Advanced Linear Models

J. Savoy
Université de Neuchâtel

Ian H. Witten, Eibe Frank: *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

Christopher M. Bishop: *Pattern Recognition and Machine Learning*. Springer, 2006

T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning*. Springer, New York, 2009.

Overview

- **Simple Winnow**
- Variants of Winnow

Winnow

A relatively simple solution when facing with numerous binary attributes. We denote by *m* the number of features / attributes.

The decision is binary (two classes).

We want to determine a linear boundary
(e.g., line, plan, hyperplan) between the two classes.

This learning scheme is related to the Perceptron (linear boundary, but additive learning (here multiplicative learning)).

N. Littlestone: Learning Quickly When Irrelevant Attributes Around: A New Linear-threshold Algorithms, *Machine Learning*, 1988, 285-318.

Winnow

We represent each example by a binary vector of size m.

We denote by a_{ji} the binary value of the i*th* attribute (feature) for the j*th* instance (or A_j).

$$A_j = [a_{j1}, a_{j2}, \ldots , a_{jm}] = \{0,1\}^m$$

Each instance can be viewed as a point in a *m*-dimensional cube.

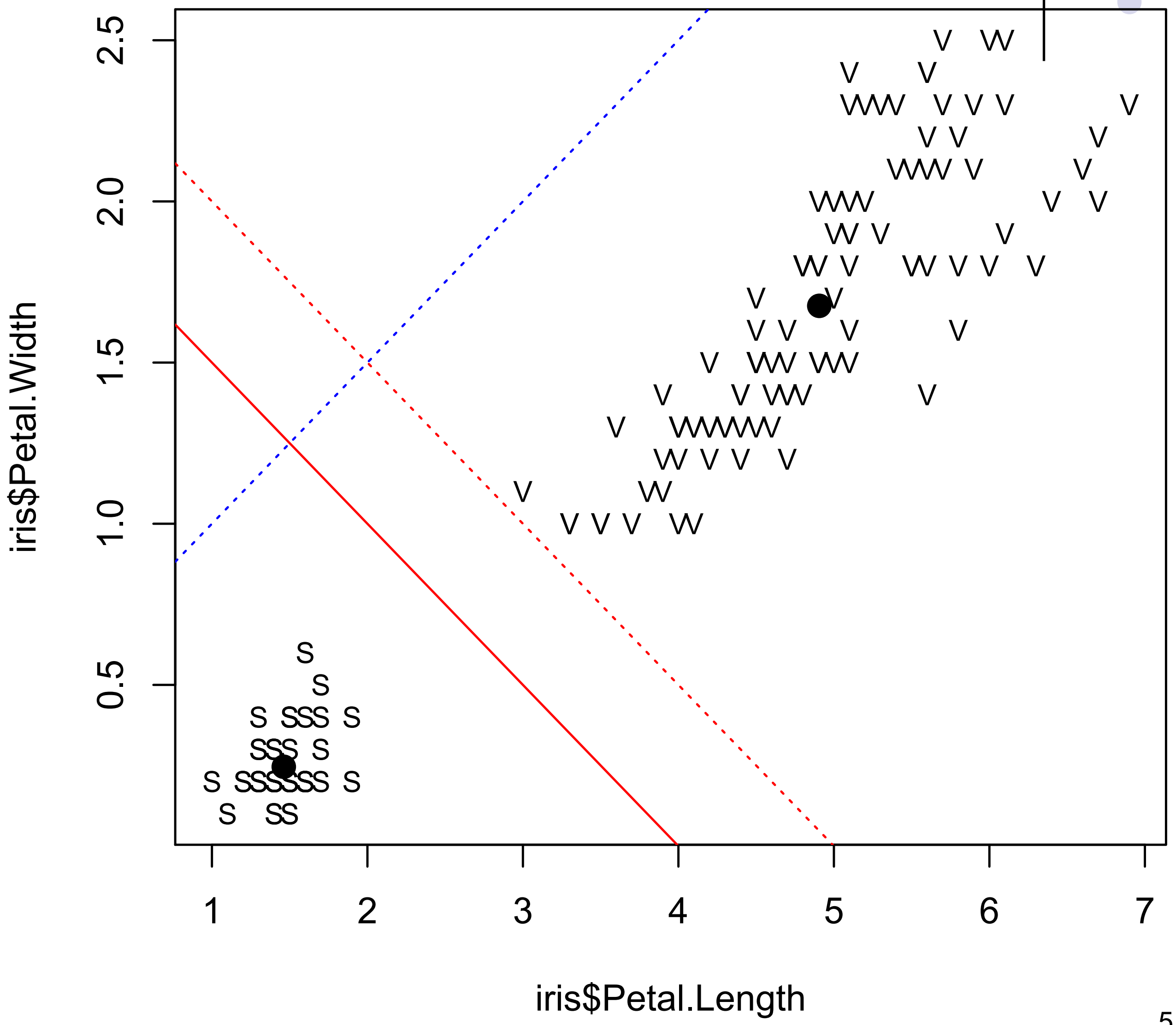
We must split this *m*-space into two distinct regions (classes) using a linear boundary.

Example

With two species of iris, separation according to the length and width of the petals.

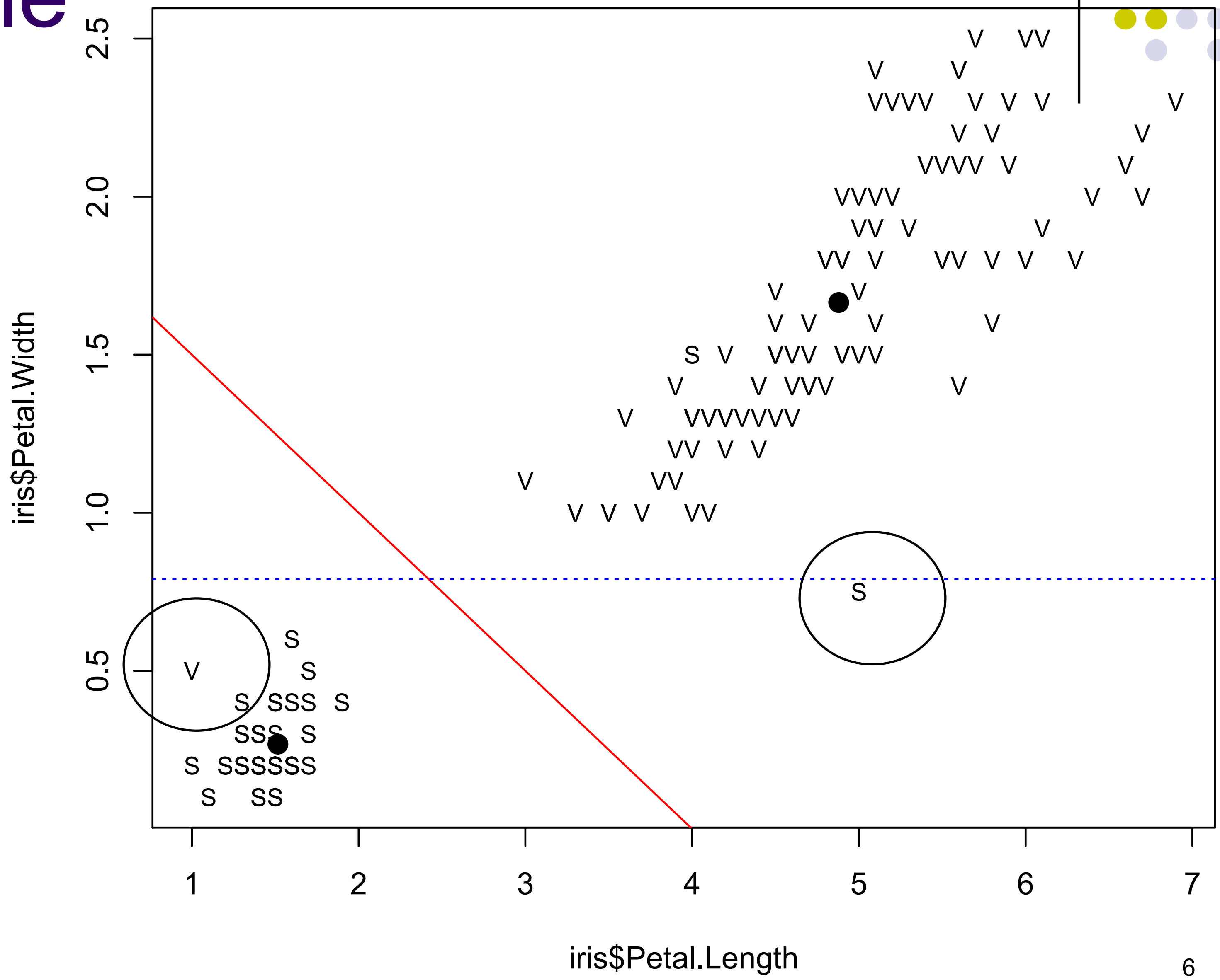
Is it possible?

How to find the linear boundary?



Example

In some case, it is impossible…



Winnow

The idea is to define a hyperplan (or simply a line in 2-D) that can separate the two classes.

This border is presented by a vector of weights (one non-negative weight per attribute) to be learned

$W = [w_1, w_2, \dots , w_m]$

For each instance A_j, we can compute the "distance" to the border and compare it to a predefined threshold (θ)

Decision:

Classify in Class 1 if

$$d_j = \sum_{i=1}^{m} w_i \cdot a_{ji} > \theta$$

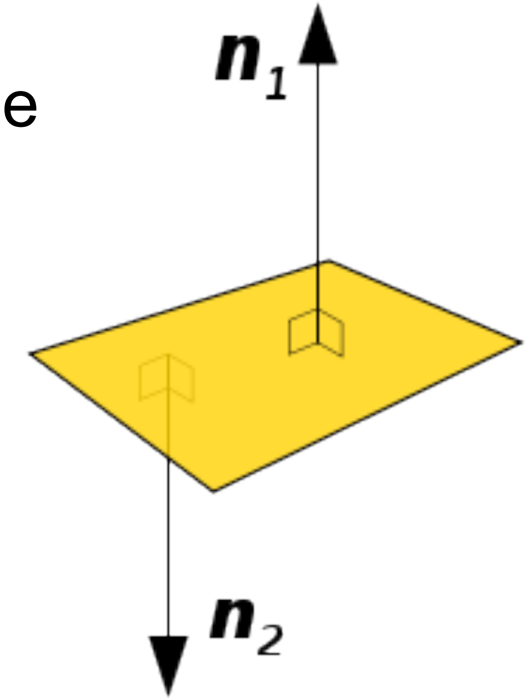
Geometrical View

The vector W of weights W = [w_1, w_2, … , w_m] is the normal vector of the separating hyperplan

This vector indicates the perpendicular direction to the (tangent) plan (at a given point).

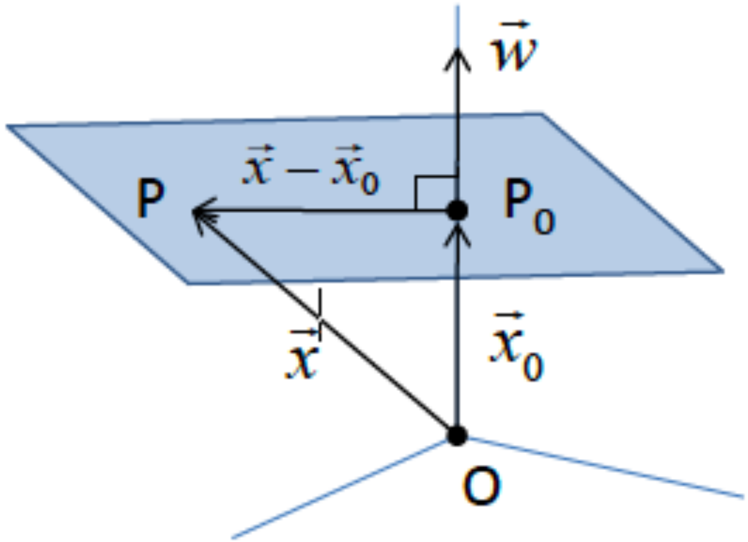
Usually, a plan is defined by a point and the normal vector.

Why useful?



Geometrical View

In R^3



$$\overrightarrow{x_0} = \overrightarrow{OP_0} \quad and \ \vec{x} = \overrightarrow{OP}$$

Is P on the plane?

Easy to answer with the normal vector

$$\vec{w} \cdot (\vec{x} - \vec{x}_0) = 0$$

$$\vec{w} \cdot \vec{x} - \vec{w} \cdot \vec{x}_0 = 0 \quad with \quad b = -\vec{w} \cdot \vec{x}_0$$

$$\vec{w} \cdot \vec{x} + b = 0$$

Winnow: Learning

Learning means modifying the weights of the vector W.

We only learn from misclassified instances (do nothing when we predict the correct decision)

If the predicted class in *incorrect*

if A_j belongs to Class 1

for each $a_{ji} = 1$, multiply w_i by α (promotion)

otherwise

for each $a_{ji} = 1$, divide w_i by α (demotion)

This is a *mistake driven* approach.

The user must specify the value for α (> 1) and θ.

Start with w_i = constant (e.g., 1)

Winnow: Example

Use our weather problem

We specify the value for $\alpha = 2$ and $\theta = 2$
(e.g., half of the mean number of $a_{ji} = 1$).

Start with w_i = constant = 1.

But the features are not binary!
Transform them into (many) binary features

Example: Weather problem

Outlook	Temperature	Humidity	Windy?	Play?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Binary representation

ID	sun?	over?	rain?	hot?	mild?	cool?	high?	norm?	Windy?	Play?
A1	1	0	0	1	0	0	1	0	0	0
A2	1	0	0	1	0	0	1	0	1	0
A3	0	1	0	1	0	0	1	0	0	1
A4	0	0	1	0	1	0	1	0	0	1
A5	0	0	1	0	0	1	0	1	0	1
A6	0	0	1	0	0	1	0	1	1	0
A7	0	1	0	0	0	1	0	1	1	1
A8	1	0	0	0	1	0	1	0	0	0
A9	1	0	0	0	0	1	0	1	0	1
A10	0	0	1	0	1	0	0	1	0	1
A11	1	0	0	0	1	0	0	1	1	1
A12	0	1	0	0	1	0	1	0	1	1
A13	0	1	0	1	0	0	0	1	0	1
A14	0	0	1	0	1	0	1	0	1	0

Winnow: Example

We specify the value for α = 2 and θ = 2

Start with w_i = constant = 1

W	1	1	1	1	1	1	1	1	1	
A1	1	0	0	1	0	0	1	0	0	**0**

Decision: 1 + 1 + 1 = 3 > 2, decision Class 1. Incorrect

Divide by α = 2

W	0.5	1	1	0.5	1	1	0.5	1	1	
A11	1	0	0	0	1	0	0	1	0	**1**

Decision: 0.5 + 1 + 1 = 2.5 > 2, decision Class 1. Correct

Winnow: Example

…

W	0.5	1	1	0.5	1	1	0.5	1	1	
A6	0	0	1	0	0	1	0	1	0	**0**

Decision: 1 + 1 + 1 = 3 > 2, decision Class 1. Incorrect
Divide by $\alpha = 2$

W	0.5	1	0.5	1	1	0.5	0.5	0.5	0.5	
A14	0	0	1	0	1	0	1	0	1	**0**

…

Winnow

We need to learn from all training examples.

The order of the presentation may have an impact on the quality of the resulting model. (why not a random order?)

(one epoch = set of all training examples)

Consider many epochs ?
until we have a perfect separation
or no improvement…

No guarantee that a *linear* boundary exists between the two classes…

Overview

- Simple Winnow
- **Variants of Winnow**

Balanced Winnow

The (unbalanced / simple) Winnow algorithm does not include negative weights.

In some applications, this could be a problem

Replace the vector W by two vectors, W^+ and W^-.

The decision rule is

Classify in Class 1 if

$$d_j = \sum_{i=1}^{m} (w_i^+ - w_i^-) \cdot a_{ji} > \theta$$

Balanced Winnow: Learning

Modifying the weights of the vector W^+ and W^-

We still only learn from misclassified instances

If the predicted class in incorrect

if A_j belongs to Class 1

for each $a_{ji} = 1$, multiply w^+_i by α, divide w^-_i by α

otherwise

for each $a_{ji} = 1$, divide w^+_i by α, multiply w^-_i by α

The weights w^+_i and w^-_i are non-negative.

Instead of dividing by α, we can multiply by β, another user-defined parameter

Balanced Winnow: Example

We specify the value for $\alpha = 2$ and $\theta = 2$

Fix w_i^+ = constant = $\theta / 2$ and w_i^- = constant = $\theta / 4$

W+	1	1	1	1	1	1	1	1	1	
W-	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
A3	0	1	0	1	0	0	1	0	0	**1**

Decision: (1-0.5) + (1-0.5) + (1-0.5) = 1.5 > 2, decision Class 0. Incorrect

W+	1	2	1	2	1	1	2	1	1	
W-	0.5	0.25	0.5	0.25	0.5	0.5	0.25	0.5	0.5	
A12	0	1	0	0	1	0	1	0	1	**1**

Margin Winnow

It will be important that the boundary is, when possible, not too close to some points (instances).

Use two boundaries, with the decision rule

$$Class\ 1\ if\ \sum_{i=1}^{m} w_i \cdot a_{ji} > \theta^+$$

$$Class\ 0\ if\ \sum_{i=1}^{m} w_i \cdot a_{ji} < \theta^-$$

incorrect if the sum is between the two limits

Multi-Class Winnow

When we have more than two classes?
Let *r* the number of possible assignments.

Maintain non-negative weights (to be learned).

We have *r* vectors W, one for each class.

Learning: mistake-driven learning.

Multiplicative updates

Each class is represented by a vector of *m* (features) + 1 for the *i*th class: $W^i = [w^i_0, w^i_1, w^i_2, \ldots, w^i_m]$

Each vector W^i is initialized with $w^i_j = 1$
w^i_0 is the bias (with the corresponding feature always = 1)

Multi-Class Winnow

Each instance is represented by a vector X of size $m + 1$ (the number of features + 1)

$$X = [1, x_1, x_2, \ldots, x_m]$$

These values x_j can take any value in [0,1]

and with $x_0 = 1$

Decision:

The predicted class t for a given instance X is

$\text{Arg Max}_i\ X \cdot W^i$ over the $i = 1, \ldots, r$ possible class

Learning:

Present the training set in a random order, and repeat this epoch s times (e.g., $s = 10$)

Multi-Class Winnow

With *t* the correct target class and *p* the predicted class,
α the positive learning constant,

If the predicted class *p* in incorrect (we expect *t*)

$w^t_j = w^t_j \cdot (1 + \alpha \cdot x_j)$ # increase the weights

$w^p_j = w^p_j / (1 + \alpha \cdot x_j)$ # decrease the weights

Mesterharm C., A Multi-class Linear Learning Algorithm Related to Winnow. Advances in Neural Information Processing Systems, 12, 200, p. 519-525

Schler J., Koppel, M., Argamon, S., Pennebacker, J., Effects of Age and Gender on Blogging. Processing AAAI, 2005

Multi-Class Winnow: Example

Available examples

ID	F1	F2	F3	F4	F5	F6	Class
1	0.5	0.01	0.45	0.02	0.25	0.02	1
2	0.3	0.02	0.52	0.05	0.3	0.09	1
3	0.25	0.03	0.49	0.09	0.26	0.05	1
4	0.7	0.04	0.39	0.05	0.32	0.03	1
5	0.02	0.05	0.47	0.04	0.31	0.03	1
6	0.03	0.5	0.12	0.22	0.02	0.04	2
7	0.01	0.45	0.2	0.18	0.09	0.03	2
8	0.06	0.54	0.18	0.17	0.11	0.01	2
9	0.05	0.6	0.09	0.21	0.09	0.05	2
10	0.02	0.65	0.26	0.16	0.07	0.03	2
11	0.2	0.1	0.01	0.5	0.31	0.02	3
12	0.1	0.3	0.02	0.48	0.37	0.07	3

Multi-Class Winnow: Example

Example #8

Id	F1	F2	F3	F4	F5	F6	
8	0.06	0.54	0.18	0.17	0.11	0.01	Class=2
W^1	1	1	1	1	1	1	Bias 0
W^2	1	1	1	1	1	1	Bias 1
W^3	1	1	1	1	1	1	Bias 2

Predict: Class 3 (Arg max = 3.07)

Update: (α = 2)

Incorrect decision:

- decrease the predicted class (W^3)
- increase the correct class (W^2)
- no change for the rest

Multi-Class Winnow: Example

Example #8

For W^2: $w_j = w_j \cdot (1 + \alpha \cdot x_j)$ # 1 * (1+2*0.06) = 1.12

For W^3: $w_j = w_j / (1 + \alpha \cdot x_j)$ # 1 / (1+2*0.06) = 0.89

Id	F1	F2	F3	F4	F5	F6	
8	0.06	0.54	0.18	0.17	0.11	0.01	Class=2
W^1	1	1	1	1	1	1	Bias 0
W^2	1.12	2.08	1.36	1.34	1.22	1.02	3.00
W^3	0.89	0.48	0.74	0.75	0.82	0.98	0.67

With the new weights: Prediction: Class 2
(Arg max = 4.81 {1.07, 4.81, 1.34})

Conclusion

- Linear separating (hyper)plan are well-known approaches
- Simple model with two classes, variants with more than two classes
- Learning: the weights of the boundary (hyper)plan
- Other new idea: repeat the learning process (epoch)
- Linear separable? not always the case (stop when no real improvement)
- Not so clear to explain the proposed decision…

Multi-Class Winnow

With *r* classes, and

with *t* the correct class and *o* the output class,

α the positive learning constant,

If the predicted class *t* in incorrect (we expect *o*)

$w^t_j = w^t_j \cdot \alpha^{(x_j^o - x_j^t)}$

then normalize the weights $sum_j (w^t_j) = 1$

When x_j^i are in {0,1}

and $x_j^o = 1$, $x_j^t = 0$, we multiply by α

and $x_j^o = 0$, $x_j^t = 1$, we divide by α

otherwise no change

Mesterharm C., A Multi-class Linear Learning Algorithm Related to Winnow. Advances in Neural Information Processing Systems, 12, 200, p. 519-525