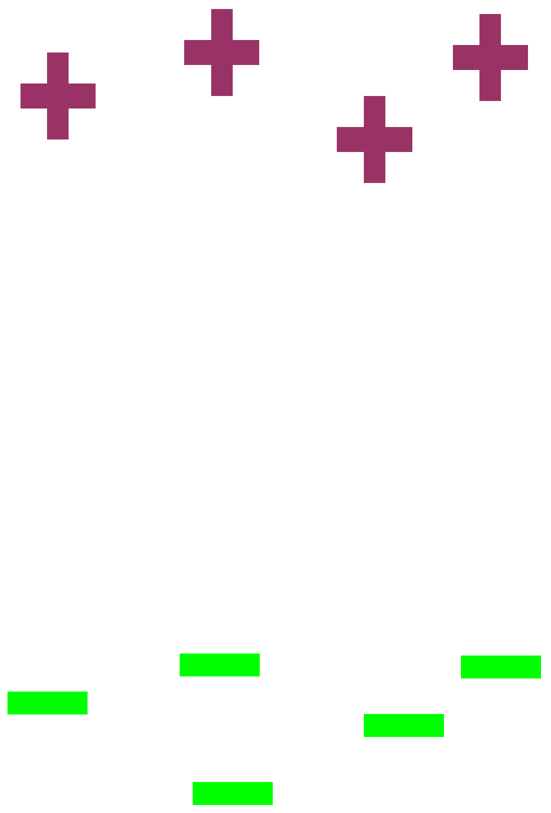
$$\boldsymbol{\gamma(w) = min_{(x,y) \in S}} \quad y(w.x)$$

- The training data $\boldsymbol{S}$ is linearly separable if there exists *at least* one weight vector w for which the margin is positive, i.e., $\boldsymbol{\gamma(w) > 0}$.

Margin: Examples



Low margin data



High margin data

Perceptron: Convergence Result

- **Theorem:** If the training data is linearly separable with margin γ, and if $||x_i|| \leq 1$ for all examples (x_i, y_i) in the training set, then perceptron makes $\leq \frac{1}{\gamma^2}$ mistakes.
  - Proof??

- Lower margin implies more mistakes

- May need more than one pass over the training data to get a classifier with no mistakes

What if data is not linearly separable?

- **Ideally, we want to find a linear separator that makes the minimum number of mistakes on the training data**
  - NP-Hard problem! (Minsky and Papert, 1969)
  - This result killed the neural networks research in 1970's

- **Perceptron still works**
  - there will be few mistakes close to the decision boundary
  - will never converge to a single w as we make more passes

Problems with Perceptron

- Doesn’t converge with inseparable data
- Weight updates may often be very “bold”
- Doesn’t optimize margin
- Sensitive to the order of examples

▲ **Voted and Averaged perceptron**

Voted Perceptron

- **Initialization:** $m = 1;\ w_1 = 0;\ c_m = 1$
- **Training Examples:** for $t = 1, 2, 3, \ldots$
  - If mistake, update weights
    - $w_{m+1} = w_m + y_t x_t$
    - $m = m + 1$
    - $c_m = 1$
  - Else
    - $c_m = c_m + 1$ // counting how long w_m survived
- **Output:** $(w_1, c_1), (w_2, c_2), (w_3, c_3), \ldots$

Voted Perceptron Classifier

$$f(x) = sign\left(\sum_{i=1}^{m} c_i \, sign(< w_i, x >)\right)$$

- Any drawbacks of voted perceptron?

Voted Perceptron Classifier

$$f(x) = sign\left(\sum_{i=1}^{m} c_i \, sign(< w_i, x >)\right)$$

- Any drawbacks of voted perceptron?
- Yes, we have to store all the classifiers (in practice could be many)
- How can we solve this problem?

Averaged Perceptron

- Same algorithm as voted perceptron, but the classification rule is different

$$f_{average}(x) = sign\left(\sum_{i=1}^{m}(< c_i w_i, x >)\right)$$

$$f_{voted}(x) = sign\left(\sum_{i=1}^{m} c_i\, sign(< w_i, x >)\right)$$

Averaged vs. Voted Perceptron

- Simple Example: If $c_1 = c_2 = c_3 = 1$

$$\boldsymbol{f_{average}(x) = sign(\langle w_1 + w_2 + w_3, x\rangle)}$$

$$\boldsymbol{f_{voted}(x) = majority\ sign\ of\ \langle w_1, x\rangle, \langle w_2, x\rangle, \langle w_3, x\rangle}$$

Some Practical Tricks

- **Shuffling**
  - shuffling the training examples in each iteration
- **Variable learning rate**
  - decrease as learning progresses
  - follow some schedule
  - Set automatically by line search (converges faster)
  - See Leon Bottou's SGD website: http://leon.bottou.org/projects/sgd
- **Averaged Perceptron can be implemented very efficiently** (See Algorithm 7 in Hal's chapter)

Perceptron vs. Averaged Perceptron

- Perceptron
  - w_{t+1} = w_t + \eta . y_t . x_t
  - X_t has "n" features, but only "d" features are non-zero => X_t is sparse
  - O(n) => O(d) If d << n, then it is a huge saving
- Averaged Perceptron
  - w = 0, w_{sum} = 0, count =0
  - If we make a mistake, then we do the following:
    - Update w = w + \delta
    - Update w_{sum} = w_{sum} + w => O(n)
    - Update count = count + 1
  - In the end, w_{avg} = w_{sum} / count

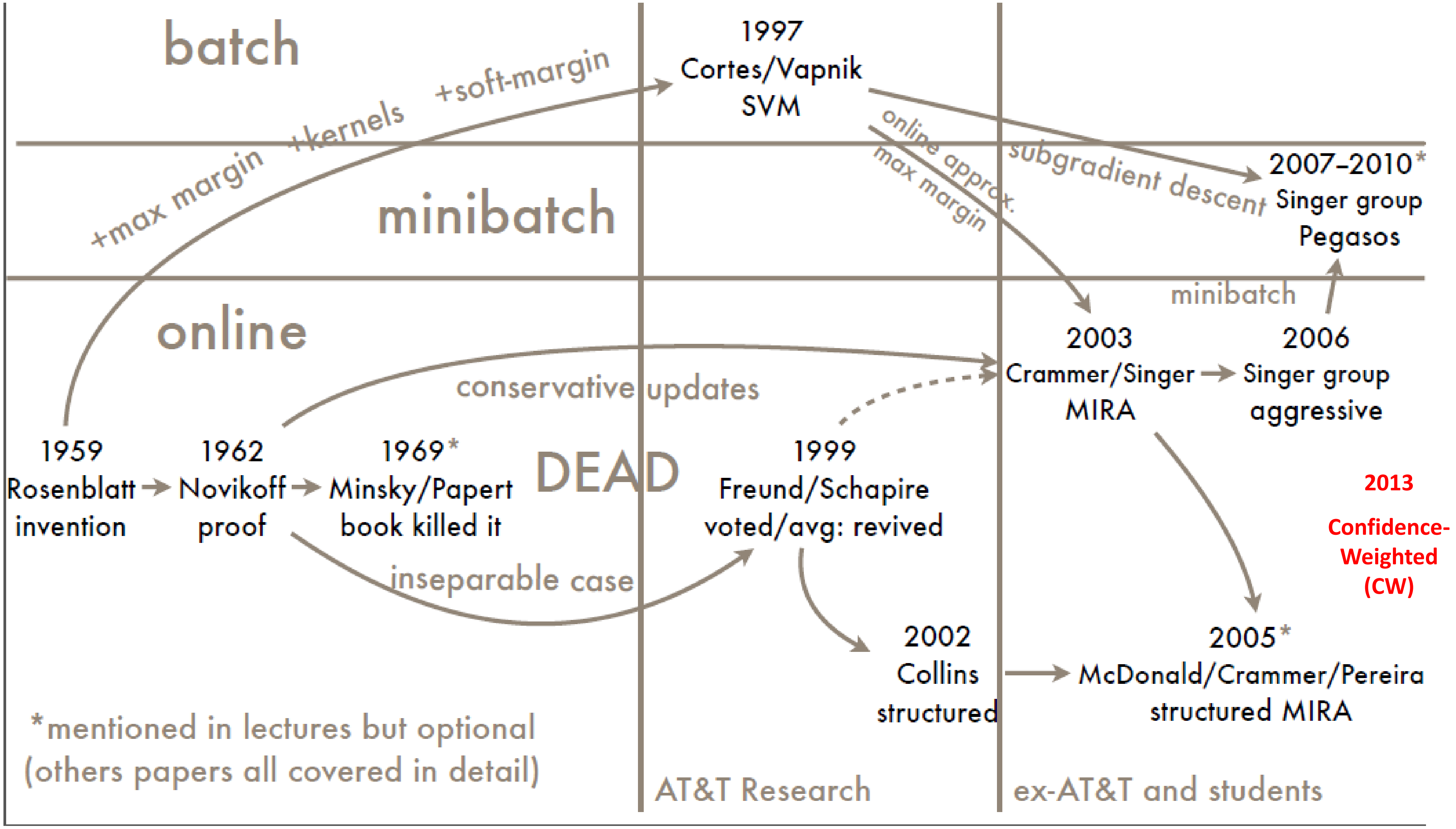
Some Practical Tricks

- **Learning Curve**
  - Training iterations vs. number of mistakes
  - You want to see that the mistakes decrease as we increase the no. of iterations (curve goes down)
  - Very useful in debugging and seeing the behavior of online learning algorithms
- **Hyper-parameter Optimization**
  - Split the training data: sub-train + validation data
  - Tune hyper-parameters (e.g., no. of iterations) on the validation data
  - The learner should not look at the test data!

Two kinds of learning curves

- General learning curve
  - Applicable for any learning algorithm
  - X-axis (number of training examples)
  - Y-axis (accuracy on unseen data)
- Online learning curve
  - Specific to online learning algorithms
  - For a fixed training set
  - X-axis (number of iterations or passes over the data)
  - Y-axis (number of mistakes made)

History of Perceptron*



batch
minibatch
online
+max margin
+kernels
+soft-margin
1997 Cortes/Vapnik SVM
online approx. max margin
subgradient descent
2007–2010* Singer group Pegasos
minibatch
2003 Crammer/Singer MIRA
2006 Singer group aggressive
conservative updates
1959 Rosenblatt invention
1962 Novikoff proof
1969* Minsky/Papert book killed it
DEAD
1999 Freund/Schapire voted/avg: revived
inseparable case
2002 Collins structured
2005* McDonald/Crammer/Pereira structured MIRA
2013 Confidence-Weighted (CW)
*mentioned in lectures but optional (others papers all covered in detail)
AT&T Research
ex-AT&T and students

* slide from Liang Huang

Passive-Aggressive (PA) Algorithm

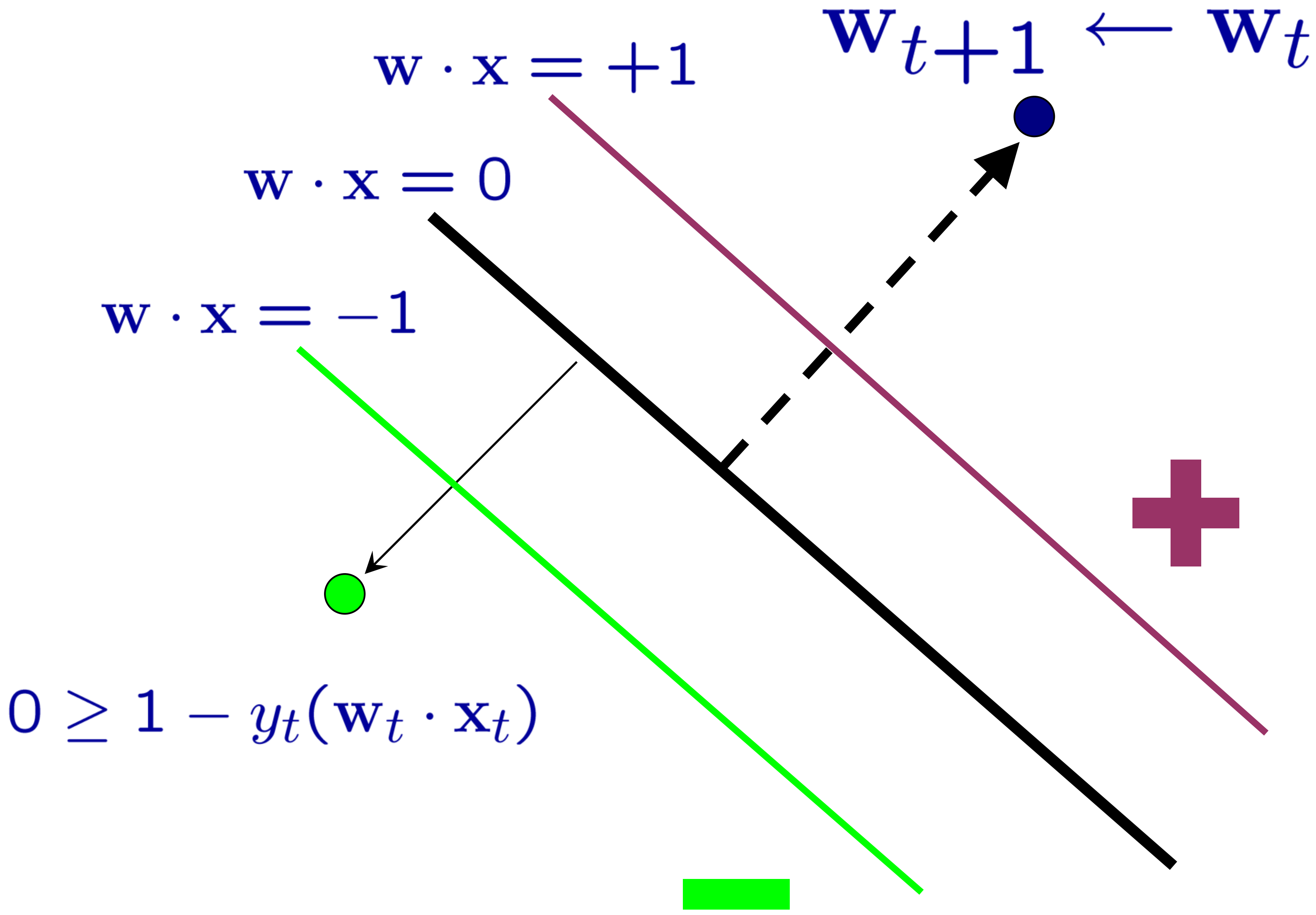
PA Algorithm: Motivation

- **Perceptron:** No guaranties of margin *after* the update
- **PA:** Enforce a minimal non-zero margin after the update
- In particular :
  - If the margin is large enough (1), then do nothing
  - If the margin is less then unit, update such that the margin *after* the update is *enforced* to be unit

Margin

- Y.(w_{t+1}.x) > 0 then the prediction is correct
  - +1 (> 0) > 0 => Sign (w_{t+1}.x) = +1
  - -1 (< 0) > 0 => Sign (w_{t+1}.x) = -1
- If I’m making a mistake, then this margin is < 0
- If margin > 0 implies corrective behavior
  - Margin > 1 also implies corrective behavior

Input Space



$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
$\mathbf{w}\cdot\mathbf{x} = +1$
$\mathbf{w}\cdot\mathbf{x} = 0$
$\mathbf{w}\cdot\mathbf{x} = -1$
$0 \geq 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$

Input Space vs. Version Space

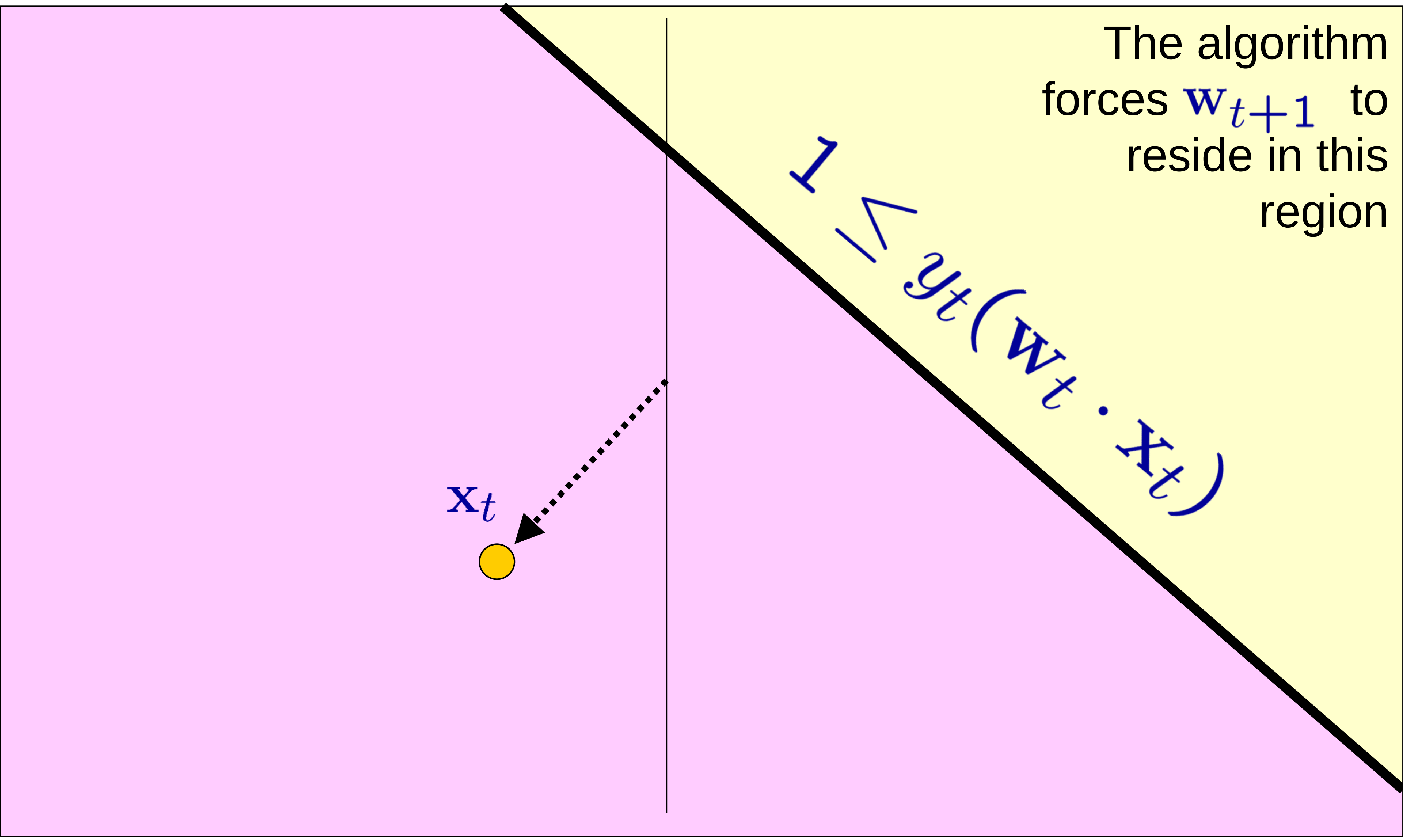
- **Input Space :**
  - Points are input data $y_t \mathbf{x}_t$
  - One constraint is induced by weight vector $\mathbf{w}$
  - Primal space
  - Half space = all input examples that are classified correctly by a given predictor (weight vector)

$$\{y\mathbf{x} \ : \ \mathbf{w} \cdot (y\mathbf{x}) \geq 0\}$$

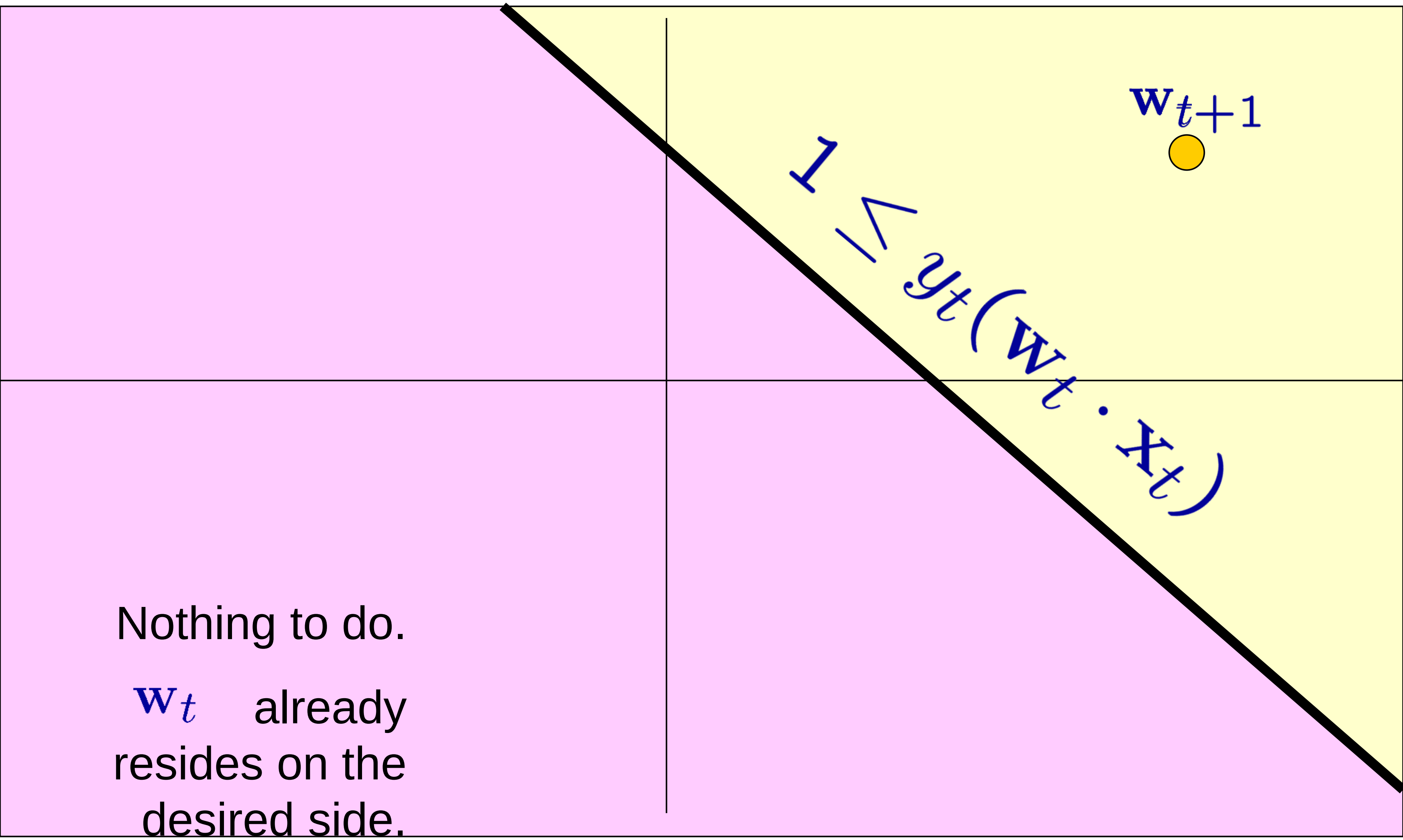
- **Version Space :**
  - Points are weight vectors $\mathbf{w}$
  - One constraints is induced by input data $y_t \mathbf{x}_t$
  - Dual space
  - Half space = all predictors (weight vectors) that classify correctly a given input example

$$\{\mathbf{w} \ : \ \mathbf{w} \cdot (y\mathbf{x}) \geq 0\}$$

Weight vector (Version) Space


The algorithm forces $\mathbf{w}_{t+1}$ to reside in this region
$1 \leq y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$
$\mathbf{x}_t$

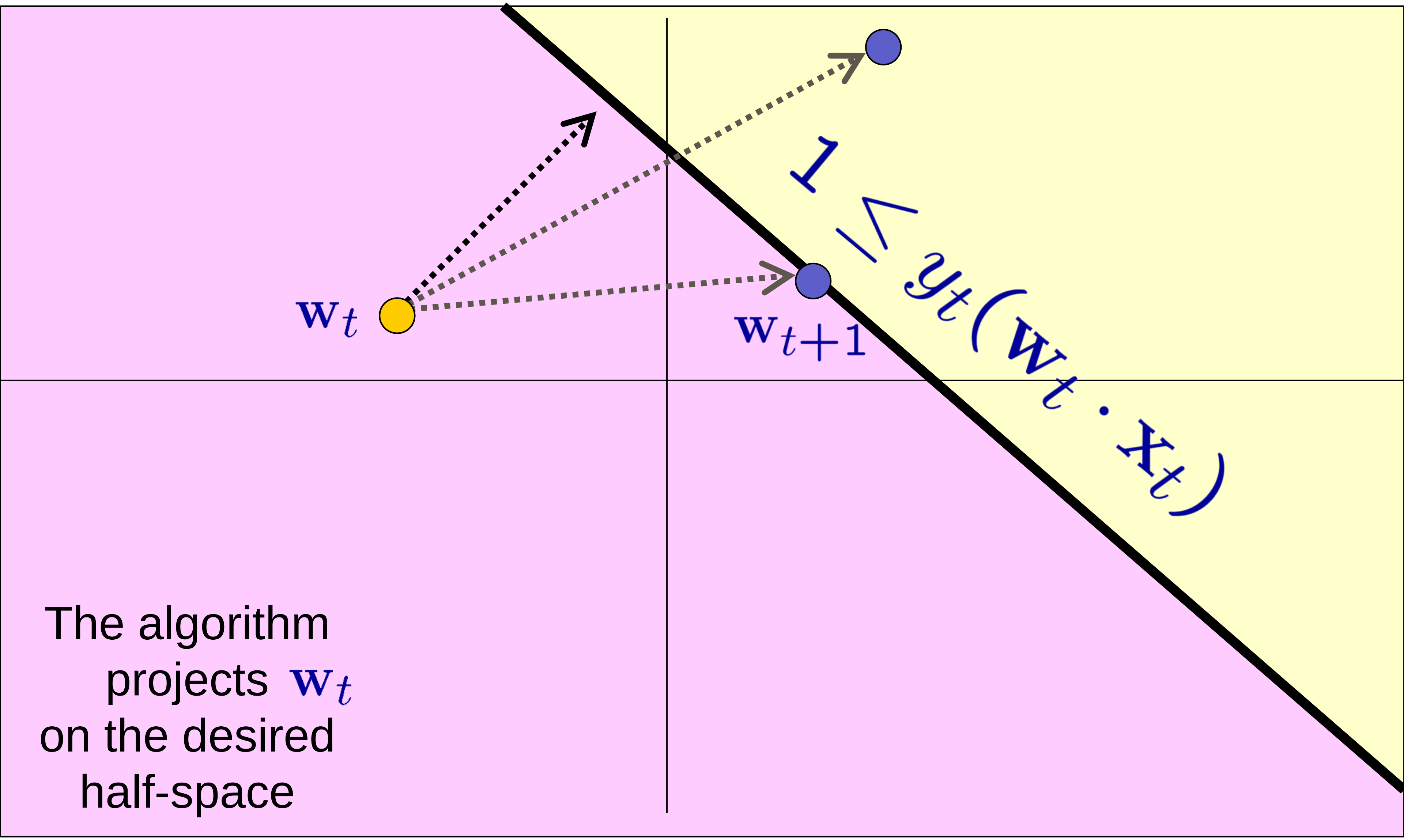
Passive Step



Nothing to do.

$\mathbf{w}_t$ already resides on the desired side.

Aggressive Step



The algorithm projects $\mathbf{w}_t$ on the desired half-space

Aggressive Update Step

- Set $\mathbf{w}_{t+1}$ to be the solution of the following optimization problem :

$$\mathbf{w}_{t+1} = \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{Conservative}$$
$$\text{s.t.} \quad y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 1 \quad \text{Corrective}$$

- The Lagrangian :

$$\mathcal{L}(\mathbf{w}, \tau) \;=\; \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \tau(1 - y_t(\mathbf{w} \cdot \mathbf{x}_t))$$

- Solve for the dual : $\max_{\tau \geq 0} \min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}, \tau)$

Norm of a vector

- L_1 norm
- L_2 norm
- L_infinity norm
- W = [w_1, w_2,…,w_d]
- ||W|| = Sqrt (w_1^2 + w_2^2 +…+w_d^2)

Aggressive Update Step

- Optimize for $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t$$

- Set the derivative to zero $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$
- Substitute back into the Lagrangian :

$$\mathcal{L}(\tau) = -\frac{1}{2}\|\mathbf{x}_t\|^2 \tau^2 + \tau(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))$$

- Dual optimization problem

$$\max_{\tau \geq 0} \quad -\frac{1}{2}\|\mathbf{x}_t\|^2 \tau^2 + \tau(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))$$

Aggressive Update Step

- Dual Problem :

$$\max_{\tau \geq 0} \quad -\frac{1}{2}\|\mathbf{x}_t\|^2\tau^2 + \tau(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))$$

- Solve it :

$$\tau \;=\; \max\left\{0, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}\right\}$$

Alternative Derivation

- Additional Constraint (linear update) :

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t + \tau y_t \mathbf{x}_t$$

- Force the constraint to hold as equality

$$1 = y_t((\mathbf{w}_t + \tau y_t \mathbf{x}_t) \cdot \mathbf{x}_t) \;= y_t(\mathbf{w}_t \cdot \mathbf{x}_t) + \tau \|\mathbf{x}_t\|^2$$

- Solve :

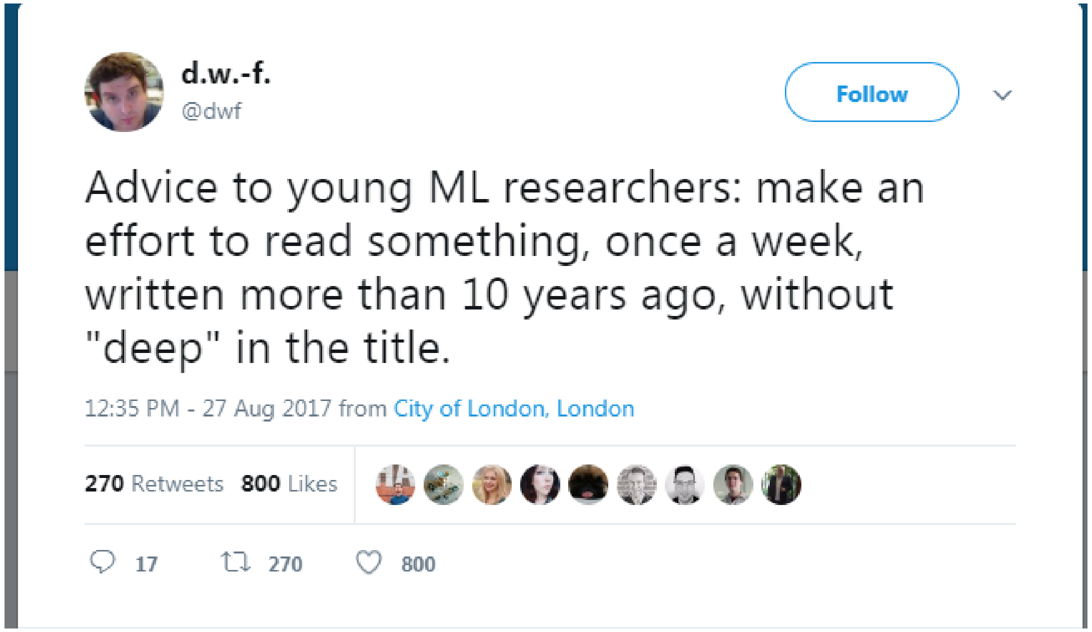
$$\tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}$$

Recap of last Lecture

- Perceptron algorithm
  - Convergence result for linearly separable data
  - Voted and Averaged perceptron
  - Practical tricks: shuffling, variable learning rate, …
- History of Perceptron
  -

Recap of Last Lecture

- **Passive-Aggressive Algorithm**
  - Smallest change to weights to ensure a margin of 1 on the new example
  - Derivation of weight update
  - Perceptron vs. Passive-Aggressive update



d.w.-f.
@dwf

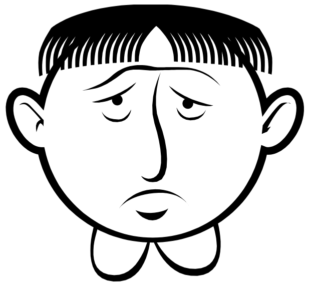
Advice to young ML researchers: make an effort to read something, once a week, written more than 10 years ago, without "deep" in the title.

12:35 PM - 27 Aug 2017 from City of London, London

270 Retweets 800 Likes

17 270 800

Passive-Aggressive Update



$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$$

$$y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \geq 1 \qquad\qquad y_t(\mathbf{w}_t \cdot \mathbf{x}_t) < 1$$

$$\tau = 0 \qquad\qquad \tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{||\mathbf{x}_t||^2}$$

Perceptron vs. PA Update

- Common Update :

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$$

- Perceptron

$$\tau = \begin{cases} 1 & y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0 \\ 0 & y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0 \end{cases}$$

- Passive-Aggressive

$$\tau = \max\left\{0, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}\right\}$$

Two kinds of updates

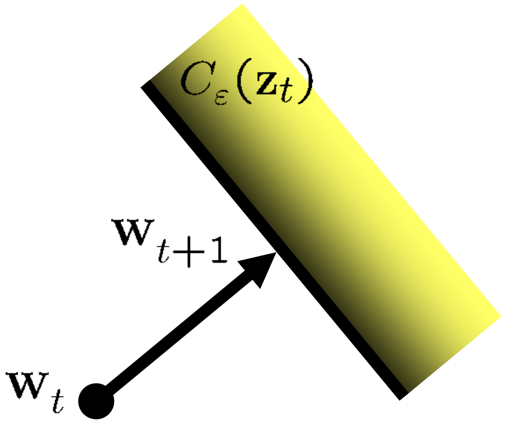
- Additive update
  - w_{t+1} = w_t + \delta (adding some gradient)
- Multiplicative update
  - W_{t+1} = w_t \times \delta
  - Example: Exponentiated Gradient Algorithm
- Yoram Singer => Hebrew Univ => Google (2007)
  - Spam classifiers in Gmail => they are passive-aggressive algorithm

The Passive-Aggressive Algorithm

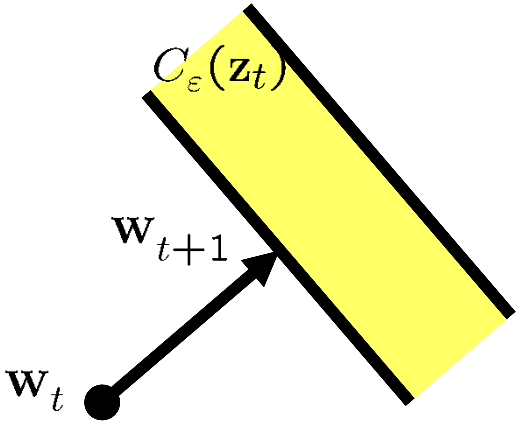
- Each example defines a set of consistent hypotheses: $C_\varepsilon(\mathbf{z}_t) = \{\mathbf{w} \mid \delta(\mathbf{w}; \mathbf{z}_t) \leq \varepsilon\}$
- The new vector $\mathbf{w}_{t+1}$ is set to be the projection of $\mathbf{w}_t$ onto $C_\varepsilon(\mathbf{z}_t)$

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\text{argmin}} \|\mathbf{w} - \mathbf{w}_t\| \text{ s.t. } \mathbf{w} \in C_\varepsilon(\mathbf{z}_t)$$

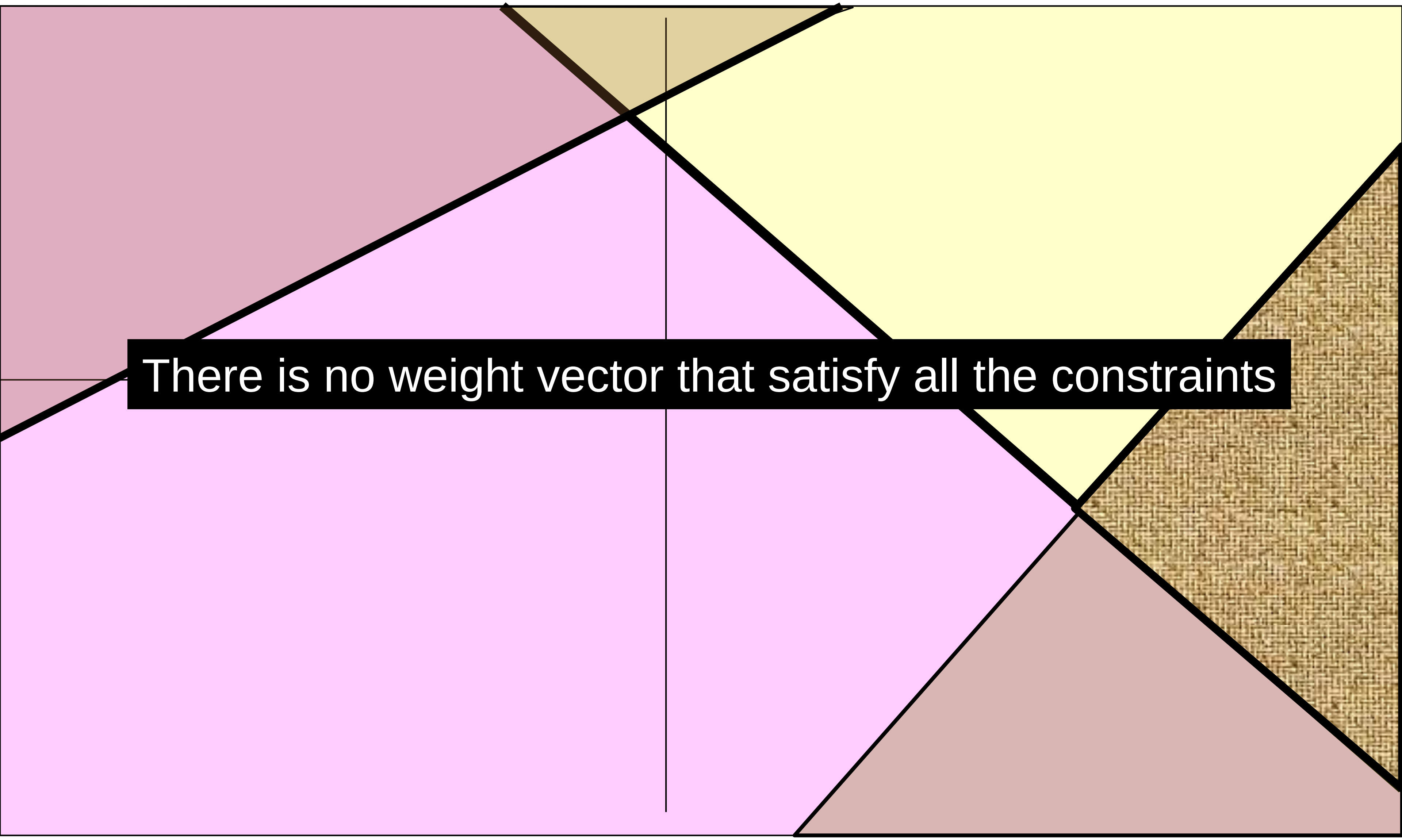
Classification



Regression



Unrealizable Case



There is no weight vector that satisfy all the constraints

Unrealizable Case

$$\mathbf{w}_{t+1} = \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2$$

$$\text{s.t.} \quad y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 1 - \xi \qquad \xi \geq 0$$

$$\Downarrow$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$$

ξ $\swarrow$ $\qquad$ $\searrow$ ξ^2

$$\min\left\{C, \max\left\{0, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}\right\}\right\} \qquad \max\left\{0, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2 + C}\right\}$$

Reduction Hammer

- Smaller problem/nail (binary classification) => Simpler solution/hammer (Perceptron or PA)
  - You have only two classes
- Harder problem/nail
  - You have more than two classes
  - How can I develop an appropriate hammer for this nail?
- Create some simpler problems that you know how to solve => Solve them => Aggregate the solutions to come up with a solution for the harder problem

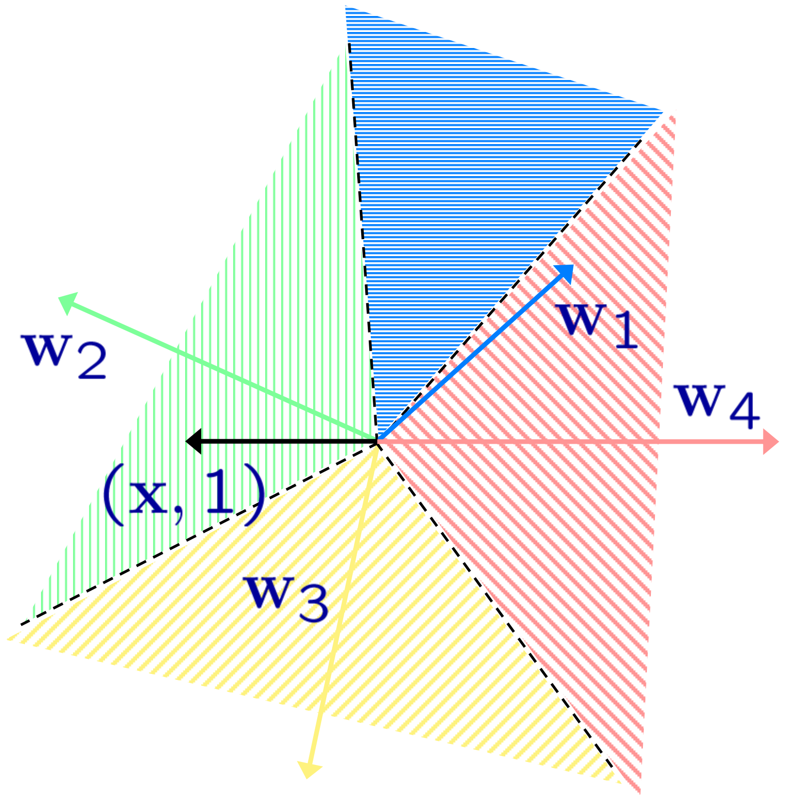
Reduction for Multi-Class Classification

- Classes = {red, blue, green}
- Bunch of training examples {x,y} pairs where y is one of the three classes
- One against others {red vs. other; blue vs. others; green vs. others}
- Increasing the dimension of margin
- Tree of classifiers to filter candidate class labels
- One vs. One: Red vs. Blue; Blue vs. Green; Green vs. Red => Given a new example => Majority vote

Multi-Class: Representation-I

- k Prototypes $\mathbf{w}_1, \mathbf{w}_2 \ldots \mathbf{w}_k$
- New instance $\mathbf{x}$
- Compute $\text{Score}(r) = \mathbf{w}_r \cdot \mathbf{x}$

Class r	$\mathbf{w}_r \cdot \mathbf{x}$
1	-1.08
2	1.66
3	0.37
4	-2.09



- Prediction:

The class achieving the highest Score

Multi-Class Representation-II

- Weight-vector per class (Representation I)
  - Intuitive
- Single weight-vector (Representation II)
  - Generalizes representation I

F(x,4) =

0	0	0	x	0

- Predict label with highest score (Inference)

$$\arg\max_{\mathbf{z}} F(\mathbf{x}, \mathbf{z}) \cdot \mathbf{w}$$

Margin for Multi-Class

- Binary :

$$\mathbf{w}_2 \cdot \mathbf{x} - \mathbf{w}_1 \cdot \mathbf{x} \geq 1$$

$$\mathbf{w} \cdot F(\mathbf{x}, 2) - \mathbf{w} \cdot F(\mathbf{x}, 1) \geq 1$$

- Multi Class :

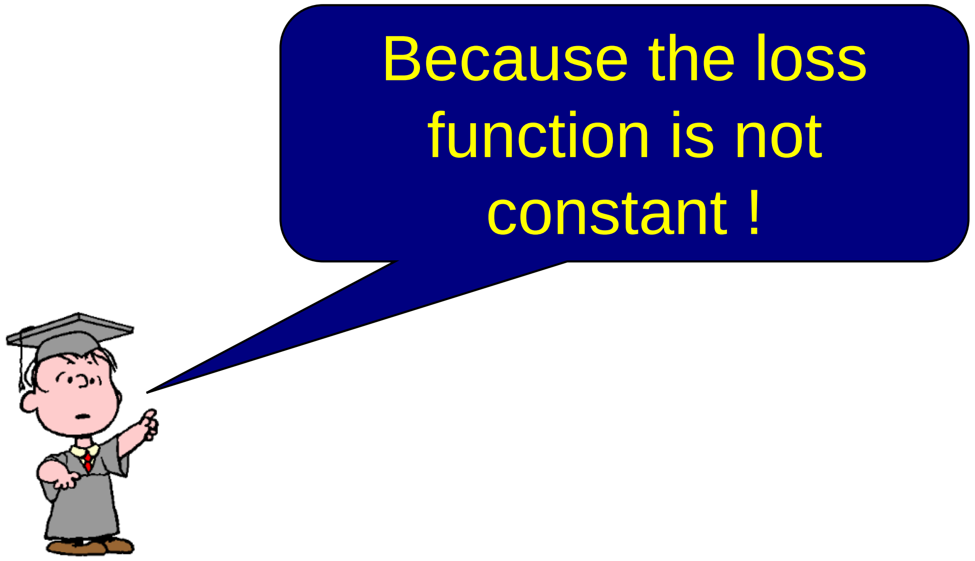
$$\mathbf{w}_y \cdot \mathbf{x} - \mathbf{w}_z \cdot \mathbf{x} \geq 1 \quad \forall z \neq y$$

$$\mathbf{w} \cdot F(\mathbf{x}, y) - \mathbf{w} \cdot F(\mathbf{x}, z) \geq \boxed{1}$$
$$\forall z \neq y$$

Margin for Multi-Class

- Multi Class :

$$\mathbf{w}\cdot F(\mathbf{x},y)-\mathbf{w}\cdot F(\mathbf{x},z)\geq 1$$
$$\forall z\neq y$$



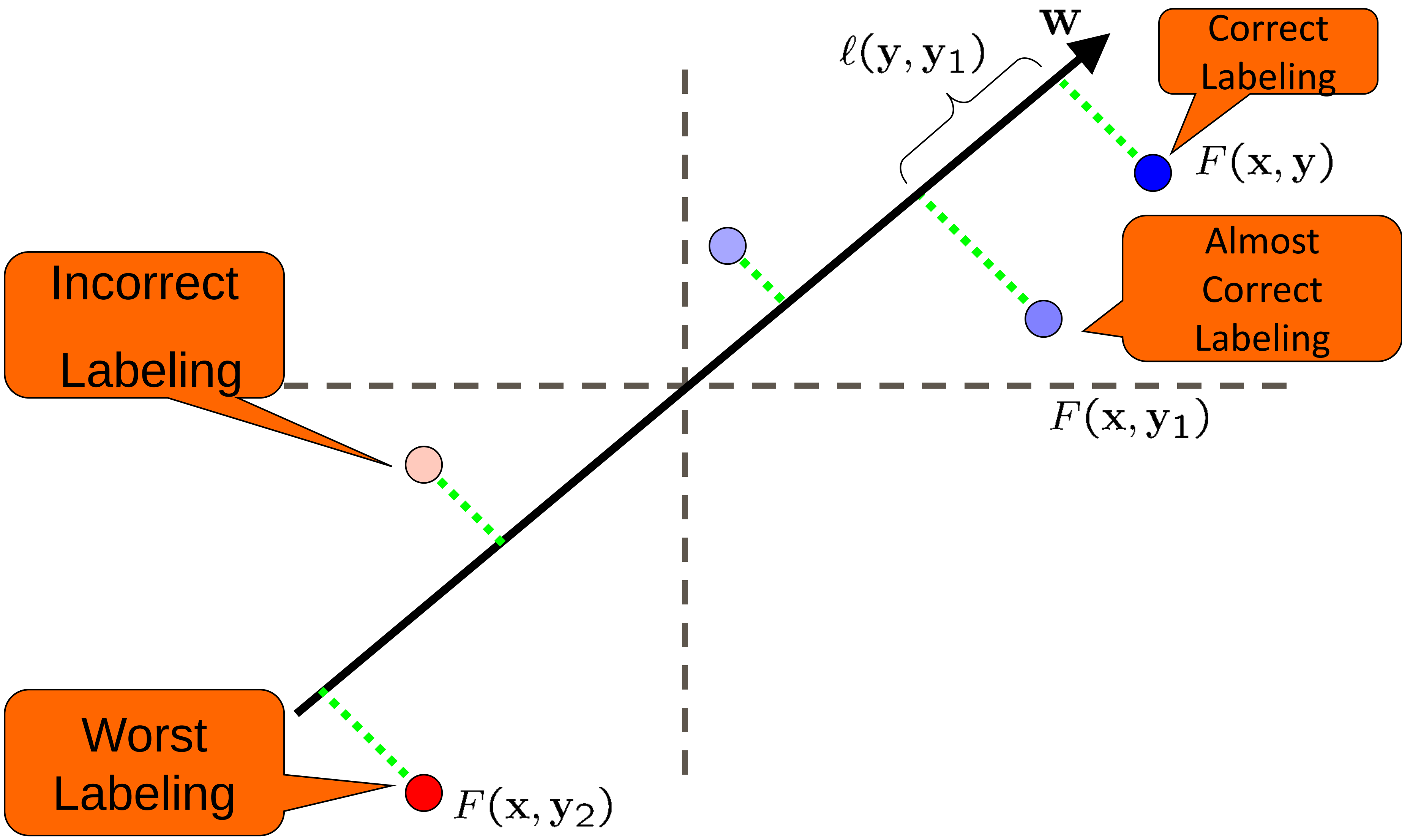
Margin for Multi-Class

- Multi Class :

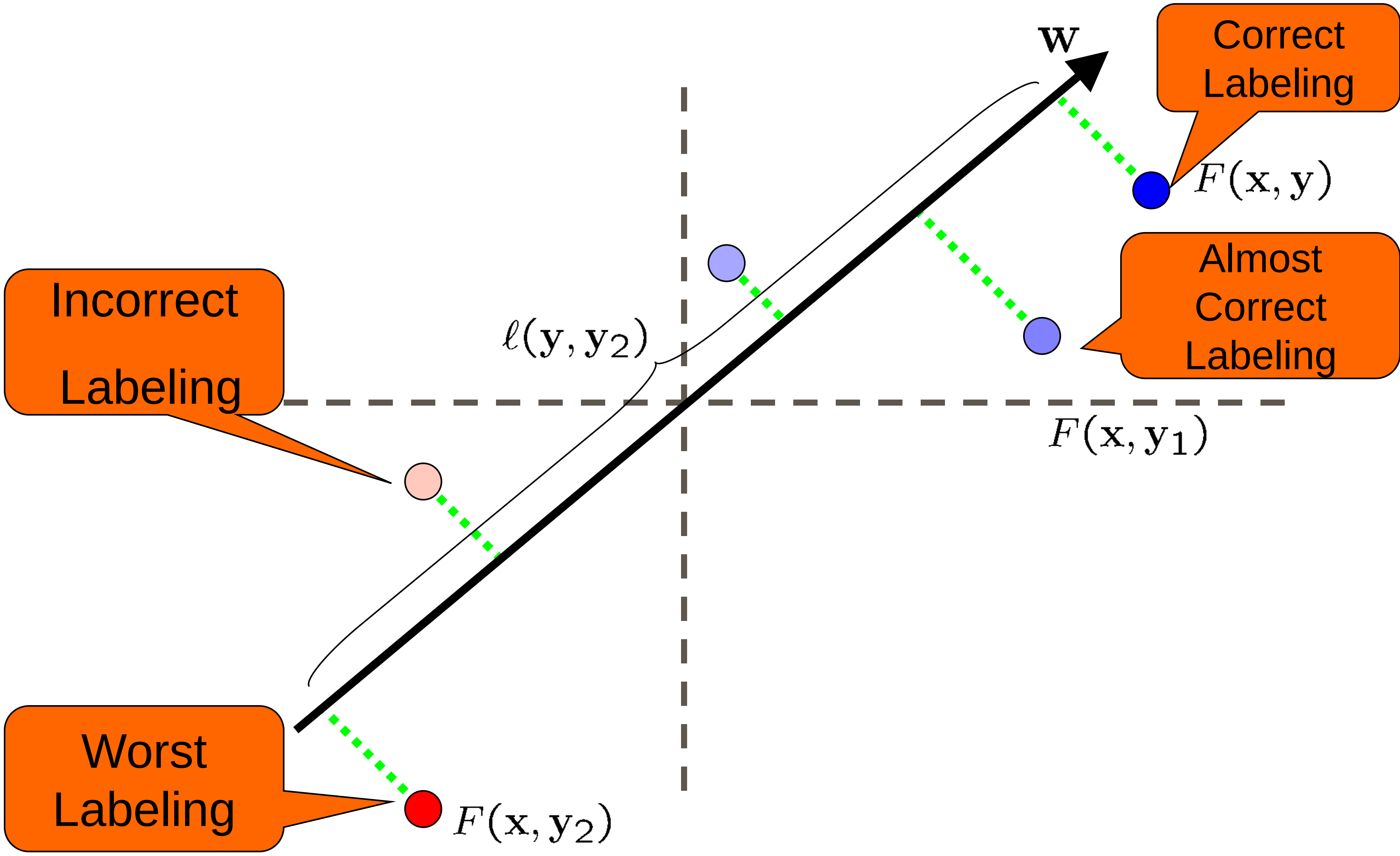
$$\mathbf{w} \cdot F(\mathbf{x}, y) - \mathbf{w} \cdot F(\mathbf{x}, z) \geq \ell(z, y)$$
$$\forall z \neq y$$

So, use it !

Margin Scaled by Loss: Illustration


$\mathbf{w}$
$\ell(\mathbf{y}, \mathbf{y}_1)$
Correct Labeling
$F(\mathbf{x}, \mathbf{y})$
Almost Correct Labeling
Incorrect Labeling
$F(\mathbf{x}, \mathbf{y}_1)$
Worst Labeling
$F(\mathbf{x}, \mathbf{y}_2)$

Margin Scaled by Loss: Illustration


$\mathbf{w}$
Correct Labeling
$F(\mathbf{x}, \mathbf{y})$
Almost Correct Labeling
Incorrect Labeling
$\ell(\mathbf{y}, \mathbf{y}_2)$
$F(\mathbf{x}, \mathbf{y}_1)$
Worst Labeling
$F(\mathbf{x}, \mathbf{y}_2)$

PA Multi-Class Update

- Project the current weight vector such that the instance ranking is consistent with loss function
- Set $\mathbf{w}_{t+1}$ to be the solution of the following optimization problem :

$$\begin{aligned} \mathbf{w}_{t+1} \quad &= \quad \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \\ \text{s.t.} \quad & \quad \mathbf{w}\cdot F(\mathbf{x}_t, y_t) - \mathbf{w}\cdot F(\mathbf{x}_t, z) \geq \ell(z, y_t) \\ & \quad \forall z \neq y_t \end{aligned}$$

PA Multi-Class Update

- **Problem**
  - intersection of constraints may be empty

$$\left\{ \mathbf{w} \;:\; \begin{array}{c} \mathbf{w}\cdot F(\mathbf{x}_t, y_t) - \mathbf{w}\cdot F(\mathbf{x}_t, z) \geq \ell(z, y_t) \\ \forall z \neq y_t \end{array} \right\} = \emptyset$$

- **Solutions**
  - Does not occur in practice
  - Add a slack variable
  - Remove constraints

Add a Slack Variable

- Add a slack variable :

$$\begin{aligned} \mathbf{w}_{t+1} \;&=\; \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{w}\cdot F(\mathbf{x}_t, y_t) - \mathbf{w}\cdot F(\mathbf{x}_t, z) \geq \ell(z, y_t) - \xi \\ & \forall z \neq y_t \\ & \xi \geq 0 \end{aligned}$$

- Rewrite the optimization :

$$\frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C \max \left\{ \begin{array}{c} \mathbf{w}\cdot F(\mathbf{x}_t, z) - \mathbf{w}\cdot F(\mathbf{x}_t, y_t) + \ell(z, y_t) \\ 0 \end{array} \right\}$$

Generalized Hinge loss!!

PA Multi-Class Update

- Remove constraints :

$$\begin{aligned} \mathbf{w}_{t+1} &= \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \\ \text{s.t.} &\quad \mathbf{w}\cdot F(\mathbf{x}_t,y_t)-\mathbf{w}\cdot F(\mathbf{x}_t,z) \geq \ell(z,y_t) \\ &\quad \cancel{\forall z \neq y_t} \end{aligned}$$

$$\begin{aligned} \mathbf{w}_{t+1} &= \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \\ \text{s.t.} &\quad \mathbf{w}\cdot F(\mathbf{x}_t,y_t)-\mathbf{w}\cdot F(\mathbf{x}_t,\hat{y}_t) \geq \ell(\hat{y}_t,y_t) \end{aligned}$$

- How to choose the single competing labeling?
  - The labeling that attains the highest score!

$$\hat{y}_t = \arg\max_z \; \mathbf{w}_t\cdot F(\mathbf{x}_t,z)$$

  - … which is the predicted label according to the current model

PA Multi-Class Online Algorithm

- Initialize $\mathbf{w}_1$
- For $t = 1 \ldots T \ldots$
  - Receive an input instance $\mathbf{x}_t$
  - Outputs a prediction $\hat{y}_t = \arg\max_z \mathbf{w}_t \cdot F(\mathbf{x}_t, z)$
  - Receives a feedback label y_t
  - Computes loss $\ell(\hat{y}_t, y_t)$
  - Update the prediction rule

$$\begin{aligned} \mathbf{w}_{t+1} &= \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \\ \text{s.t.} & \quad \mathbf{w}\cdot F(\mathbf{x}_t, y_t) - \mathbf{w}\cdot F(\mathbf{x}_t, \hat{y}_t) \geq \ell(\hat{y}_t, y_t) \end{aligned}$$

Multi-Class Online Algorithm

- Initialize $\mathbf{w}_1$
- For $t = 1 \ldots T \ldots$
  - Receive an input instance $\mathbf{x}_t$
  - Outputs a prediction $\hat{y}_t = \arg\max_z \mathbf{w}_t \cdot F(\mathbf{x}_t, z)$
  - Receives a feedback label y_t
  - Computes loss $\ell(\hat{y}_t, y_t)$
  - Update the prediction rule

$$w_{t+1} = w_t + \tau \cdot (F(x_t, y_t) - F(x_t, \hat{y}_t))$$

Recap of Last Lecture

- **Multi-Class Representations**
  - Multi-prototype (one weight vector $\boldsymbol{w_i}$ for each class $\boldsymbol{i}$)
  - Single-prototype (one weight vector $\boldsymbol{w}$ – concatenation of all the $\boldsymbol{k}$ weight vector) via extended feature space $\boldsymbol{F(x, y)}$
- **Multi-Class Passive-Aggressive Algorithm**
  - Margin for multi-class classification
  - Margin scaled by the loss function
  - Mathematical optimization problem with $\boldsymbol{k-1}$ constraints

Multi-Class Online Algorithm

- Initialize $\mathbf{w}_1$
- For $t = 1 \ldots T \ldots$
  - Receive an input instance $\mathbf{x}_t$
  - Outputs a prediction $\hat{y}_t = \arg\max_z \mathbf{w}_t \cdot F(\mathbf{x}_t, z)$
  - Receives a feedback label y_t
  - Computes loss $\ell(\hat{y}_t, y_t)$
  - Update the prediction rule

$$w_{t+1} = w_t + \bigcirc\!\!\!\!\!\tau \cdot (F(x_t, y_t) - F(x_t, \hat{y}_t))$$

$$\tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}$$

Binary

$$\tau = \frac{1 - (w_t \cdot F(x_t, y_t) - w_t \cdot F(x_t, \hat{y}_t))}{\|F(x_t, y_t) - F(x_t, \hat{y}_t)\|^2}$$

Multi-class

Multi-Class Classification Implementation

- "d" features for each input example
- W1 (d weights) for class 1
- W2 (d weights for class 2
- ..
- Score(class 1) = W1.X
- W_{correct} = W_{correct} + \eta.X
- W_{wrong} = W_{wrong} - \eta.X

Multi-Class Update

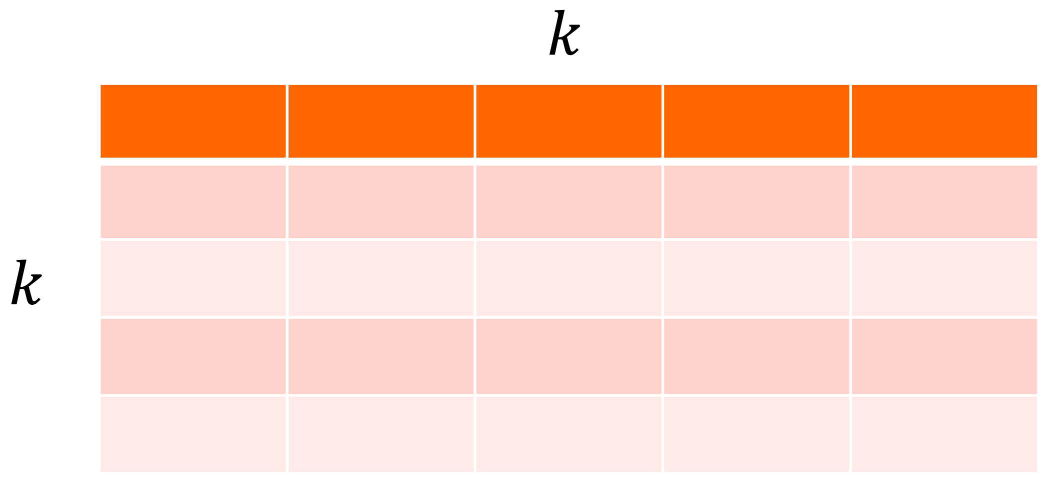
- Score(correct-label) = 3.4
  - W_{correct-label}.X
- Score(wrong-label) = 5.6
  - W_{wrong-label}.X
- Weight update
  - W_{correct-label} = W_{correct-label} + \tau. X
  - W_{wrong-label} = W_{wrong-label} - \tau.X
- Updated score(correct-label)
  - (W_{correct-label} + \tau. X).X = \tau.X^2 (added)
- Updated score(wrong-label)

Binary vs. Multi-Class Interpretation

- Correct-label y_t
- Wrong-label \hat{y_t}
- Feature vectors
  - F(x, y_t)
  - F(x, \hat{y_t})
- F(x, y_t) – F(x, \hat{y_t}) == Input feature vector
- Output label = +1
- F(x, \hat{y_t}) – F(x, y_t) == Input feature vector
- Output label = -1

Confusion Matrix

- Measures which classes are easy or hard to separate -- k classes implies $k \times k$ matrix



- $C_{ij} = \frac{\#\ examples\ with\ class\ label\ j\ that\ are\ classified\ as\ label\ i}{\#\ examples\ with\ label\ j}$
- High diagonal entry implies class is easy to classify
- High off-diagonal entry implies classes are easily confused

Learning Curves: Online vs. General

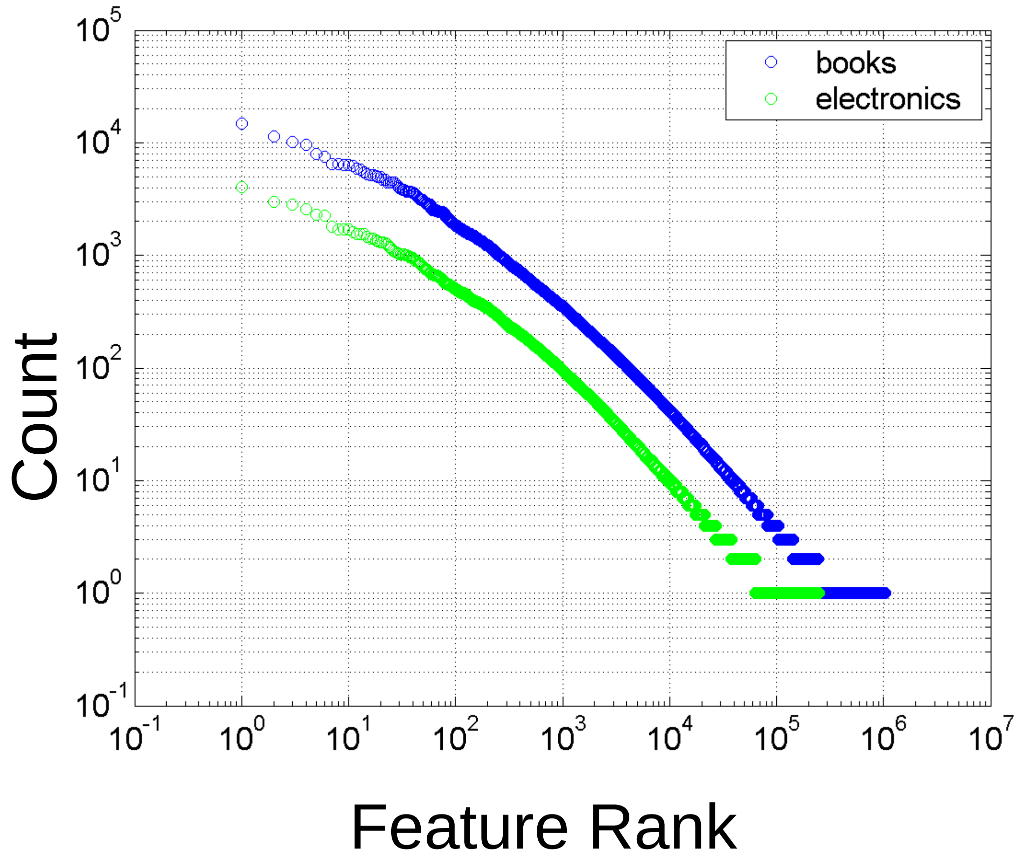
- **Online learning curve**
  - training iterations vs. Mistakes
  - only applicable for online learning algorithms

- **General learning curve**
  - amount of training data vs. accuracy
  - applicable to any learning algorithm

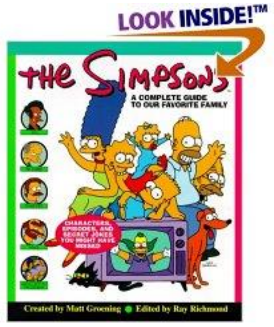
Confidence-Weighted (CW) Algorithm

Motivation: NLP problems

- Big datasets, large number of features
- Many features are only weakly correlated with target label
- Heavy-tailed feature distribution
- Linear classifiers: features are associated with word counts



Motivation: Sentiment Classification



- **Who needs this Simpsons book? You DOOOOOOOO**
  This is one of the most extraordinary volumes I've ever encountered encapsulating a television series … . Exhaustive, informative, and ridiculously entertaining, it is the best accompaniment to the best television show … . Even if you only "enjoy" the Simpsons (as opposed to being a raving fanatic, which most people who watch the show are, after all … Very highly recommended!

Motivation: Sentiment Classification

- Many positive reviews with the word best

$$W_{\text{best}}$$

- Later negative review
  - *"boring book – best if you want to sleep in seconds"*

- Linear update will reduce both

$$W_{\text{best}} \; W_{\text{boring}}$$

- But best appeared more than boring
- How to adjust weights at different rates? W_{boring} W_{best}

Span based Update Rules

- The weight vector is a linear combination of examples

$$\underbrace{w_f}_{\text{Weight of feature f}} \leftarrow \underbrace{w_f}_{\text{Weight of feature f}} + \underbrace{\eta}_{\text{Learning rate}} \underbrace{y}_{\text{Target label, -1 or 1}} \underbrace{x_f}_{\text{Value of feature f of instance}} \quad \forall f$$

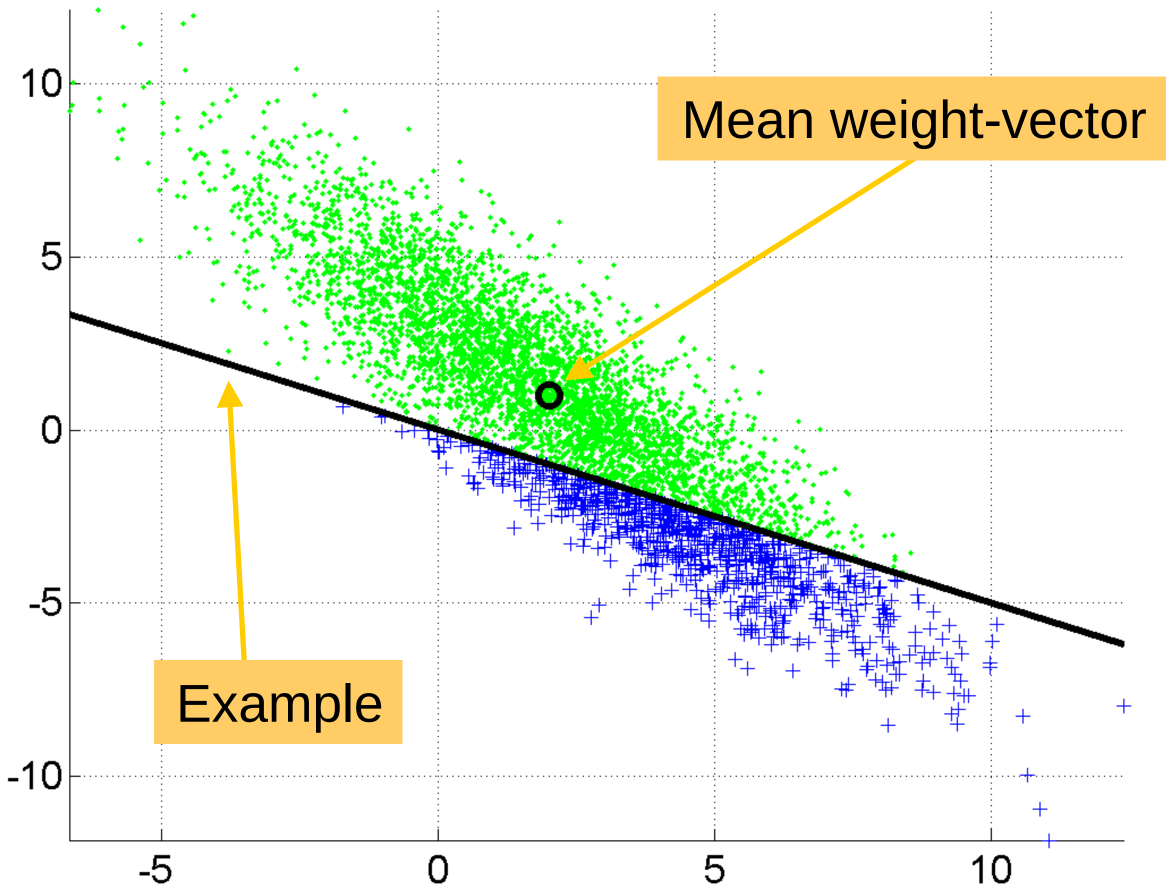
- Two rate schedules (among others):
  - Perceptron algorithm, conservative:

$$\eta = 1$$

  - Passive-aggressive

$$\eta = \max\left\{0, \frac{1 - y(\mathbf{w} \cdot \mathbf{x})}{||\mathbf{x}||^2}\right\}$$

Distributions in Version Space



Margin as a Random Variable

- Signed margin

$$M = y(\boldsymbol{w} \cdot \boldsymbol{x})$$

is a Gaussian-distributed variable

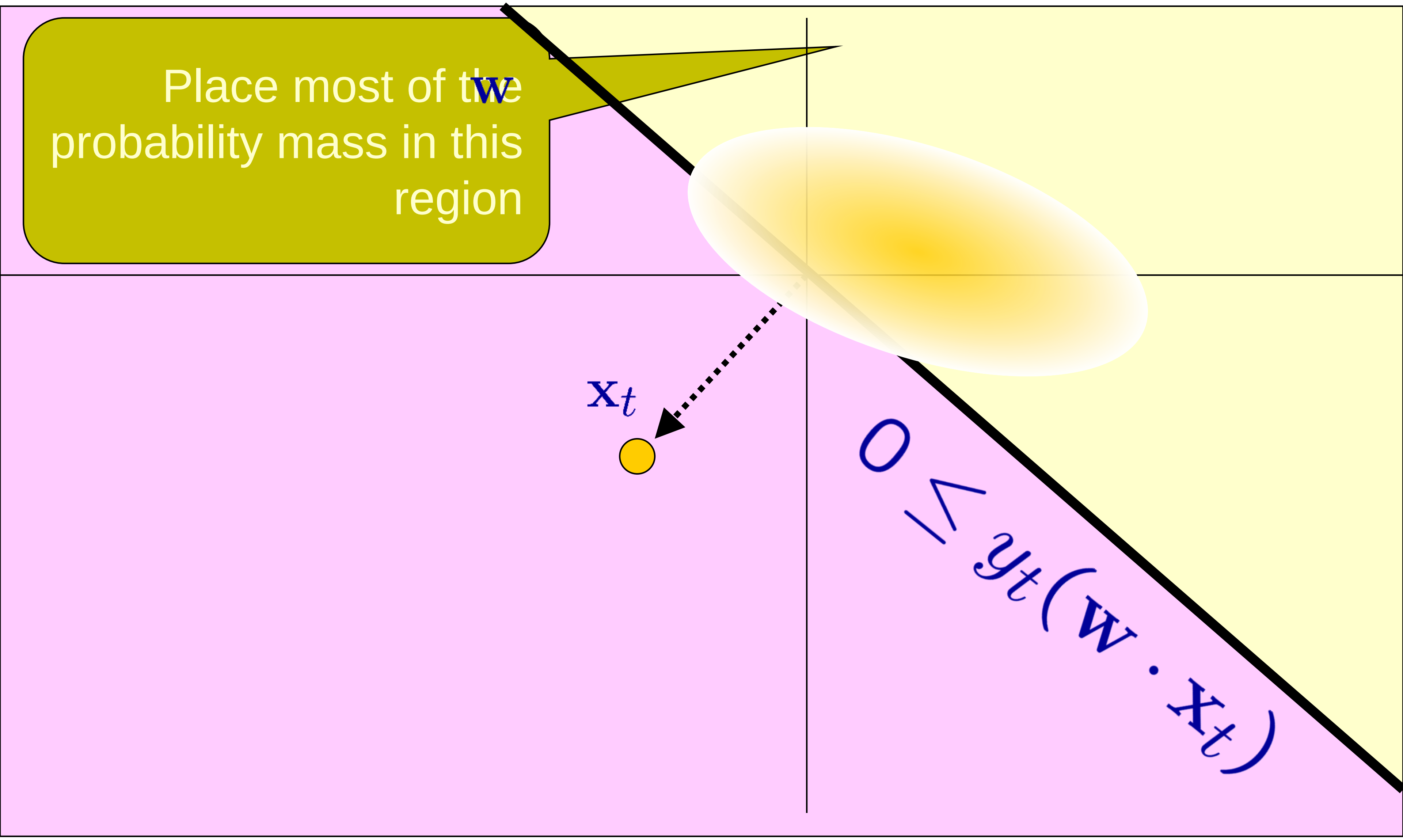
$$M \sim \mathcal{N}\left(y(\boldsymbol{x} \cdot \boldsymbol{\mu})\ ,\ \boldsymbol{x}^\top \Sigma \boldsymbol{x}\right)$$

- Thus:

$$\Pr\left[y\left(\boldsymbol{w} \cdot \boldsymbol{x}\right) \geq 0\right] = \Phi\left(\frac{y(\boldsymbol{x}\cdot\boldsymbol{\mu})}{\sqrt{\boldsymbol{x}^\top \Sigma \boldsymbol{x}}}\right)$$

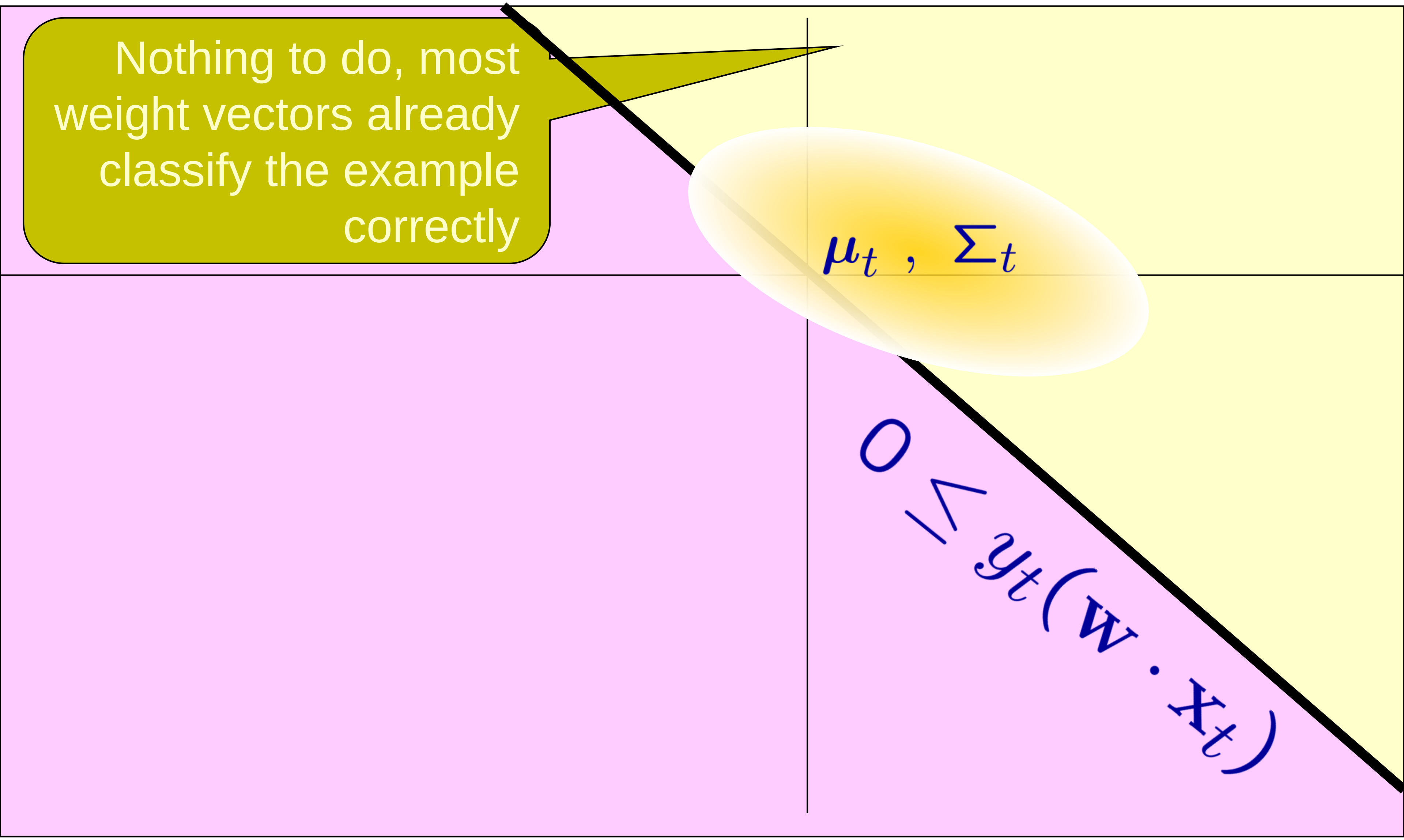
$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2} dt$$

Version (weight vector) Space

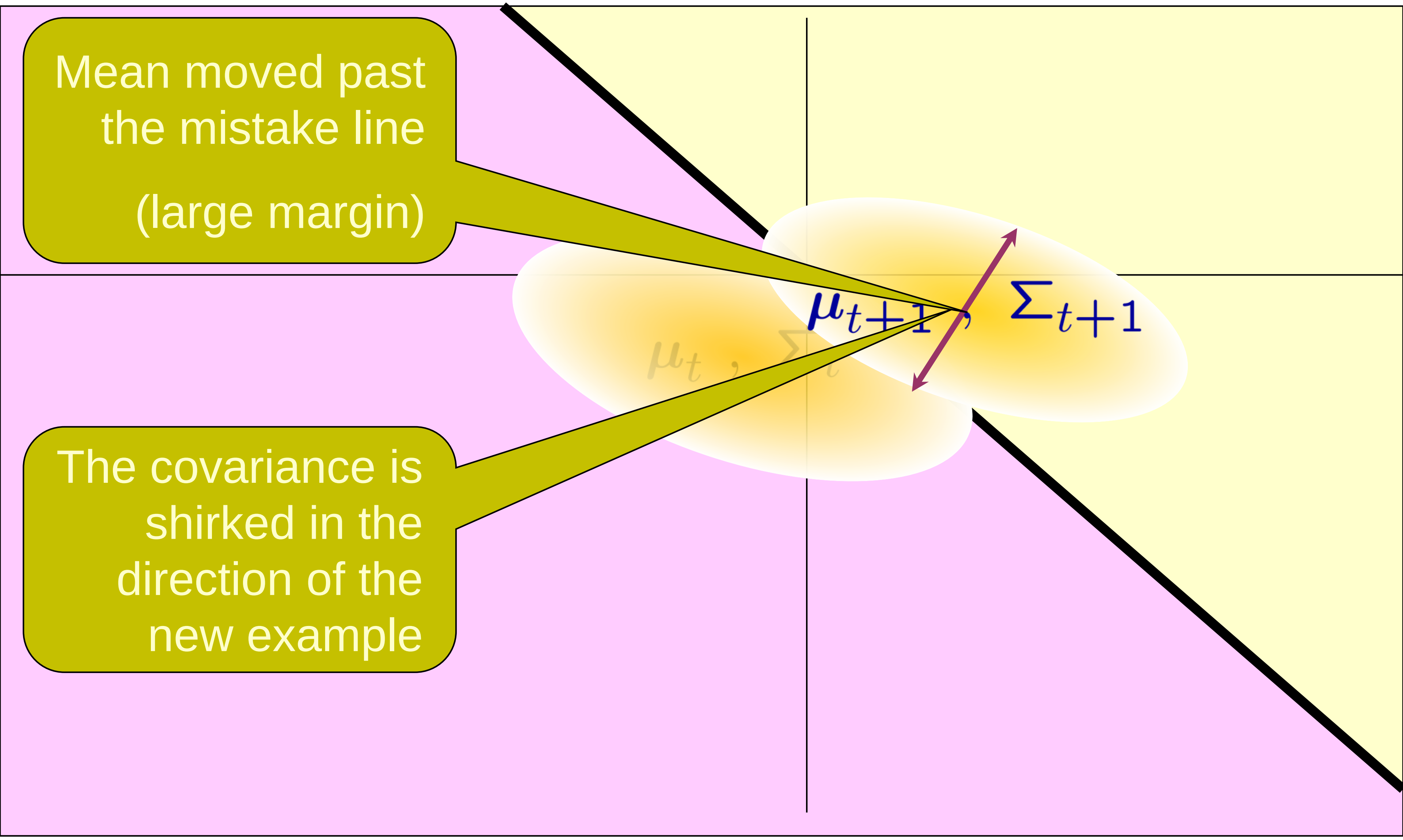


Place most of the probability mass in this region
$\mathbf{w}$
$\mathbf{x}_t$
$0 \leq y_t(\mathbf{w} \cdot \mathbf{x}_t)$

Passive Step


Nothing to do, most weight vectors already classify the example correctly
$\boldsymbol{\mu}_t \;,\; \Sigma_t$
$0 \leq y_t(\mathbf{w} \cdot \mathbf{x}_t)$

Aggressive Step



Mean moved past the mistake line

(large margin)

The covariance is shirked in the direction of the new example

PA like Update

- PA:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_i\|^2$$
$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i) \geq 1$$

- New Update :

Confidence Parameter

$\eta \in (0.5, 1)$

$$\min_{\boldsymbol{\mu},\Sigma} \quad \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu},\Sigma\right) \| \mathcal{N}\left(\boldsymbol{\mu}_i,\Sigma_i\right)\right)$$
$$\text{s.t.} \quad \Pr\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \geq \eta$$

Summary of Online Learning

- **Online learning**
  - Iterative game between teacher and learner
- **Design principles of online learning**
  - Trade-off amount of change (conservative) and reduction in loss (corrective)
- **Online learning algorithms**
  - Perceptron (fixed learning rate for all examples)
  - Passive-Aggressive (fixed learning rate for each example)
  - Confidence-weighted classifier (fixed learning for each feature and each example)

Questions?