Learning and Linear Classifiers

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: V. Hlaváč, J. Matas, O. Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

http://cmp.felk.cvut.cz

7/Nov/2016

Last update: 3/Nov/2016

LECTURE PLAN

- The problem of classifier design.
- Learning in pattern recognition.
- Linear classifiers.
- Perceptron algorithms.
- Optimal separating plane with the Kozinec algorithm.

Classifier Design (1)

The object of interest is characterised by observable properties $\mathbf{x} \in X$ and its class membership (unobservable, hidden state) $k \in K$, where X is the space of observations and K the set of hidden states.

The objective of classifier design is to find a strategy $q^*\colon X \to K$ that has some optimal properties.

Bayesian decision theory solves the problem of minimisation of risk

$$R(q) = \sum_{\mathbf{x},k} p(\mathbf{x},k)\, W(q(\mathbf{x}),k)$$

given the following quantities:

- $p(\mathbf{x},k), \forall \mathbf{x} \in X, k \in K$ – the statistical model of the dependence of the observable properties (measurements) on class membership
- $W(q(\mathbf{x}),k)$ the loss of decision $q(\mathbf{x})$ if the true class is k

Classifier Design (2)

Non-Bayesian decision theory solves the problem if $p(\mathbf{x}|k), \forall\mathbf{x} \in X, k \in K$ are known, but $p(k)$ are unknown (or do not exist). Constraints or preferences for different errors depend on the problem formulation.

However, in applications typically:

- none of the probabilities are known. The designer is only given a training multiset $\mathcal{T} = \{(\mathbf{x}_1, k_1) \ldots (\mathbf{x}_L, k_L)\}$, where L is the length (size) of the training multiset.
- the desired properties of the classifier $q(\mathbf{x})$ are known

Classifier Design via Parameter Estimation

- Assume $p(\mathbf{x}, k)$ have a particular form, e.g. Gaussian (mixture), piece-wise constant, etc., with a finite (i.e. small) number of parameters θ_k.
- Estimate the parameters from the using training set $\mathcal{T}$
- Solve the classifier design problem (e.g. risk minimisation), substituting the estimated $\hat{p}(\mathbf{x}, k)$ for the true (and unknown) probabilities $p(\mathbf{x}, k)$

? : What estimation principle should be used?

– : There is no direct relationship between known properties of estimated $\hat{p}(\mathbf{x}, k)$ and the properties (typically the risk) of the obtained classifier $q'(\mathbf{x})$

– : If the true $p(\mathbf{x}, k)$ is not of the assumed form, $q'(\mathbf{x})$ may be arbitrarily bad, even for the asymptotic case $L \to \infty$.

+ : Implementation is often straightforward, especially if parameters θ_k for each class are assumed independent.

+ : Performance on test data can be predicted by crossvalidation.

Learning in Statistical Pattern Recognition

- Choose a class Q of decision functions (classifiers) $q: X \to K$.
- Find $q^* \in Q$ minimising some criterion function on the training set that approximates the risk $R(q)$ (true risk is uknown).
- Objective functions:

Empirical risk R_{emp} (training set error) minimization. True risk approximated by

$$R_{emp}(q_\theta(\mathbf{x})) = \frac{1}{L}\sum_{i=1}^{L} W(q_\theta(\mathbf{x}_i), k_i) ,$$

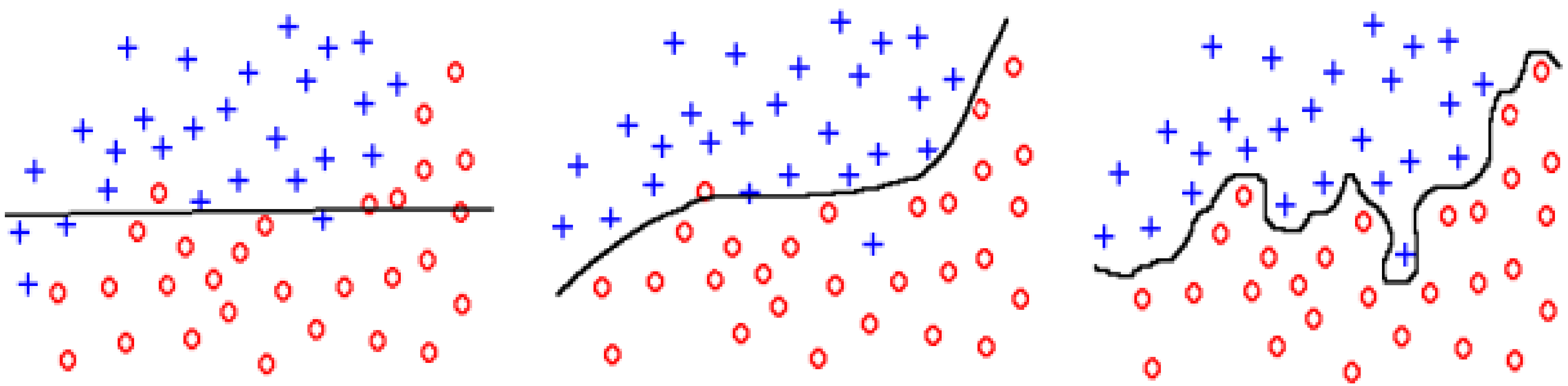
$$\theta^* = \operatorname*{argmin}_\theta R_{emp}(q_\theta(\mathbf{x}))$$

Examples: Perceptron, Neural nets (Back-propagation), etc.

Structural risk minimization.
Example: SVM (Support Vector Machines).

Overfitting and Underfitting

- How wide a class Q of classifiers $q_\theta(\mathbf{x})$ should be used?
- The problem of generalization is a key problem of pattern recognition: a small empirical risk R_{emp} need not imply a small true expected risk R.



underfit fit overfit

As discussed previously, a suitable model can be selected e.g. using cross-validation.

Structural Risk Minimization Principle (1)

We would like to minimise the risk

$$R(q) = \sum_{x,k} p(\mathbf{x}, k)\, W(q_\theta(\mathbf{x}), k)\,,$$

but $p(\mathbf{x}, k)$ is unknown.

Vapnik and Chervonenkis proved a remarkable inequality

$$R(q) \leq R_{emp}(q) + R_{str}\left(h, \frac{1}{L}\right)\,,$$

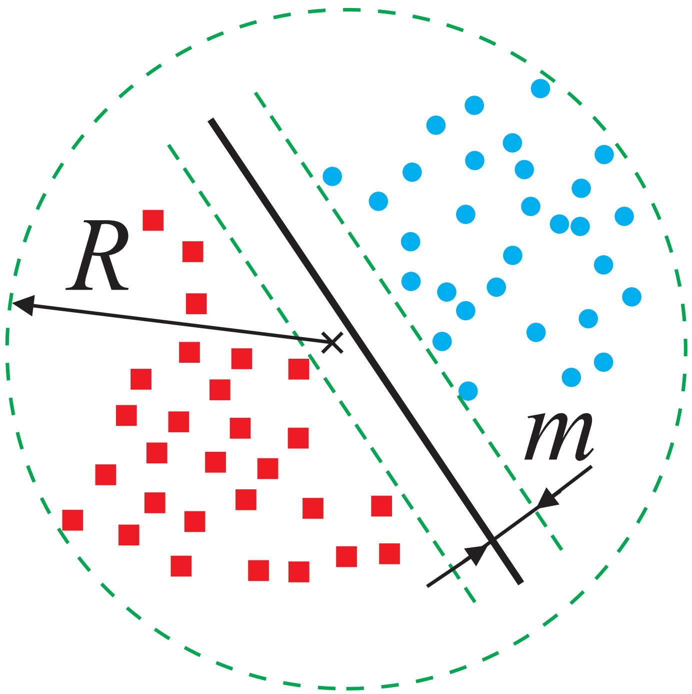
where h is VC dimension (capacity) of the class of strategies Q.

Notes:

+ R_{str} does not depend on the unknown $p(\mathbf{x}, k)$
+ R_{str} known for some classes of Q, e.g. linear classifiers.

Structural Risk Minimization Principle (2)

- There are more types of upper bounds on the expected risk.
  E.g. for linear discriminant functions



VC dimension (capacity)

$$h \leq \frac{R^2}{m^2} + 1$$

- Examples of learning algorithms: SVM or ε-Kozinec.

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w},b}{\text{argmax}} \; \min \left(\min_{x \in X_1} \frac{\mathbf{w} \cdot \mathbf{x} + b}{|\mathbf{w}|}, \min_{x \in X_2} \frac{\mathbf{w} \cdot \mathbf{x} + b}{|\mathbf{w}|} \right) .$$

Empirical Risk Minimisation, Notes

Is then empirical risk minimisation = minimisation of training set error, e.g. neural networks with backpropagation, useless? No, because:

– R_{str} may be so large that the upper bound is useless.

+ Vapnik's theory justifies using empirical risk minimisation on classes of functions with finite VC dimension.

+ Vapnik suggests learning with progressively more complex classes Q.

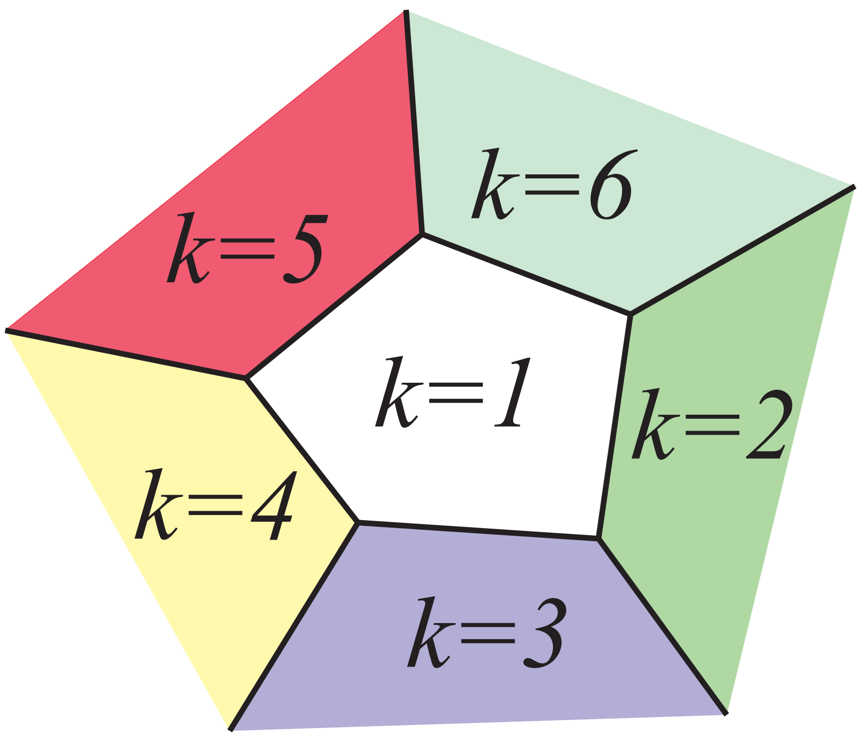
+ Empirical risk minimisation is computationally hard (impossible for large L). Most classes of decision functions Q where empirical risk minimisation (at least local) can be effeciently organised are often useful.

Linear Classifiers

- For some statistical models, the Bayesian or non-Bayesian strategy is implemented by a linear discriminant function.
- Capacity (VC dimension) of linear strategies in an n-dimensional space is $n + 2$. Thus, the learning task is well-posed, i.e., strategy tuned on a finite training multiset does not differ much from correct strategy found for a statistical model.
- There are efficient learning algorithms for linear classifiers.
- Some non-linear discriminant functions can be implemented as linear after the feature space transformation.

Linear Discriminant Function

- $f_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} + w_{0,k}$
- A strategy $k^* = \operatorname*{argmax}_k f_k(\mathbf{x})$ divides X into $|K|$ convex regions.



Linear Separability (Two Classes)

Consider a dataset $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), ..., (\mathbf{x}_L, k_N)\}$, with $\mathbf{x}_i \in \mathbb{R}^D$ and $k_i \in \{-1, 1\}$ $(i = 1, 2, ..., L.)$

The data are **linearly separable** if there exists a hyperplane which divides $\mathbb{R}^D$ to two half-spaces such that the data of a given class are all in one half-space.

Formally, the data are linearly separable if

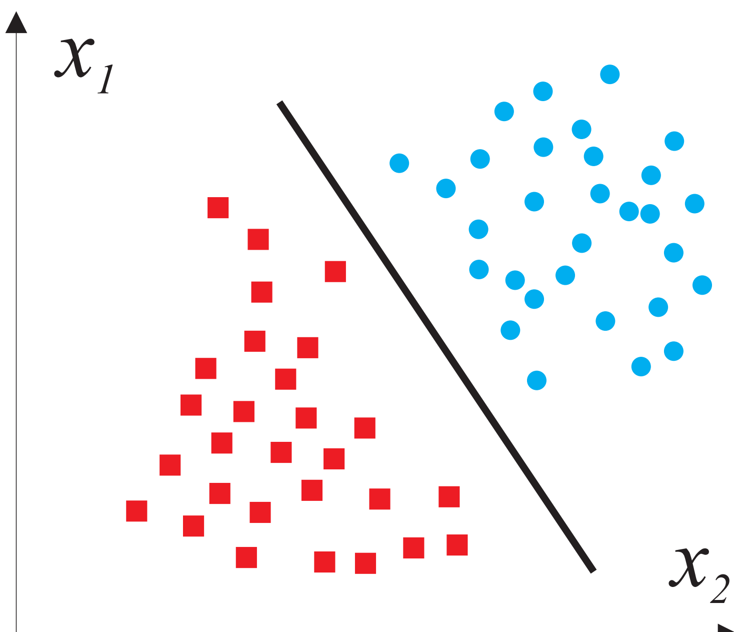
$$\exists \mathbf{w} \in \mathbb{R}^{D+1} \colon \text{ sign}\left(\mathbf{w} \cdot \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}\right) = k_i \quad \forall i = 1, 2, ..., L\,. \tag{1}$$

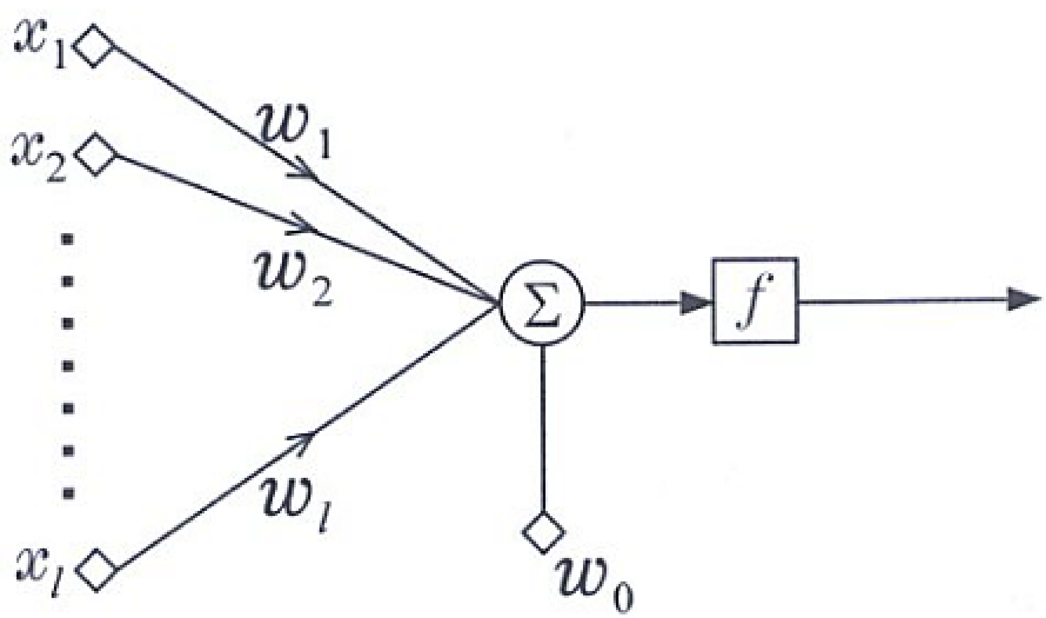
Example of linearly separable data is on the next slide.

Dichotomy, Two Classes Only

$|K| = 2$, i.e. two hidden states (typically also classes)

$$q(x) = \begin{cases} k = 1\,, & \text{if} \quad \mathbf{w} \cdot \mathbf{x} + w_0 > 0\,, \\ k = -1\,, & \text{if} \quad \mathbf{w} \cdot \mathbf{x} + w_0 < 0\,. \end{cases} \tag{2}$$





Perceptron Classifier

Input: $\mathcal{T} = \{(\mathbf{x}_1, k_1) \ldots (\mathbf{x}_L, k_L)\}, \qquad k \in \{-1, 1\}$

Goal: Find a weight vector w and offset w_0 such that :

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_j + w_0 > 0 \quad \text{if} \quad k_j = 1\,, \qquad (\forall j \in \{1, 2, ..., L\}) \\ \mathbf{w} \cdot \mathbf{x}_j + w_0 < 0 \quad \text{if} \quad k_j = -1 \end{aligned} \tag{3}$$

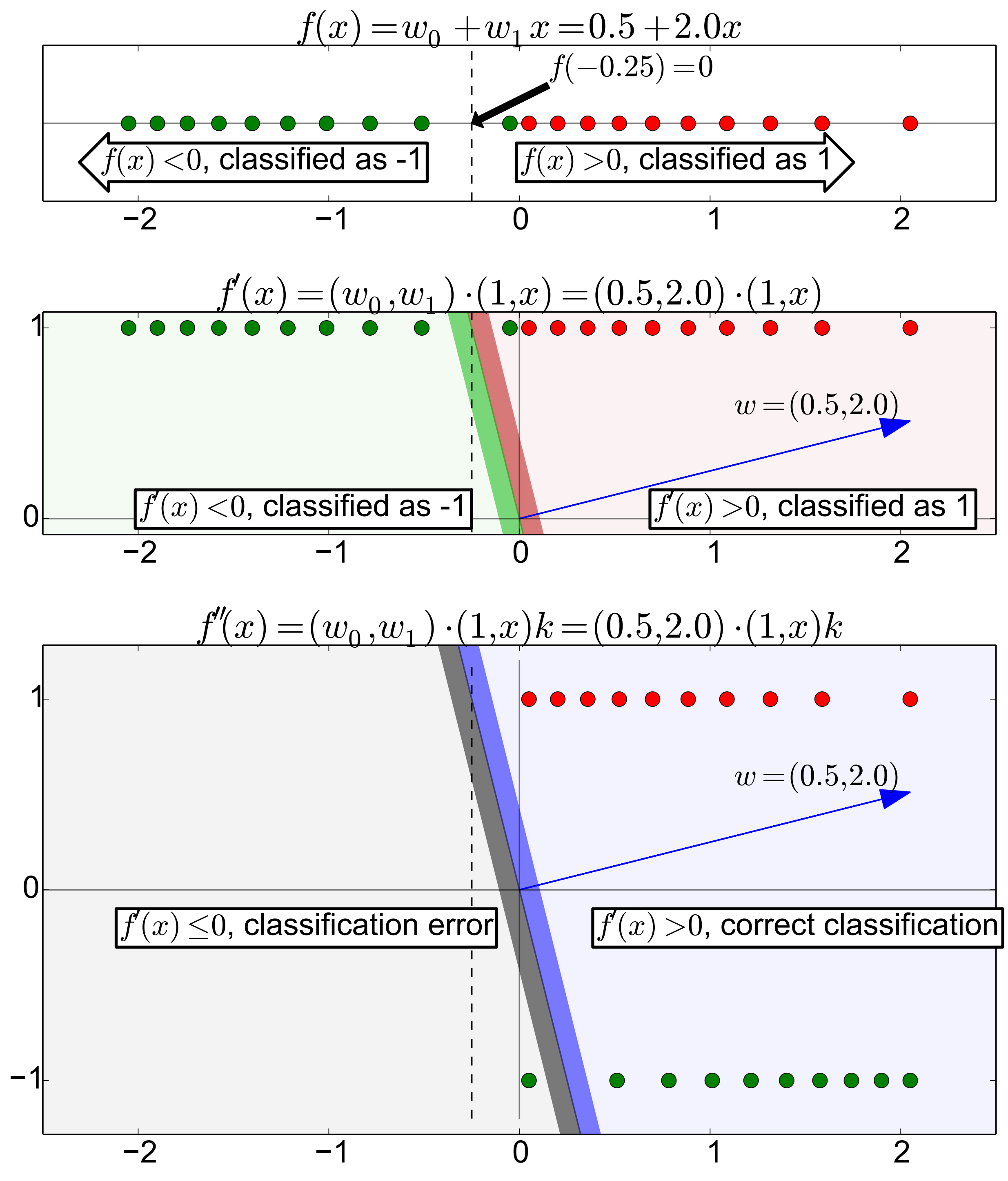
Equivalently, (as in the logistic regression lecture), with $\mathbf{x}' = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$ and $\mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$:

$$\begin{aligned} \mathbf{w}' \cdot \mathbf{x}'_j > 0 \quad \text{if} \quad k_j = 1 \qquad (\forall j \in \{1, 2, ..., L\})\,, \\ \mathbf{w}' \cdot \mathbf{x}'_j < 0 \quad \text{if} \quad k_j = -1\,, \end{aligned} \tag{4}$$

or, with $\mathbf{x}''_j = k_j \mathbf{x}'_j$,

$$\mathbf{w}' \cdot \mathbf{x}''_j > 0\,, \qquad (\forall j \in \{1, 2, ..., L\}\,.) \tag{5}$$

Perceptron Classifier Formulation, Example



● class 1, ● class -1

Data points, $x \in \mathbb{R}$

Augmenting by 1's, $\mathbf{x}'_j \in \mathbb{R}^2$

Multiplying by k_j, $k_j \mathbf{x}''_j \in \mathbb{R}^2$

Perceptron Learning: Algorithm

We use the last representation ($\mathbf{x}''_j = k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$, $\mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$) and drop the dashes to avoid notation clutter.

Goal: Find a weight vector $\mathbf{w} \in \mathbb{R}^{D+1}$ (original feature space dimensionality is D) such that:

$$\mathbf{w} \cdot \mathbf{x}_j > 0 \qquad (\forall j \in \{1, 2, ..., L\}) \tag{6}$$

Perceptron algorithm, (Rosenblat 1962):

1. $t = 0$, $\mathbf{w}^{(t)} = 0$.
2. Find a wrongly classified observation $\mathbf{x}_j$:

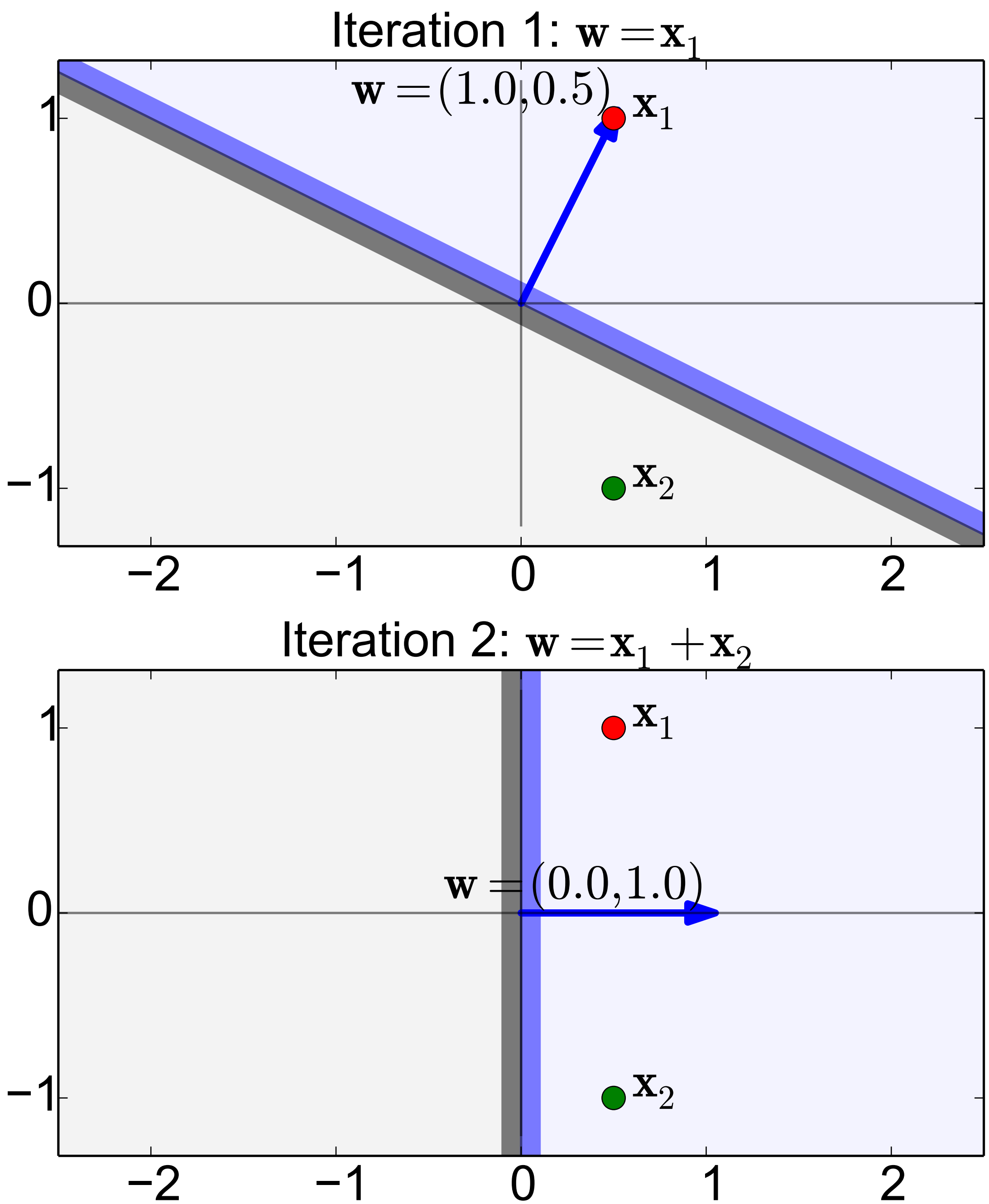
$$\mathbf{w}^{(t)} \cdot \mathbf{x}_j \leq 0, \qquad (j \in \{1, 2, ..., L\}.)$$

3. If there is no misclassified observation then terminate. Otherwise,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{x}_j \; .$$

4. Goto 2.

Perceptron, Example 1

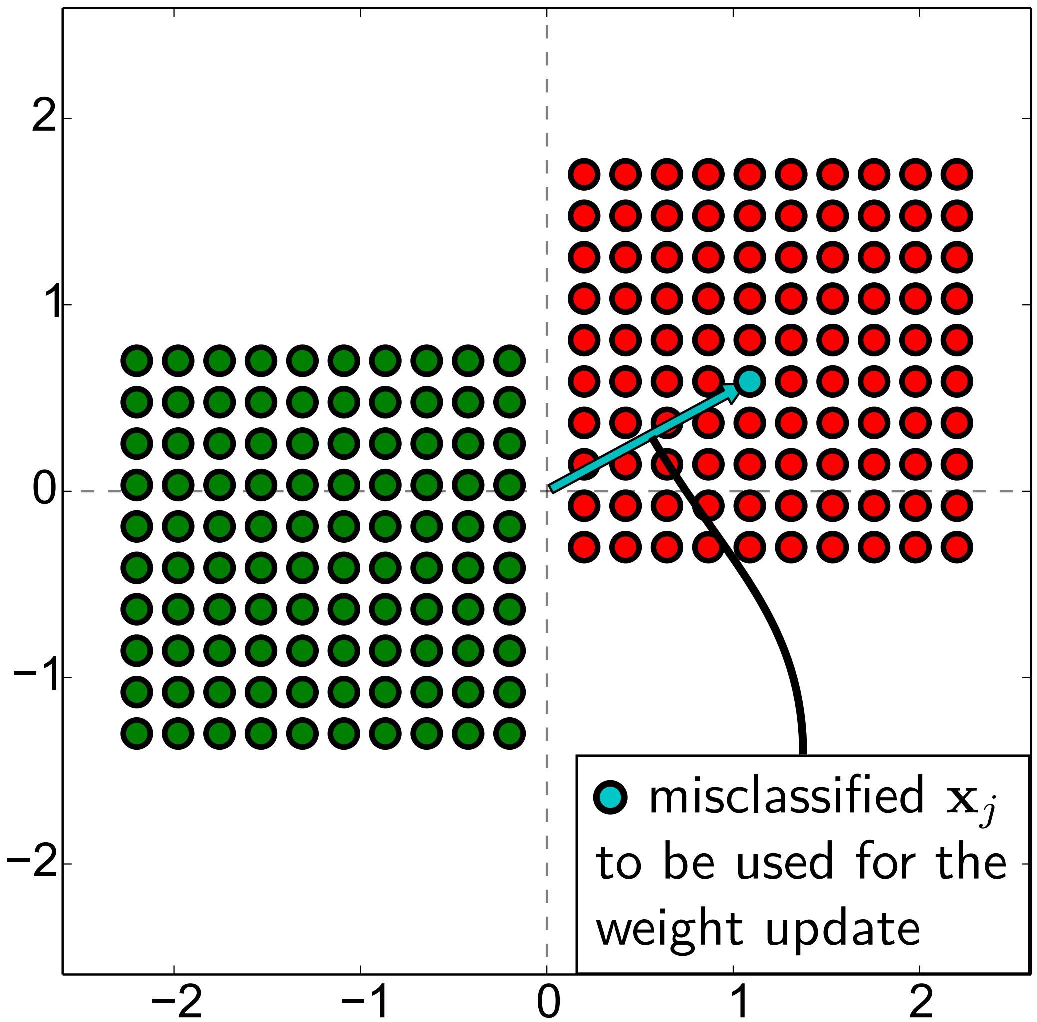


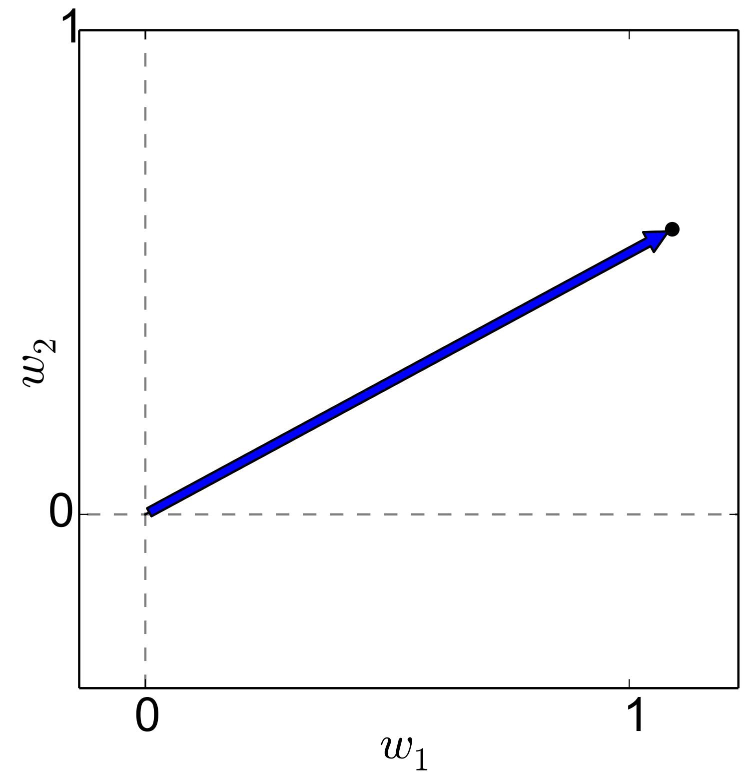
Consider this dataset with 2 points. As $\mathbf{w}^{(0)} = 0$, all points are misclassified. Order the points randomly and go over this dataset. Find the first misclassified point. It is $\mathbf{x}_1$, say. Make the update of weight, $\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} + \mathbf{x}_1$.
Note that $\mathbf{x}_2$ is misclassified.

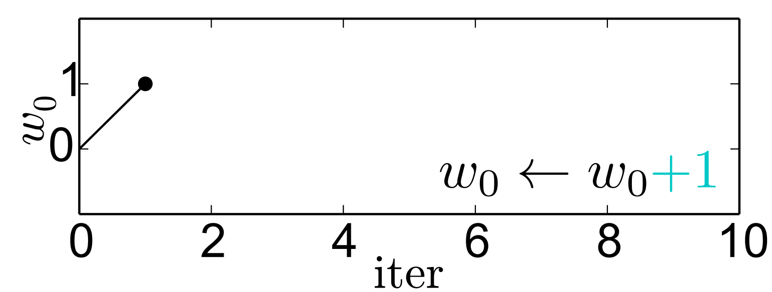
$\mathbf{w}^{(2)} \leftarrow \mathbf{w}^{(1)} + \mathbf{x}_2$. The whole dataset is correctly classified. Done.

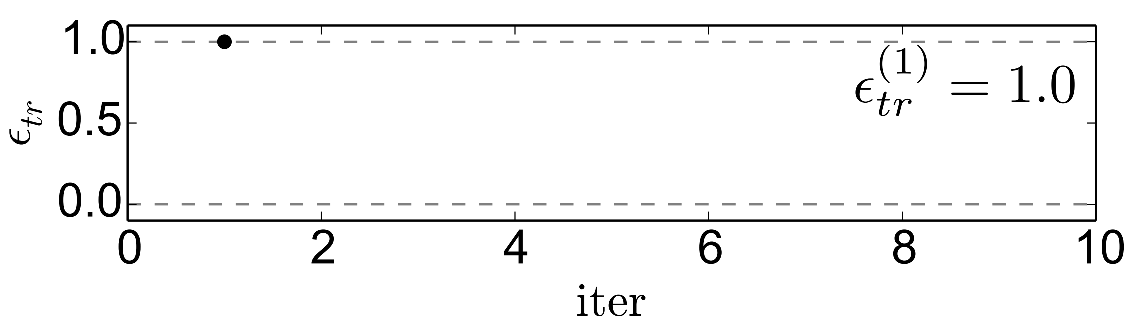
Perceptron, Example 2, Iter. 1

● class 1, ● class -1, ●/●=misclassified,









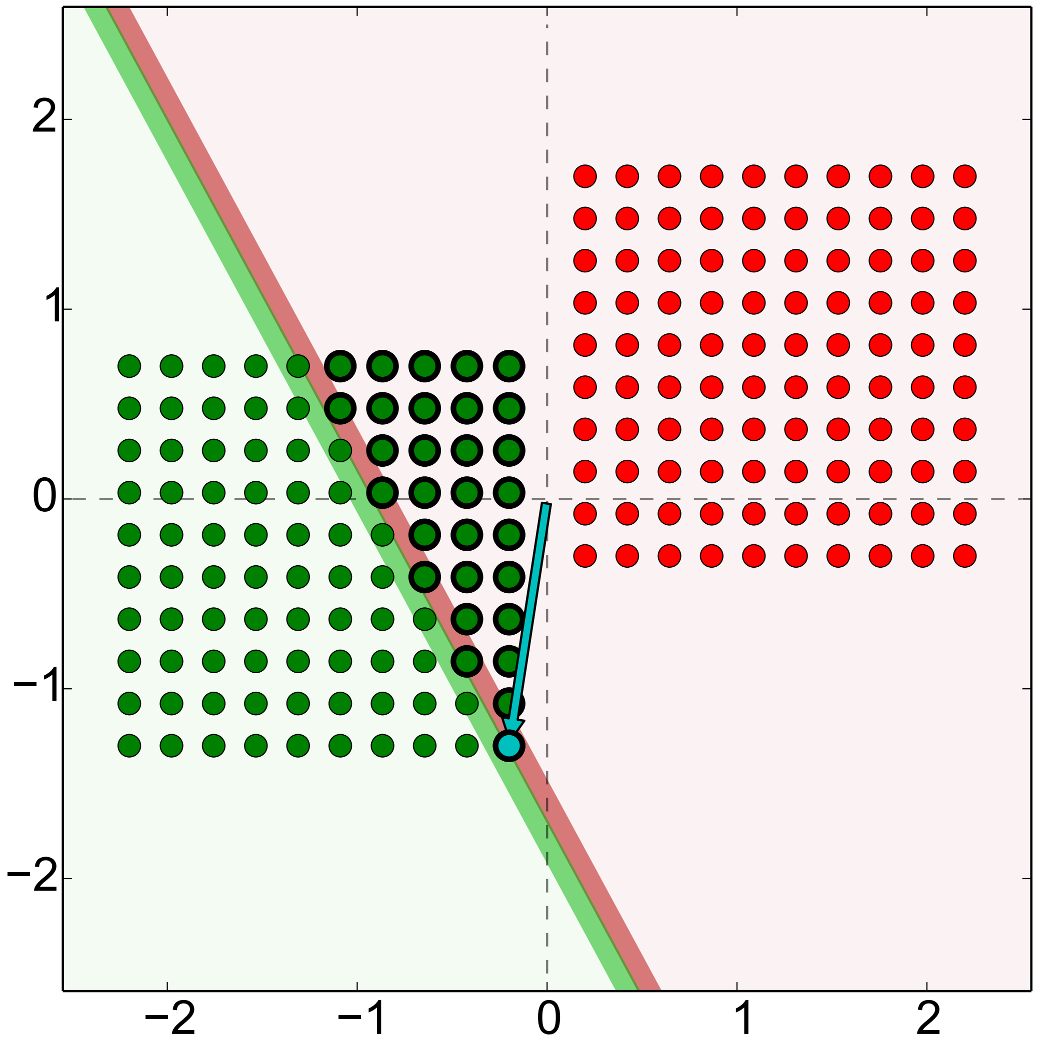
$$\mathbf{w}^{(1)} = \underbrace{\mathbf{w}^{(0)}}_{=0} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$$

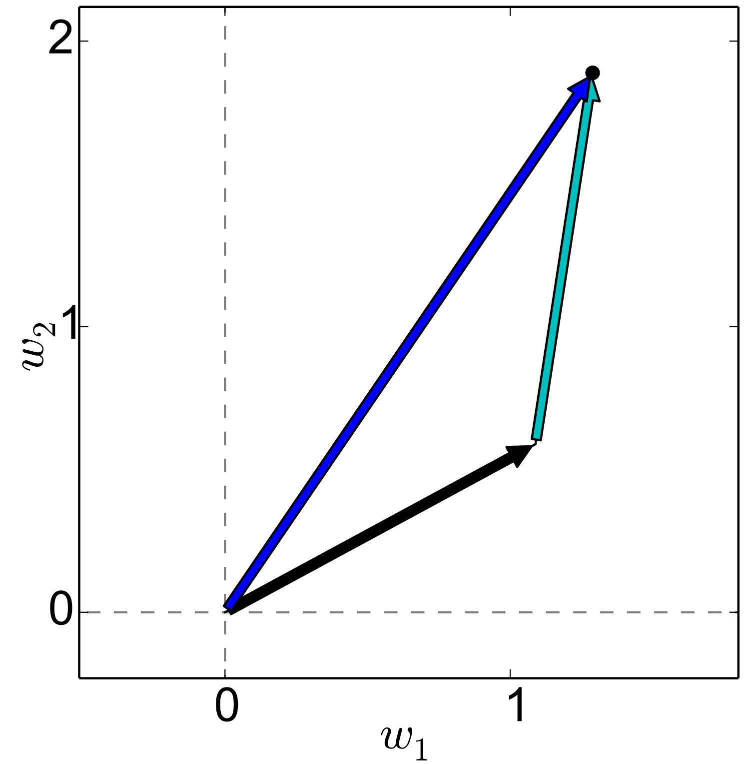
1 point visited so far.

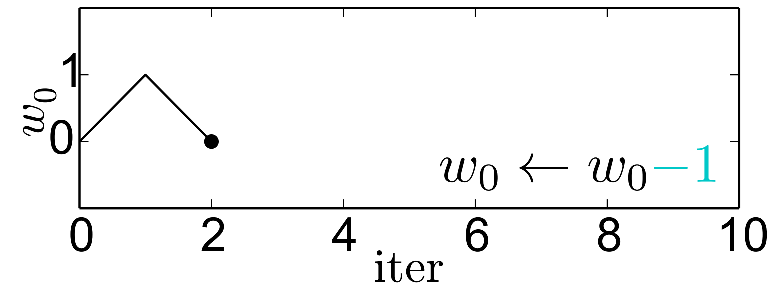
All data are misclassified.

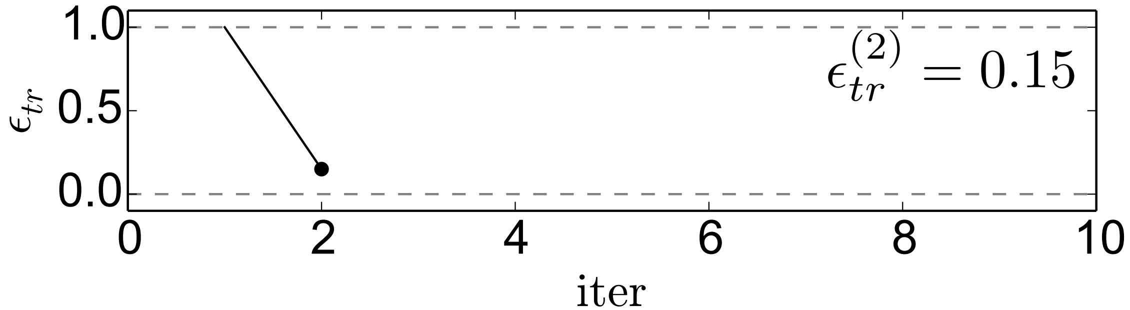
Perceptron, Example 2, Iter. 2

class 1, class -1, ⊙/⊙=misclassified, the weight updater







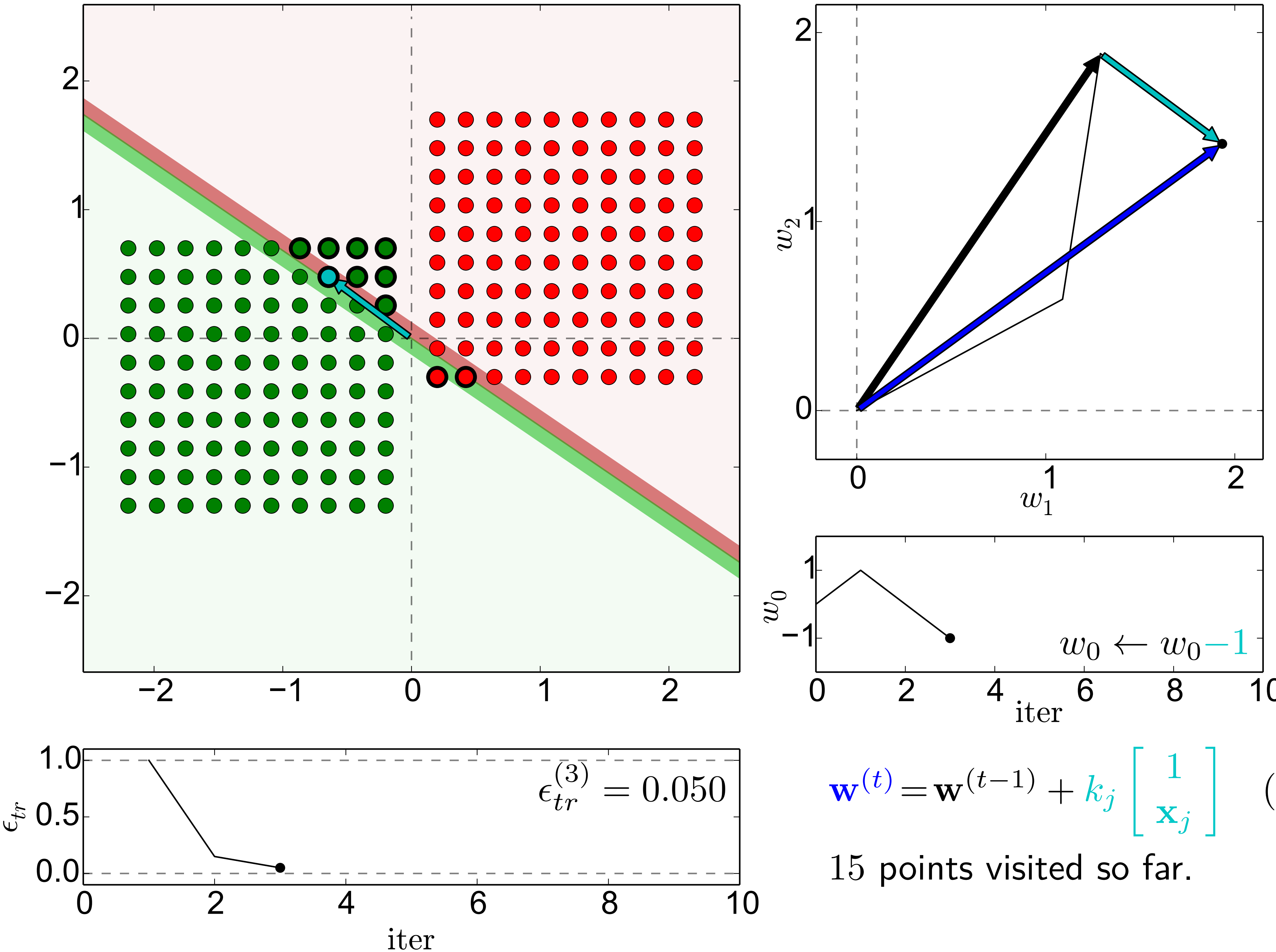


$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$$

2 points visited so far.

Perceptron, Example 2, Iter. 3

● class 1, ● class -1, ◉/◉=misclassified, ◉ the weight updater

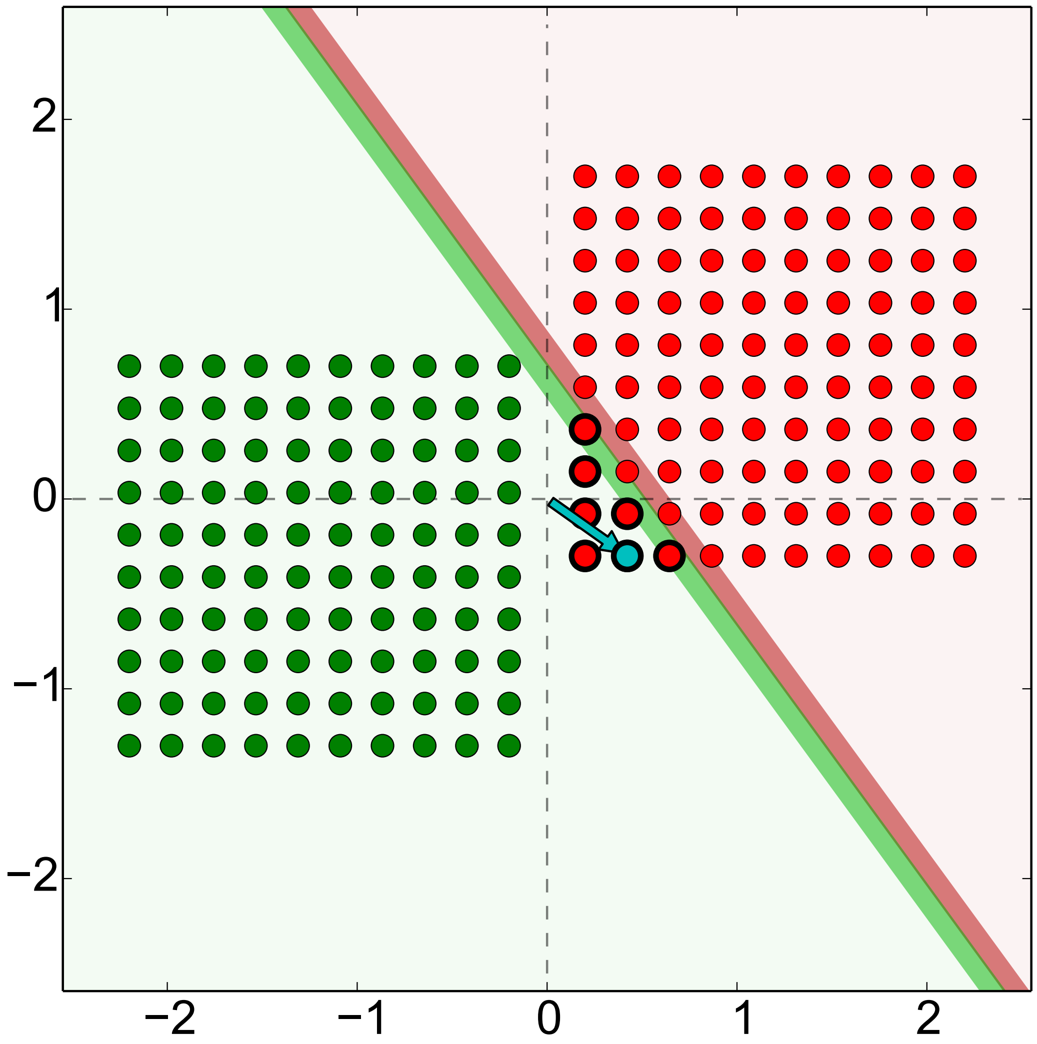


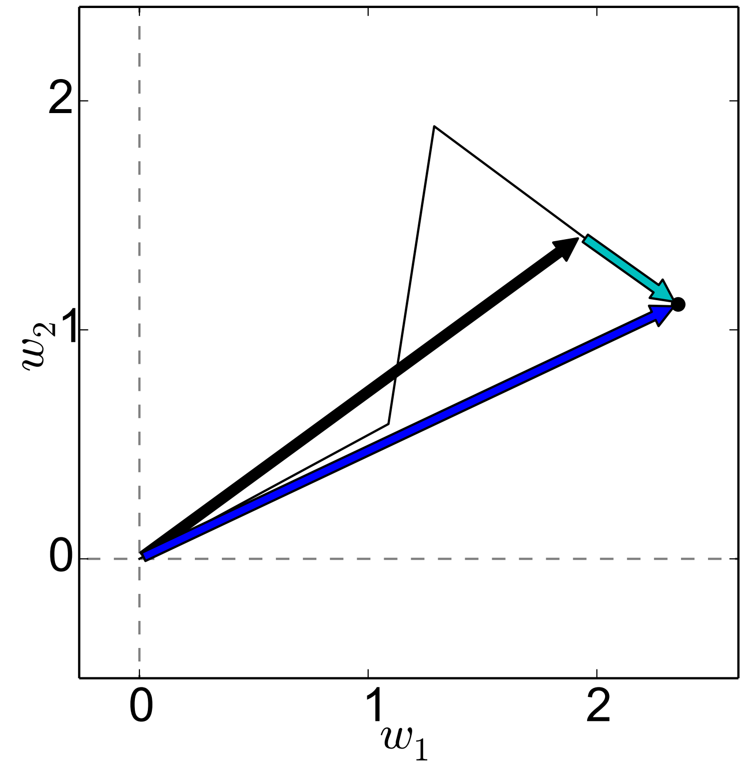
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 3)$$

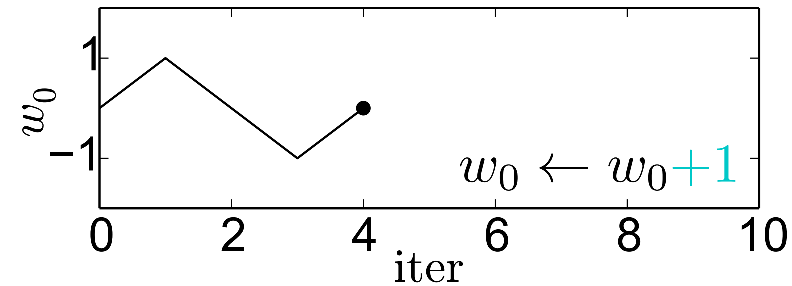
15 points visited so far.

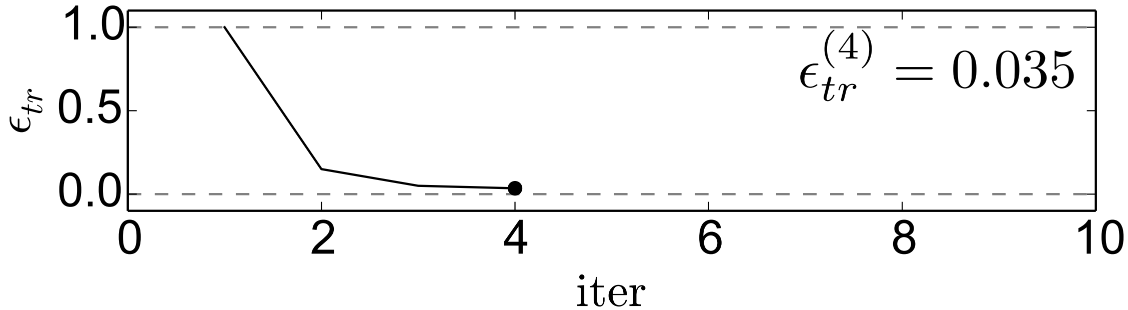
Perceptron, Example 2, Iter. 4

class 1, class -1, ◉/◉=misclassified, ◉ the weight updater







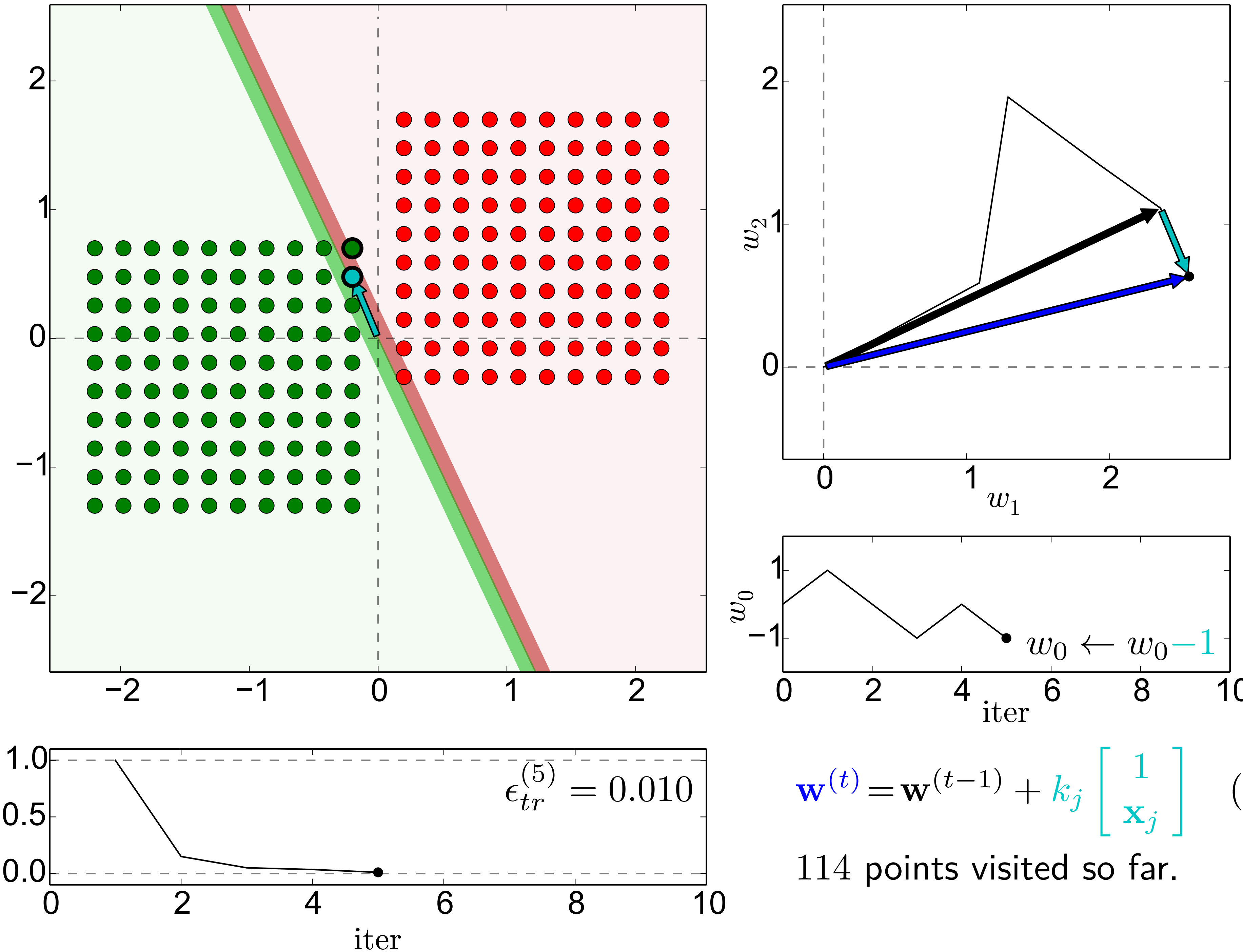


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \qquad (t = 4)$$

19 points visited so far.

Perceptron, Example 2, Iter. 5

class 1, class -1, ○/○=misclassified, ○ the weight updater

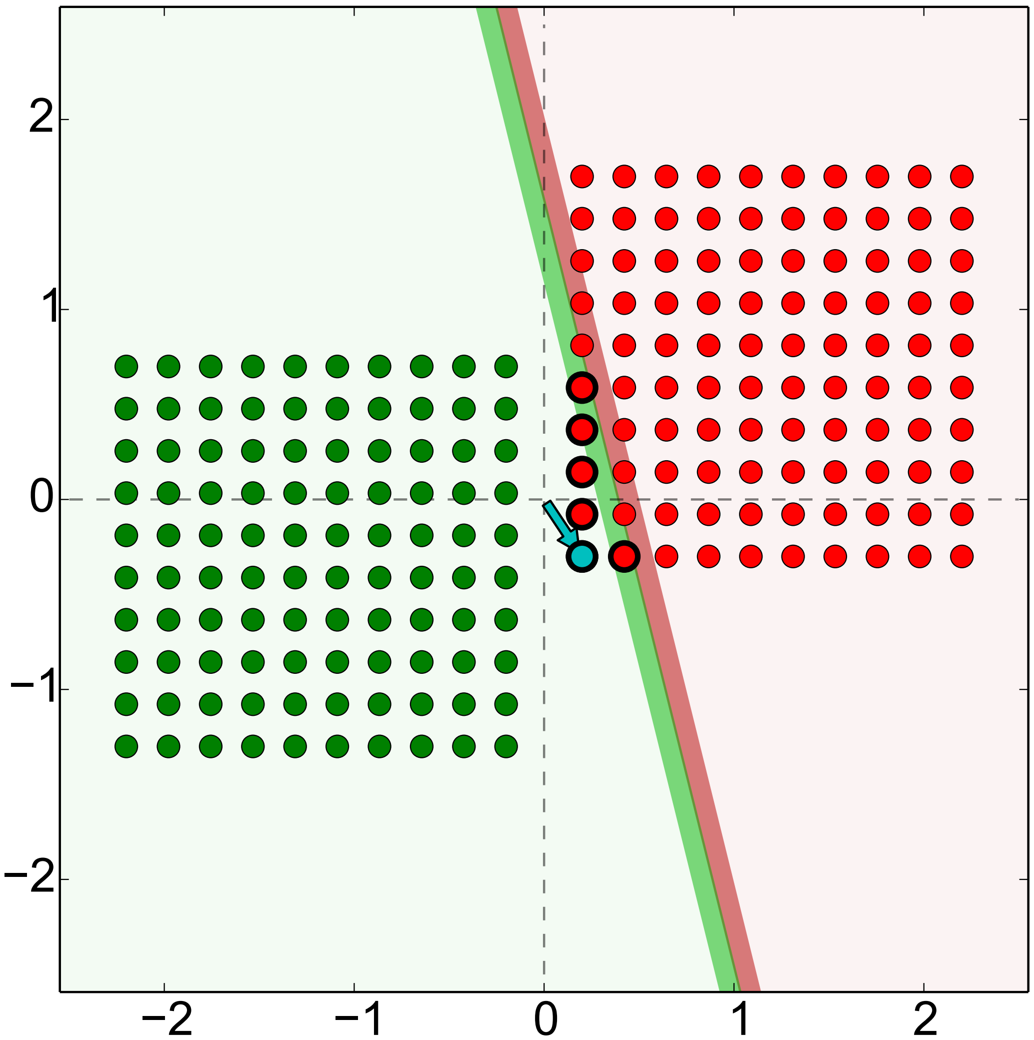


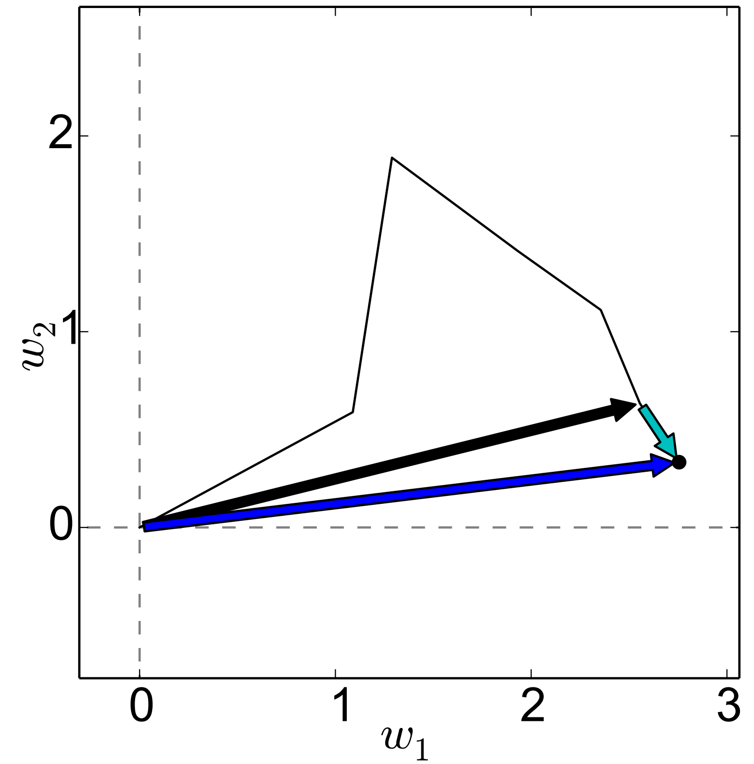
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \qquad (t = 5)$$

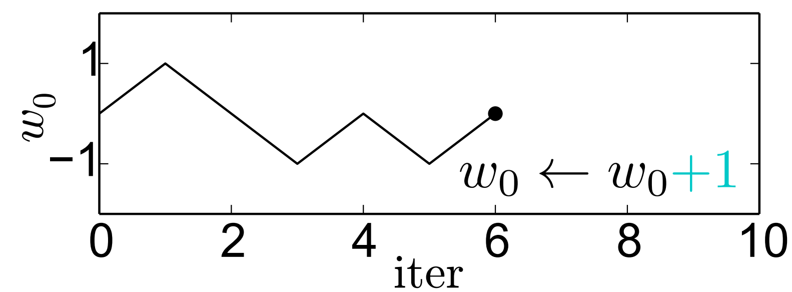
114 points visited so far.

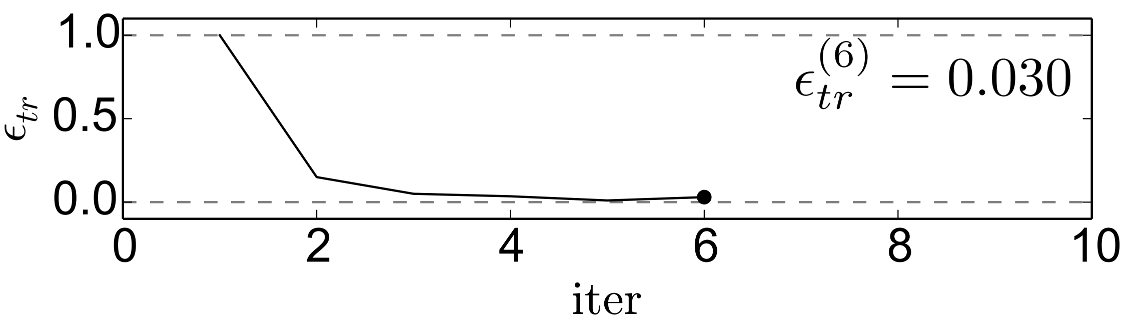
Perceptron, Example 2, Iter. 6

● class 1, ● class -1, ●/●=misclassified, ● the weight updater







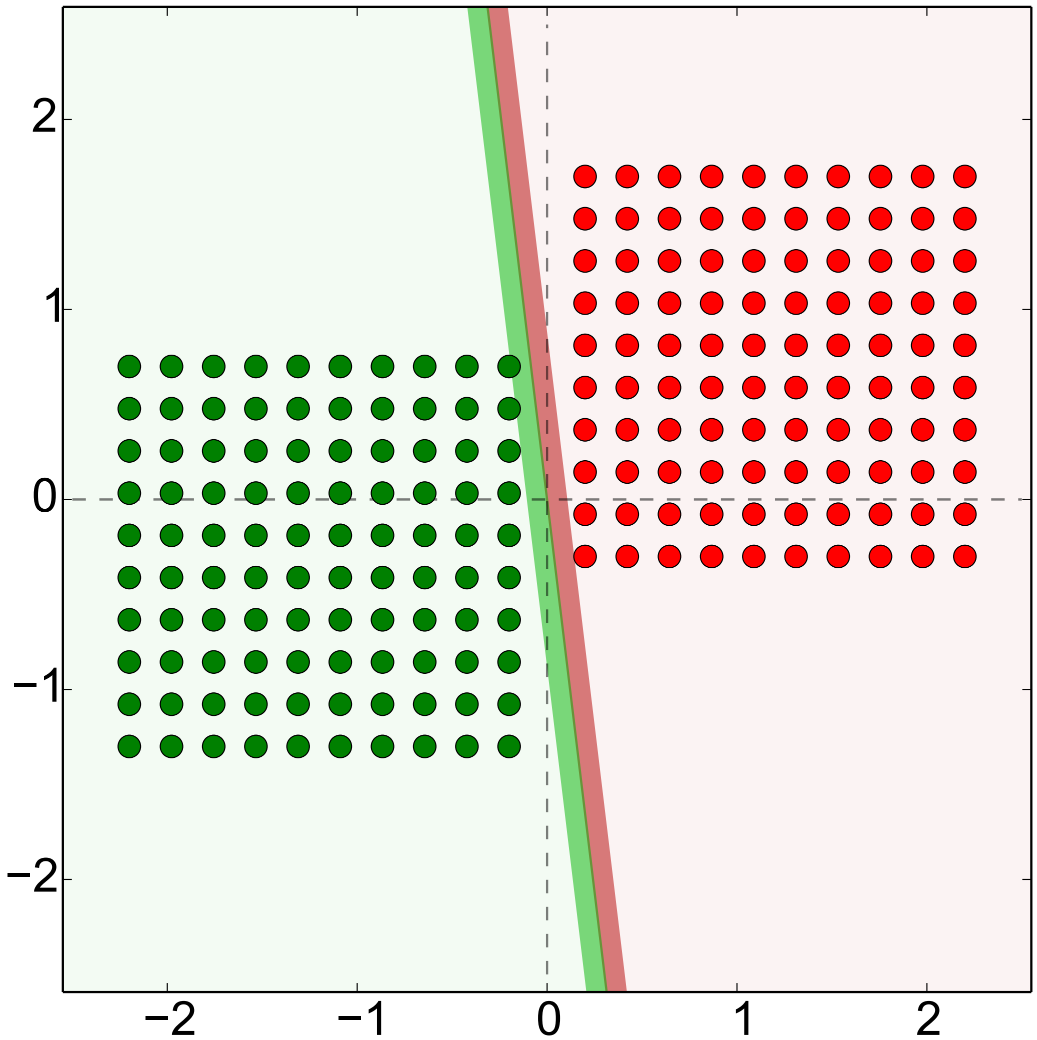


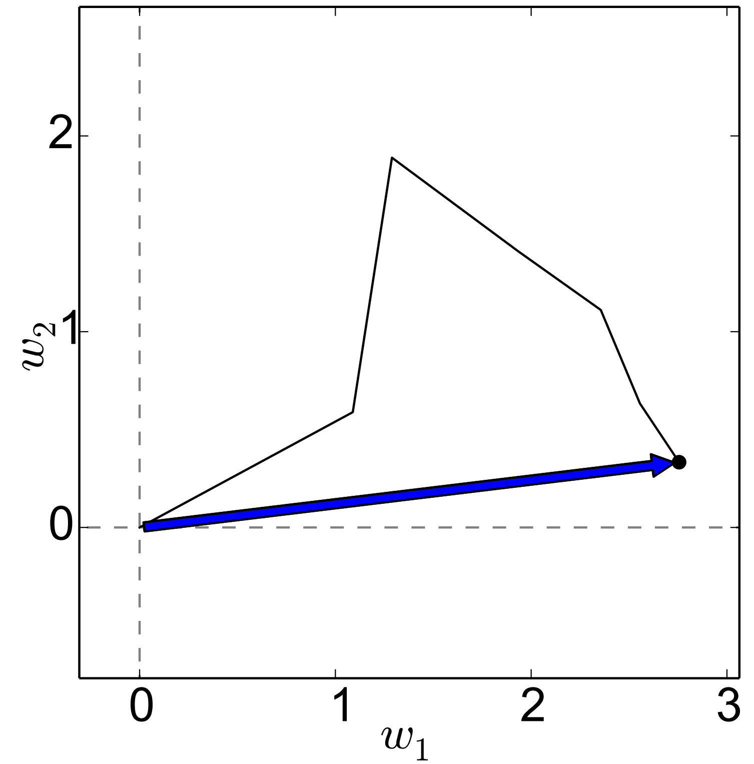
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 6)$$

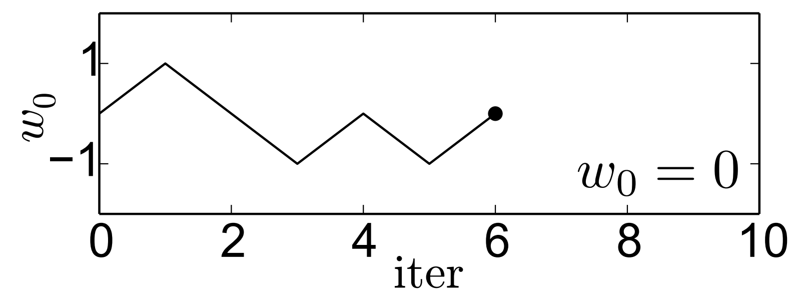
186 points visited so far.

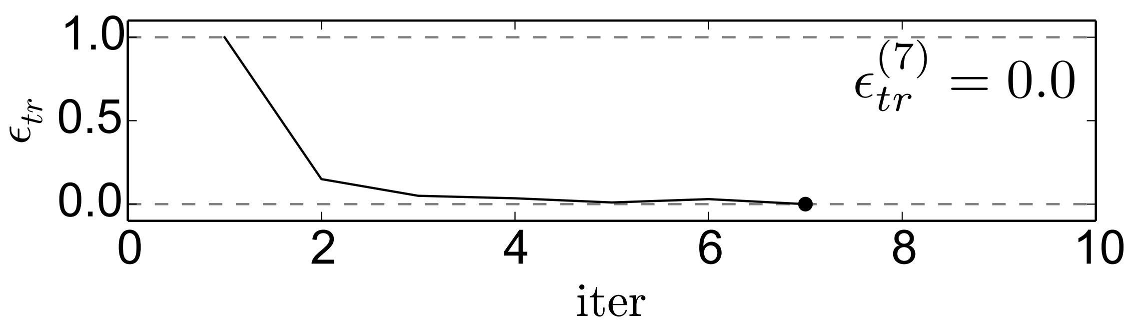
Perceptron, Example 2, Iter. 7

● class 1, ● class -1, ●/●=misclassified,









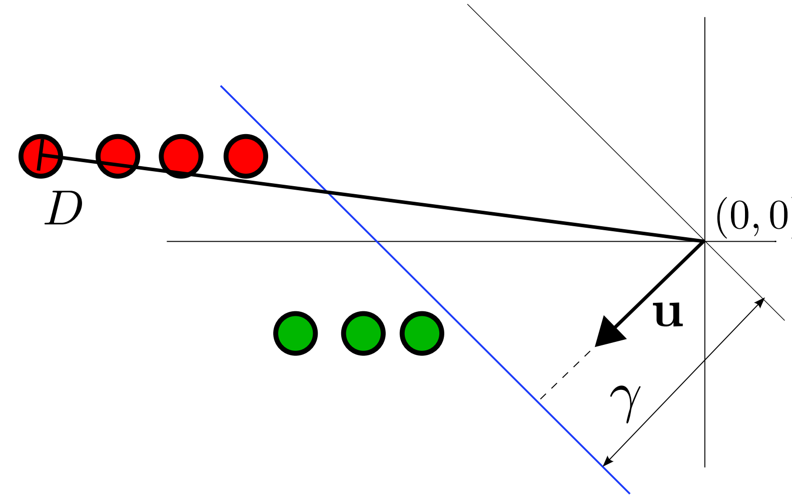
400 points visited.
All data classified correctly. Done.

Final weight:
$\mathbf{w} = (0, 2.76, 0.33)^\top$

Novikoff Theorem

Let the data be linearly separable and let there be a unit vector $\mathbf{u}$ and a scalar $\gamma \in \mathbb{R}^+$ such that

$$\mathbf{u} \cdot \mathbf{x}_j \geq \gamma \qquad \forall j \in \{1, 2, ..., L\} \qquad (\|\mathbf{u}\| = 1) \tag{7}$$



Let the norm of the longest vector in the dataset be D:

$$D = \max_{x \in \mathcal{T}} \|\mathbf{x}\| \,. \tag{8}$$

Then the perceptron algorithm will **finish in a finite number of steps** t^*, with

$$t^* \leq \frac{D^2}{\gamma^2} \,. \tag{9}$$

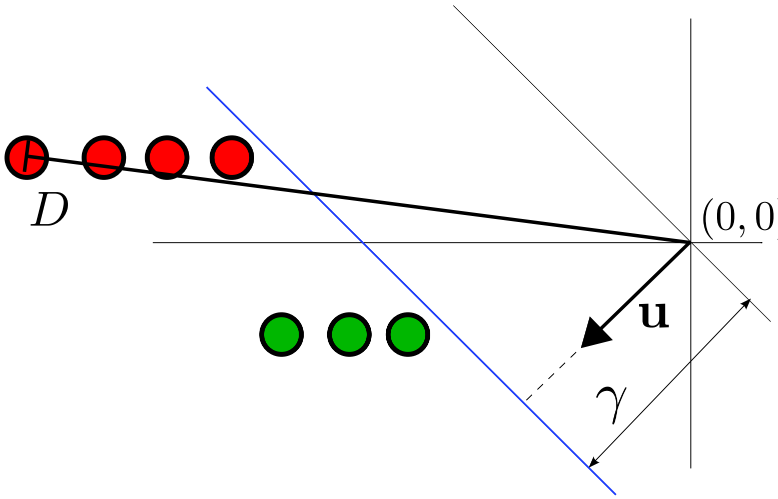
? What if the data is not separable?

? How to terminate perceptron learning?

Novikoff Theorem, Proof (1)

Let $\mathbf{x}^{(t)}$ be the point which is incorrectly classified at time t, so

$$\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)} \leq 0\,. \qquad (10)$$



Recall that the weight $\mathbf{w}^{(t+1)}$ is computed using this update $\mathbf{x}^{(t)}$ as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{x}^{(t)}\,. \qquad (11)$$

For the squared norm of $\mathbf{w}^{(t+1)}$, we have

$$\|\mathbf{w}^{(t+1)}\|^2 = \mathbf{w}^{(t+1)} \cdot \mathbf{w}^{(t+1)} = (\mathbf{w}^{(t)} + \mathbf{x}^{(t)}) \cdot (\mathbf{w}^{(t)} + \mathbf{x}^{(t)}) \qquad (12)$$

$$= \|\mathbf{w}^{(t)}\|^2 + 2\,\underbrace{\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}}_{\leq 0} + \underbrace{\|\mathbf{x}^{(t)}\|^2}_{\leq D^2} \qquad (13)$$

$$\leq \|\mathbf{w}^{(t)}\|^2 + D^2 \quad \leq \quad \|\mathbf{w}^{(t-1)}\|^2 + 2D^2 \qquad (14)$$

$$\leq \|\mathbf{w}^{(t-2)}\|^2 + 3D^2 \quad \leq \quad \ldots \quad \leq \quad \|\mathbf{w}^{(0)}\|^2 + (t+1)D^2 \qquad (15)$$

$$\|\mathbf{w}^{(t+1)}\|^2 \quad \leq (t+1)D^2 \qquad (16)$$

Novikoff Theorem, Proof (2)

We also have that

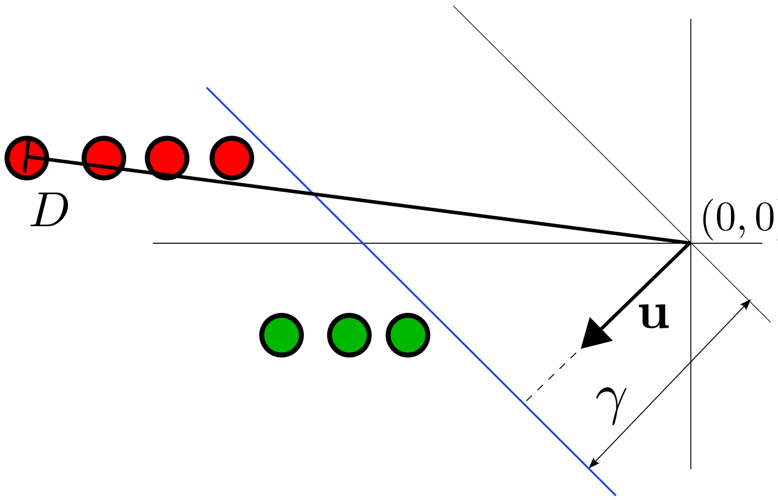
$$\mathbf{w}^{(t+1)} \cdot \mathbf{u} = \mathbf{w}^{(t)} \cdot \mathbf{u} + \underbrace{\mathbf{x}^{(t)} \cdot \mathbf{u}}_{\geq \gamma} \tag{17}$$

$$\geq \mathbf{w}^{(t)} \cdot \mathbf{u} + \gamma \geq \mathbf{w}^{(t-1)} \cdot \mathbf{u} + 2\gamma \tag{18}$$

$$\geq \mathbf{w}^{(t-2)} \cdot \mathbf{u} + 3\gamma \geq \dots \tag{19}$$

$$\geq \mathbf{w}^{(0)} \cdot \mathbf{u} + (t+1)\gamma \tag{20}$$

$$\mathbf{w}^{(t+1)} \cdot \mathbf{u} \quad \geq (t+1)\gamma \tag{21}$$



We take the two inequalities together, to obtain

$$(t+1)D^2 \geq \|\mathbf{w}^{(t+1)}\|^2 \geq (\mathbf{w}^{(t+1)} \cdot \mathbf{u})^2 \geq (t+1)^2\gamma^2 \qquad (\|\mathbf{u}\| = 1) \tag{22}$$

Therefore,

$$(t+1) \leq \frac{D^2}{\gamma^2}\,. \tag{23}$$

Perceptron Learning as an Optimisation Problem (1)

Perceptron algorithm, batch version, handling non-separability, another perspective:

Input: $\mathcal{T} = \{\mathbf{x}_1, \dots \mathbf{x}_L\}$
Output: a weight vector $\mathbf{w}$ minimising

$$J(\mathbf{w}) = |\{\mathbf{x} \in X : \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0\}| \tag{24}$$

or, equivalently

$$J(\mathbf{w}) = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0}} 1 \tag{25}$$

What would the most common optimisation method, i.e. gradient descent, perform?

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla J(\mathbf{w}) \tag{26}$$

The gradient of $J(\mathbf{w})$ is, however, either 0 or undefined. The gradient minimisation cannot proceed.

Perceptron Learning as an Optimisation Problem (2)

Let us redefine the cost function:

$$J_p(\mathbf{w}) = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w} \cdot \mathbf{x} \leq 0}} (-\mathbf{w} \cdot \mathbf{x}) \tag{27}$$

$$\nabla J_p(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}} = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w} \cdot \mathbf{x} \leq 0}} (-\mathbf{x}) \tag{28}$$

- The Perceptron Algorithm is a gradient **descent** method for $J_p(\mathbf{w})$ (gradient for a single misclassified sample is $-\mathbf{x}$, so the weight update is $\mathbf{x}$)
- Learning and empirical risk minimisation is just an instance of an optimization problem.
- Either gradient minimisation (backpropagation in neural networks) or convex (quadratic) minimisation (in mathematical literature called convex programming) is used.

Perceptron Learning: Non-Separable Case

Perceptron algorithm, batch version, handling non-separability:

Input: $\mathcal{T} = \{\mathbf{x}_1, \ldots \mathbf{x}_L\}$
Output: weight vector $\mathbf{w}^*$

1. $\mathbf{w}^{(0)} = 0$, $E = |T| = L$, $\mathbf{w}^* = 0$.
2. Find all mis-classified observations $X^- = \{\mathbf{x} \in X : \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0\}$.
3. if $|X^-| < E$ then $E = |X^-|; \mathbf{w}^* = \mathbf{w}^{(t)}$
4. if TermCond$(\mathbf{w}^*, t, t_{\text{lup}})$ then terminatate (t_{lup} is the time of the last update) else:
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t \sum_{x \in X^-} x$$
5. Goto 2.

- The algorithm converges with probability 1 to the optimal solution.
- Convergence rate is not known.
- Termination condition TermCond$(\cdot)$ is a complex function of the quality of the best solution, time since last update $t - t_{\text{lup}}$ and requirements on the solution.

Optimal Separating Plane and The Closest Point To The Convex Hull

The problem of optimal separation by a hyperplane

$$(1) \qquad \mathbf{w}^* = \underset{w}{\operatorname{argmax}} \min_j \frac{\mathbf{w}}{|\mathbf{w}|} \cdot \mathbf{x}_j \tag{29}$$

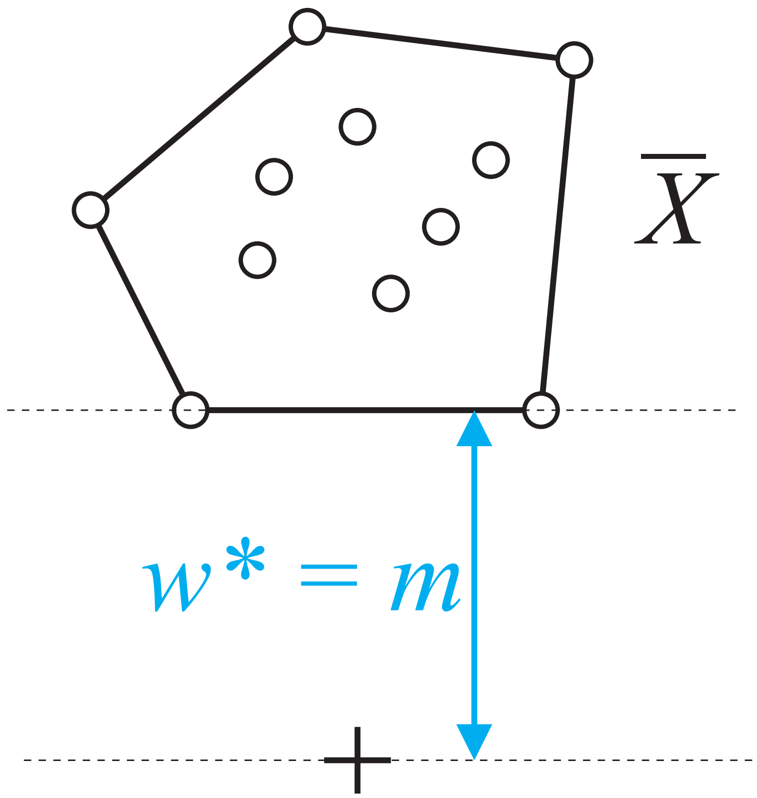
can be converted to searching for the closest point to a convex hull (denoted by the overline)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \overline{X}}{\operatorname{argmin}} \ |\mathbf{x}|$$

There holds that $\mathbf{x}^*$ solves also the problem (29).

Recall that the classfier that maximises separation minimises the structural risk R_{str} (page 8)

Convex Hull, Illustration



$$\min_j \left(\frac{\mathbf{w}}{|\mathbf{w}|} \cdot \mathbf{x}_j \right) \leq \quad m \quad \leq |\mathbf{w}| \,, \; \mathbf{w} \in \overline{X}$$

lower bound upper bound

ε-Solution

- The aim is to speed up the algorithm.
- The allowed uncertainty ε is introduced.

$$|\mathbf{w}| - \min_j \left(\frac{\mathbf{w}}{|\mathbf{w}|} \cdot \mathbf{x}_j \right) \leq \varepsilon$$

Training Algorithm 2 – Kozinec (1973)

1. $\mathbf{w}^{(0)} = \mathbf{x}_j$, i.e. any observation.
2. A wrongly classified observation $\mathbf{x}_j$ is sought, i.e., $\mathbf{w}^{(t)} \cdot \mathbf{x}_j \le 0$, $j \in J$.
3. If there is no wrongly classified observation then the algorithm finishes otherwise

$$\mathbf{w}^{(t+1)} = (1 - \kappa^*)\, \mathbf{w}^{(t)} + \kappa^*\, \mathbf{x}_j ,$$

$$\kappa^* = \operatorname*{argmin}_{\kappa \in (0,1)} \|(1 - \kappa)\, \mathbf{w}^{(t)} + \kappa\, \mathbf{x}_j\|$$

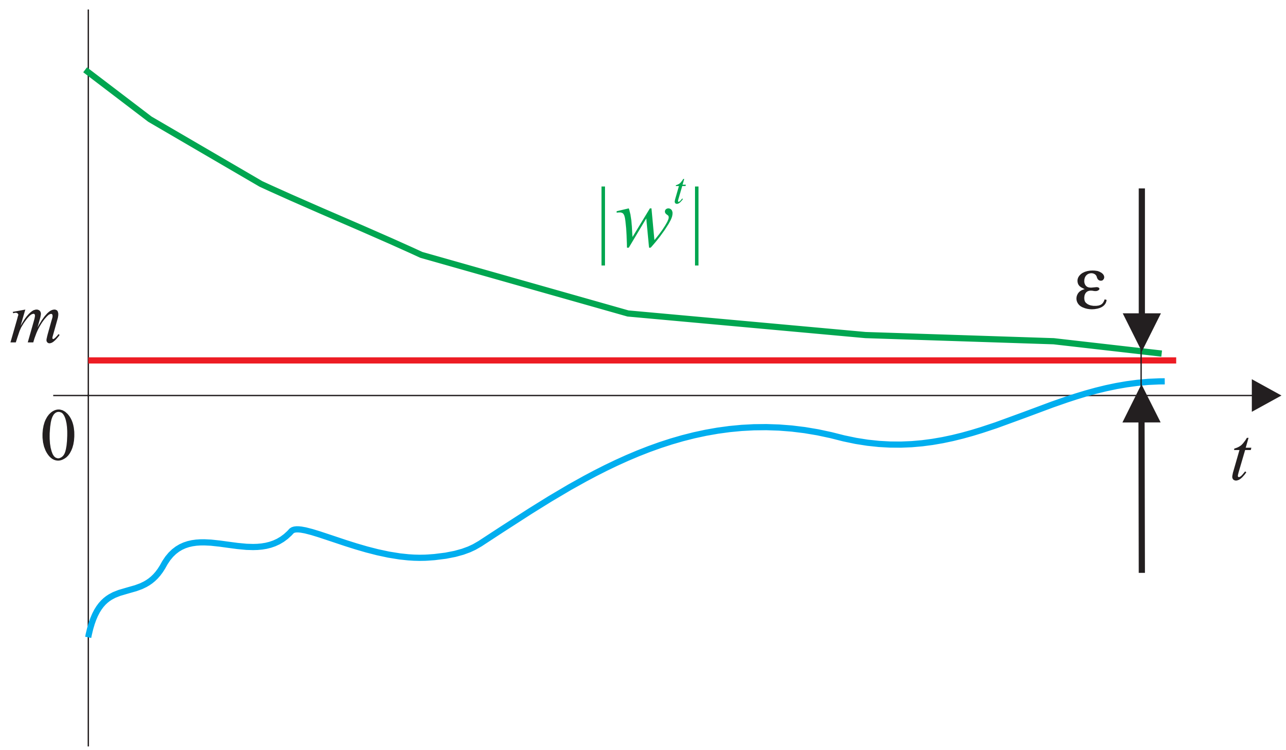
4. Goto 2.

Kozinec and ε-Solution

The second step of Kozinec algorithm is modified to:

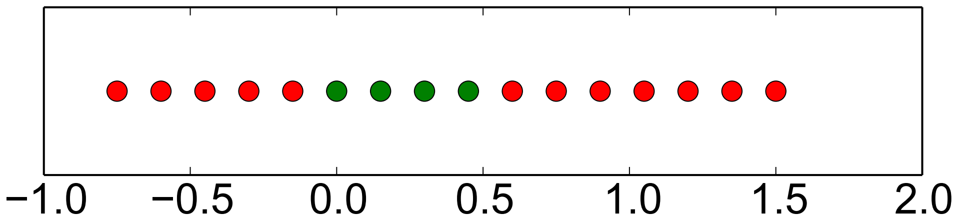
A wrongly classified observation $\mathbf{x}_j$ is sought for which

$$|\mathbf{w}^{(t)}| - \min_j \left(\frac{\mathbf{w}^{(t)}}{|\mathbf{w}^{(t)}|} \cdot \mathbf{x}_j \right) \geq \varepsilon$$



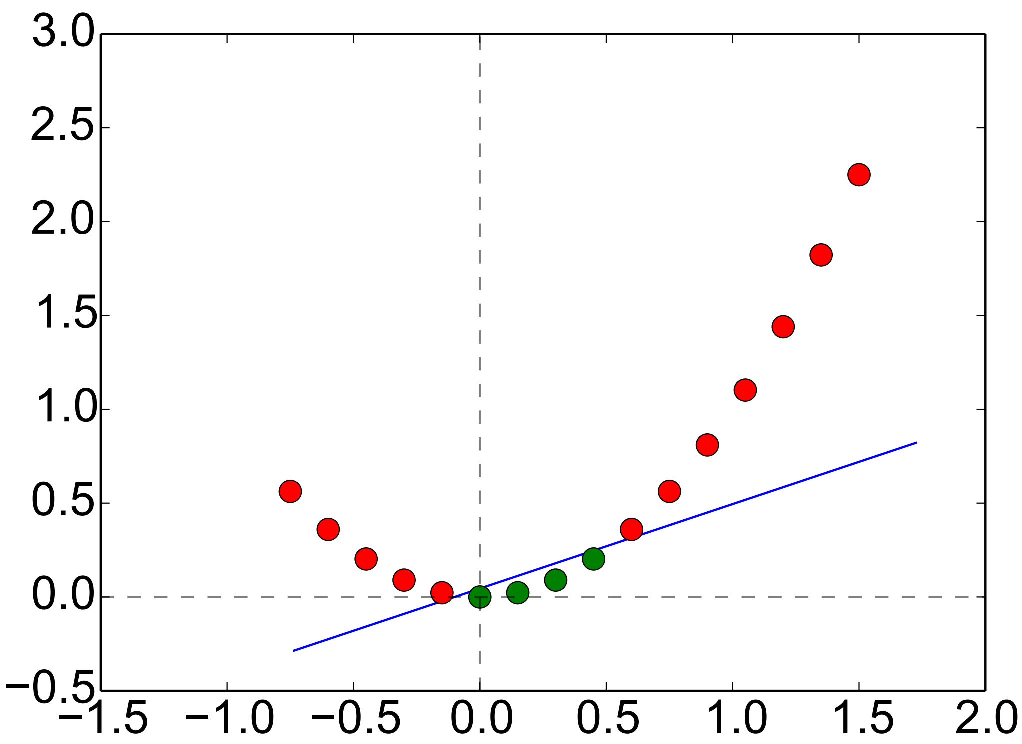
Dimension Lifting

Consider the data on the right. They are not linearly separable, because there is no $\mathbf{w} \in \mathbb{R}^2$ such that $\mathrm{sign}(w_0 + w_1 x)$ would correctly classify the data.



Let us artificially enlarge the dimensionality of the feature space by a mapping $\phi(x) : \mathbb{R} \to \mathbb{R}^2$:

$$\mathbf{x} \leftarrow \phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix} \qquad (30)$$



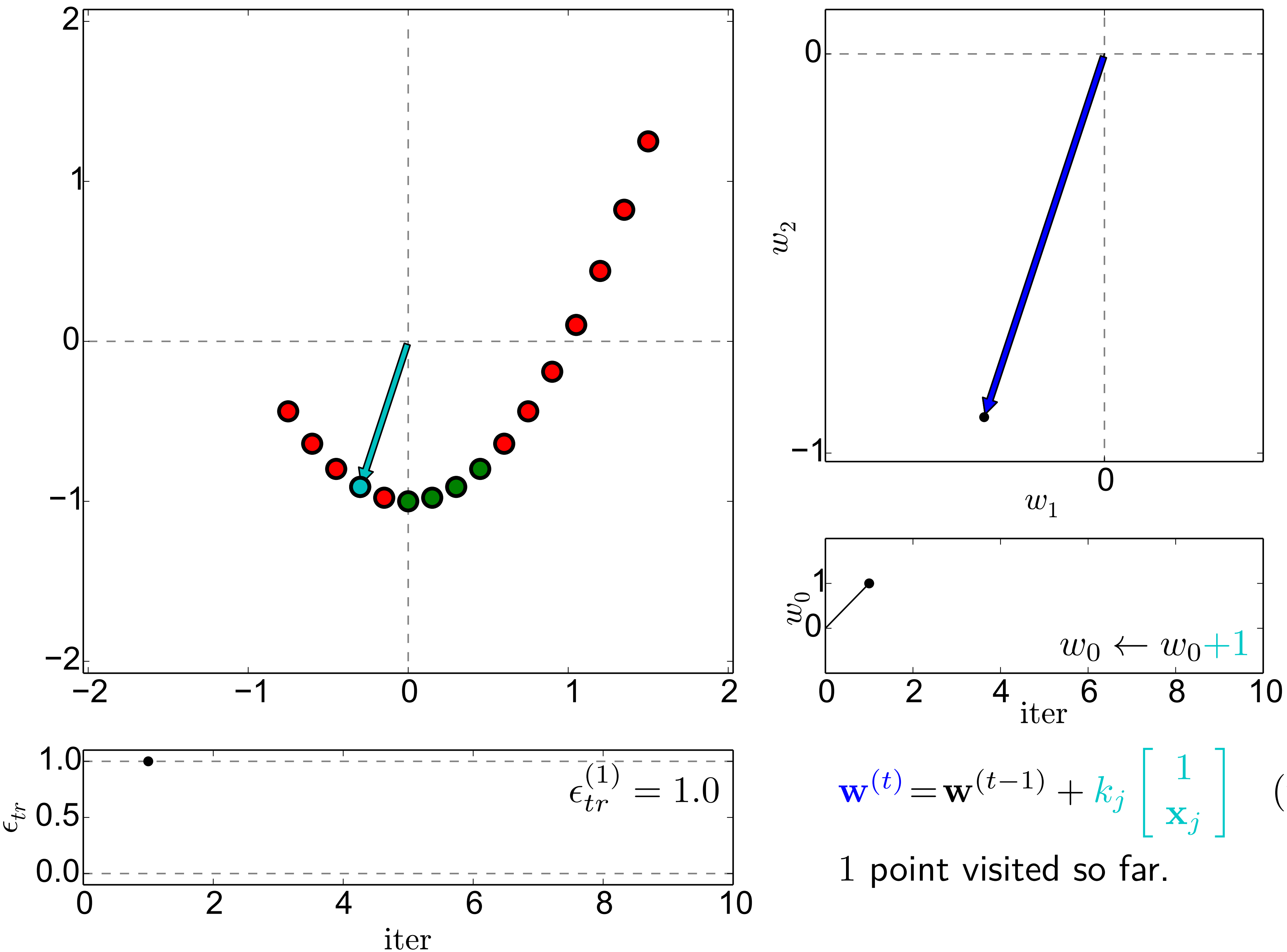
After such mapping, the data become linearly separable (the separator is shown on the right).

In general, lifting the feature space means adding D' dimensions and replacing the original feature vectors by

$$\mathbf{x} \leftarrow \phi(\mathbf{x}), \quad \phi(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^{D+D'}. \qquad (31)$$

Lifting, Example 1, Iter. 1

class 1, class -1, ●/●=misclassified, ● the weight updater

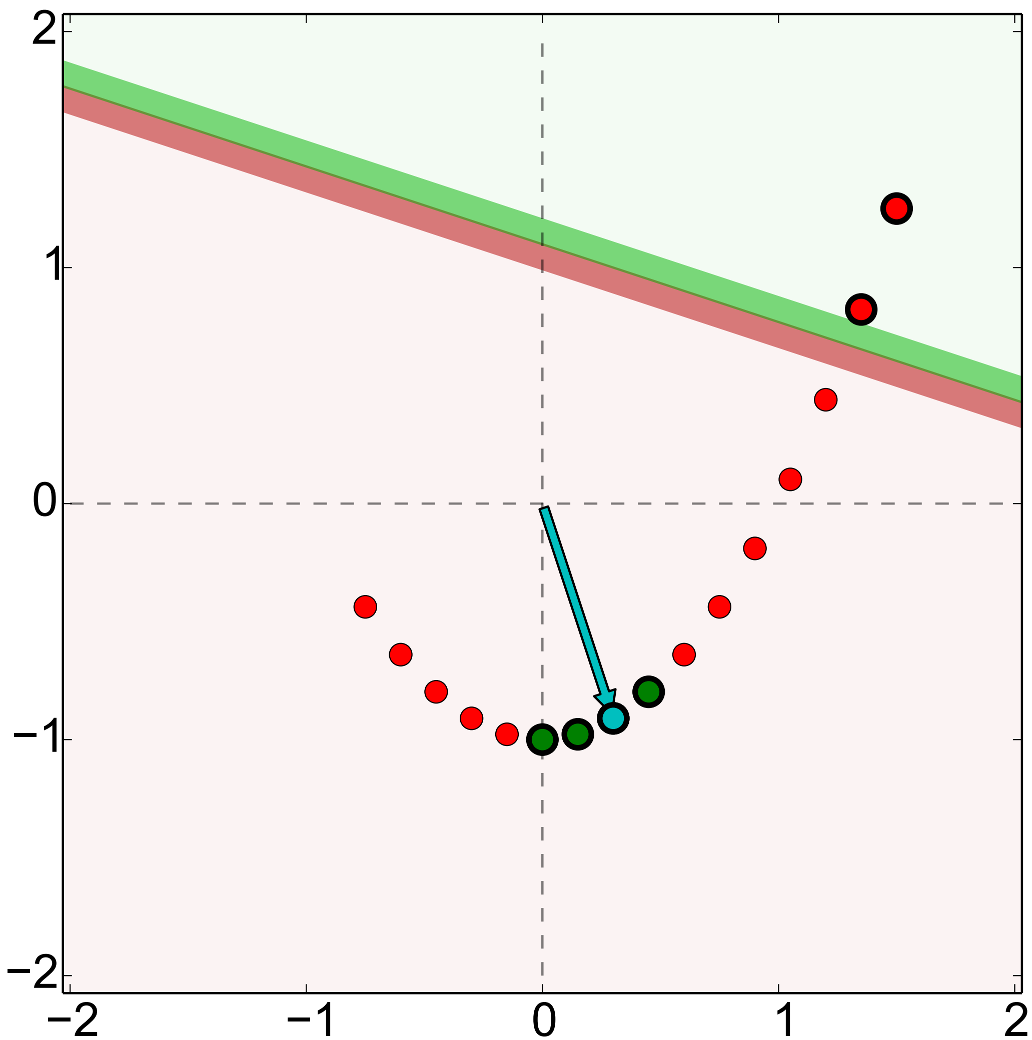


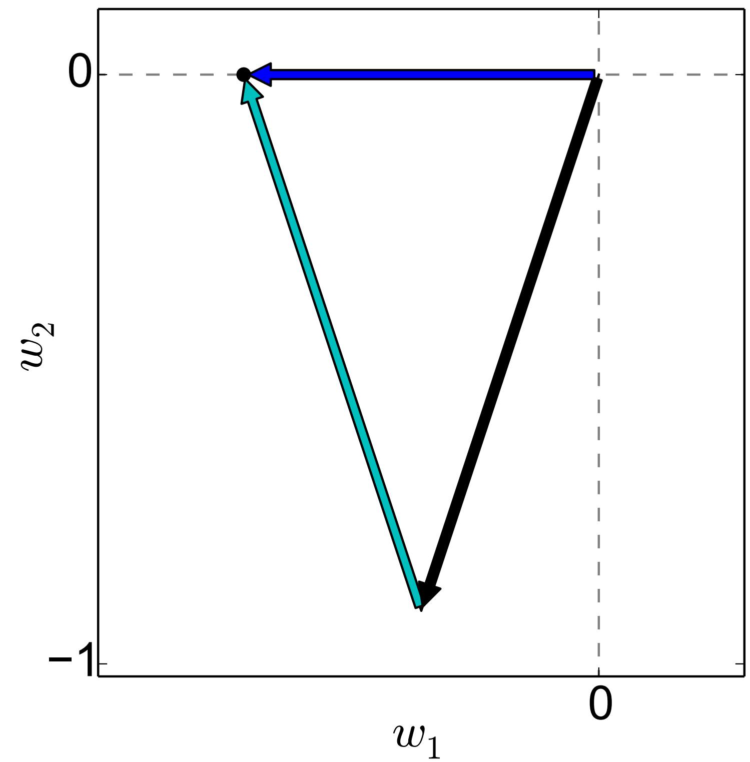
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 1)$$

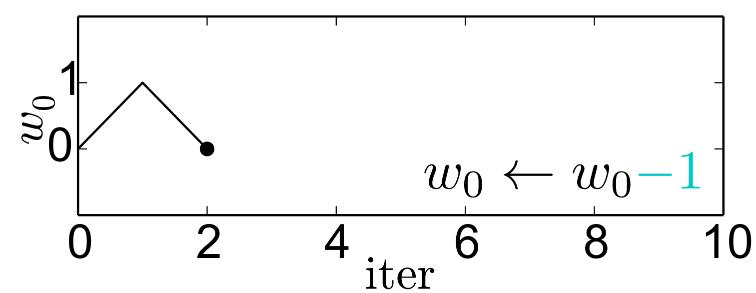
1 point visited so far.

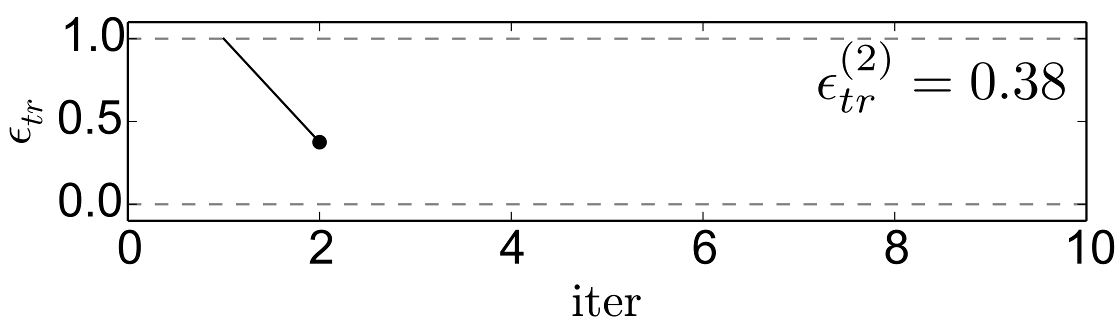
Lifting, Example 1, Iter. 2

● class 1, ● class -1, ●/●=misclassified, ● the weight updater







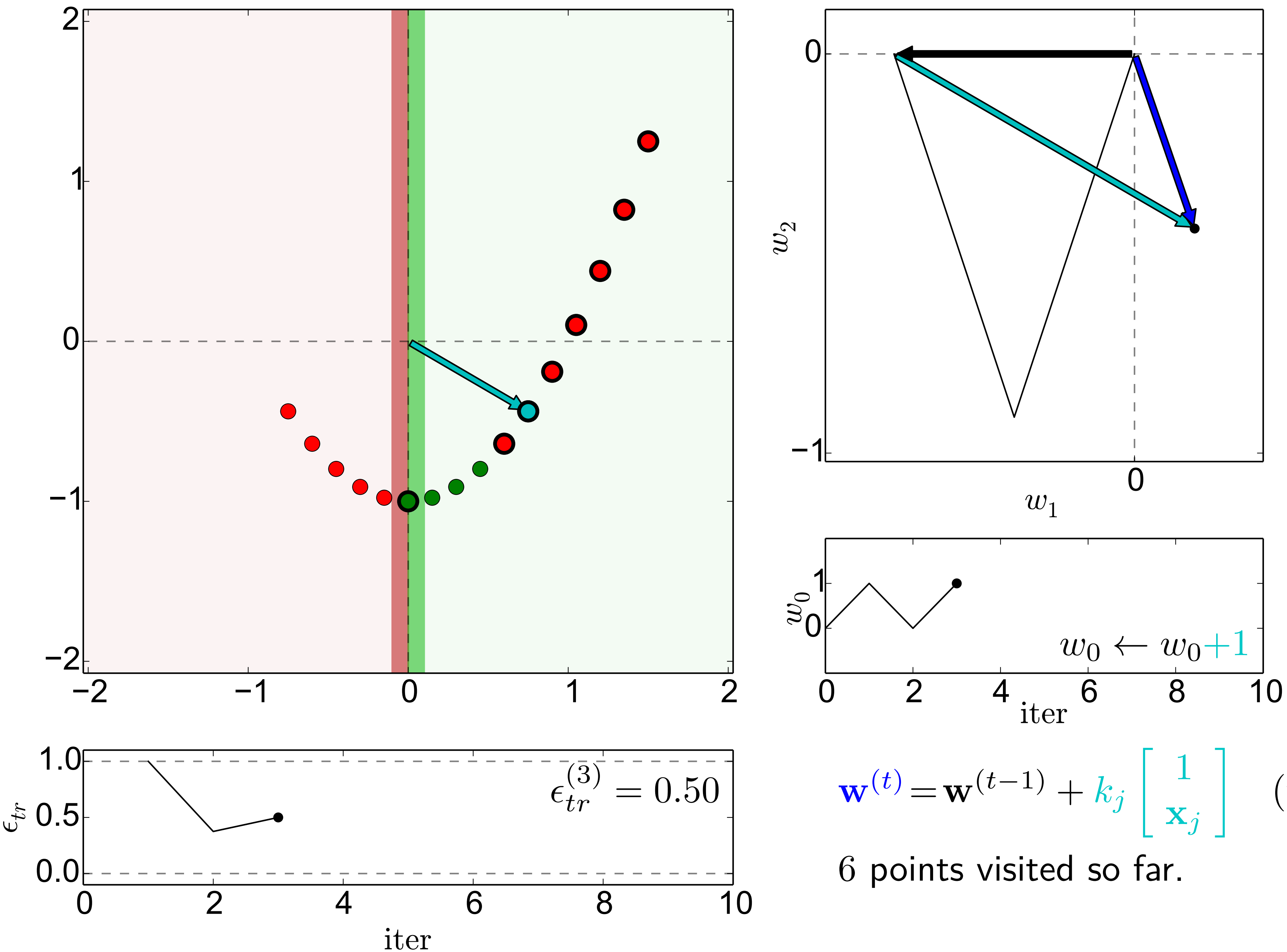


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 2)$$

3 points visited so far.

Lifting, Example 1, Iter. 3

class 1, class -1, ●/●=misclassified, ● the weight updater

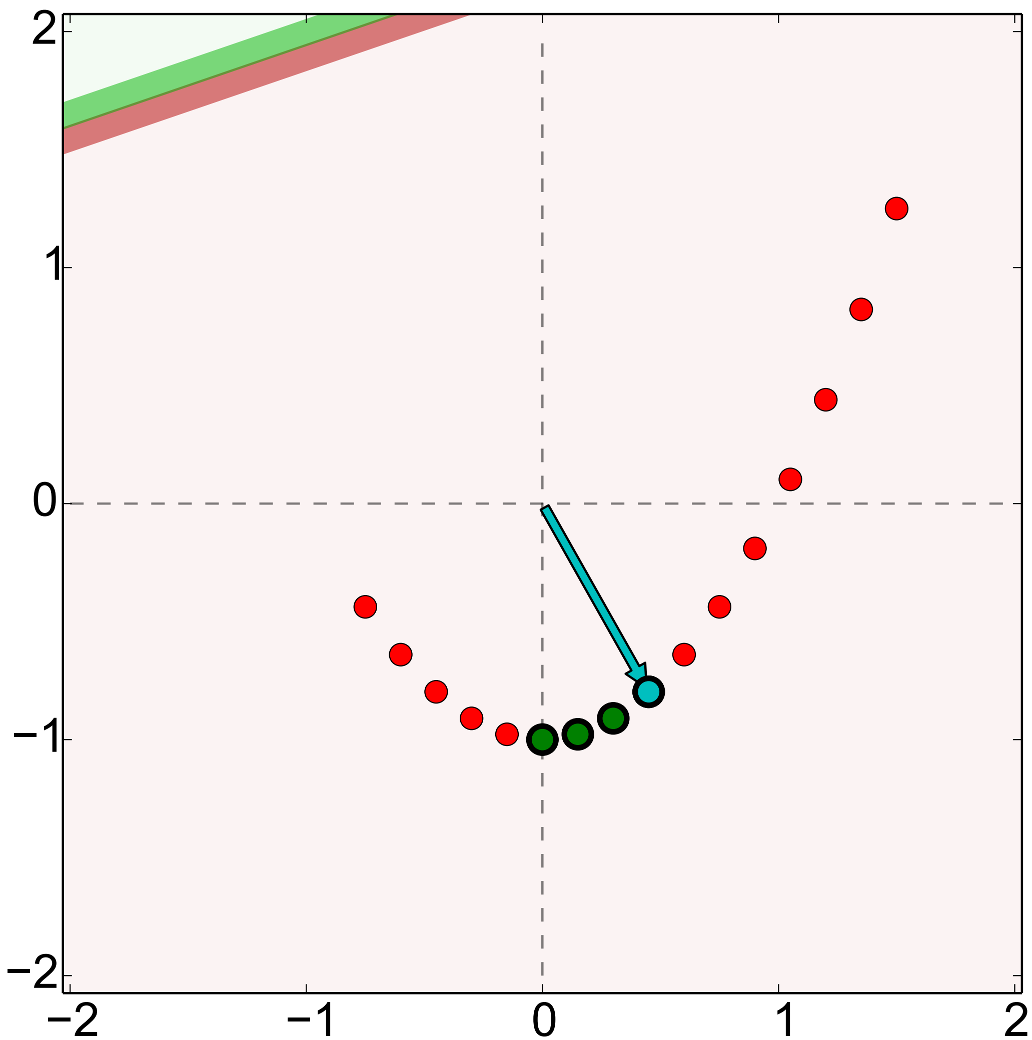


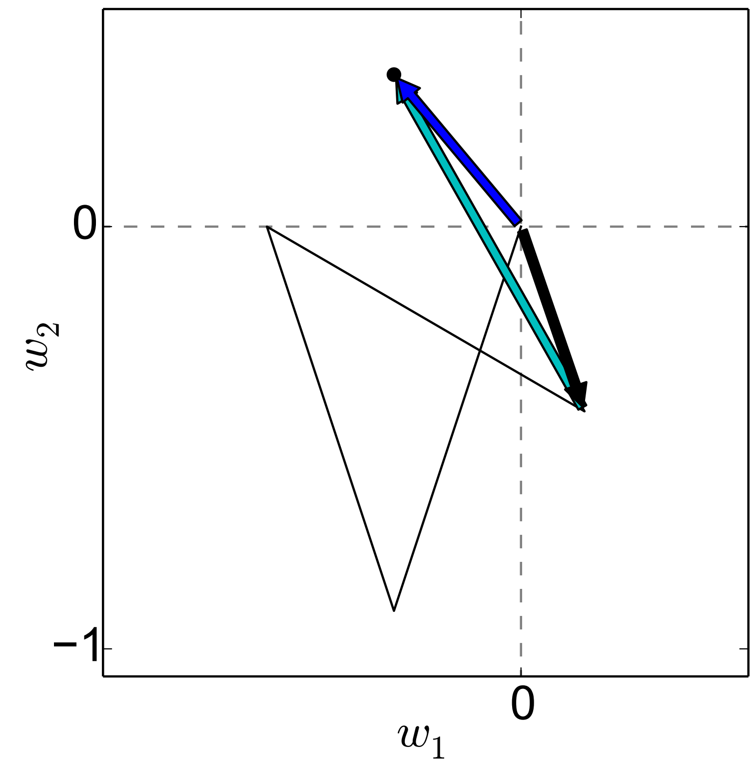
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 3)$$

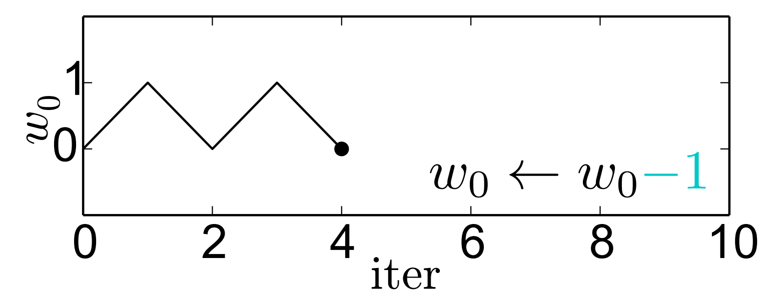
6 points visited so far.

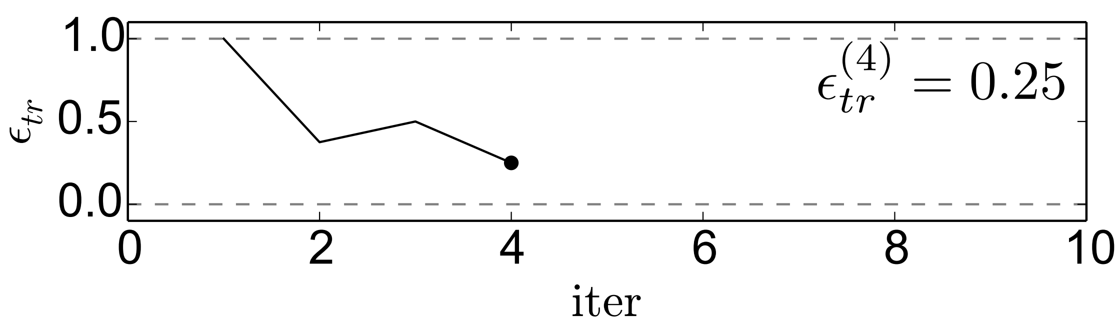
Lifting, Example 1, Iter. 4

class 1, class -1, ●/●=misclassified, ● the weight updater







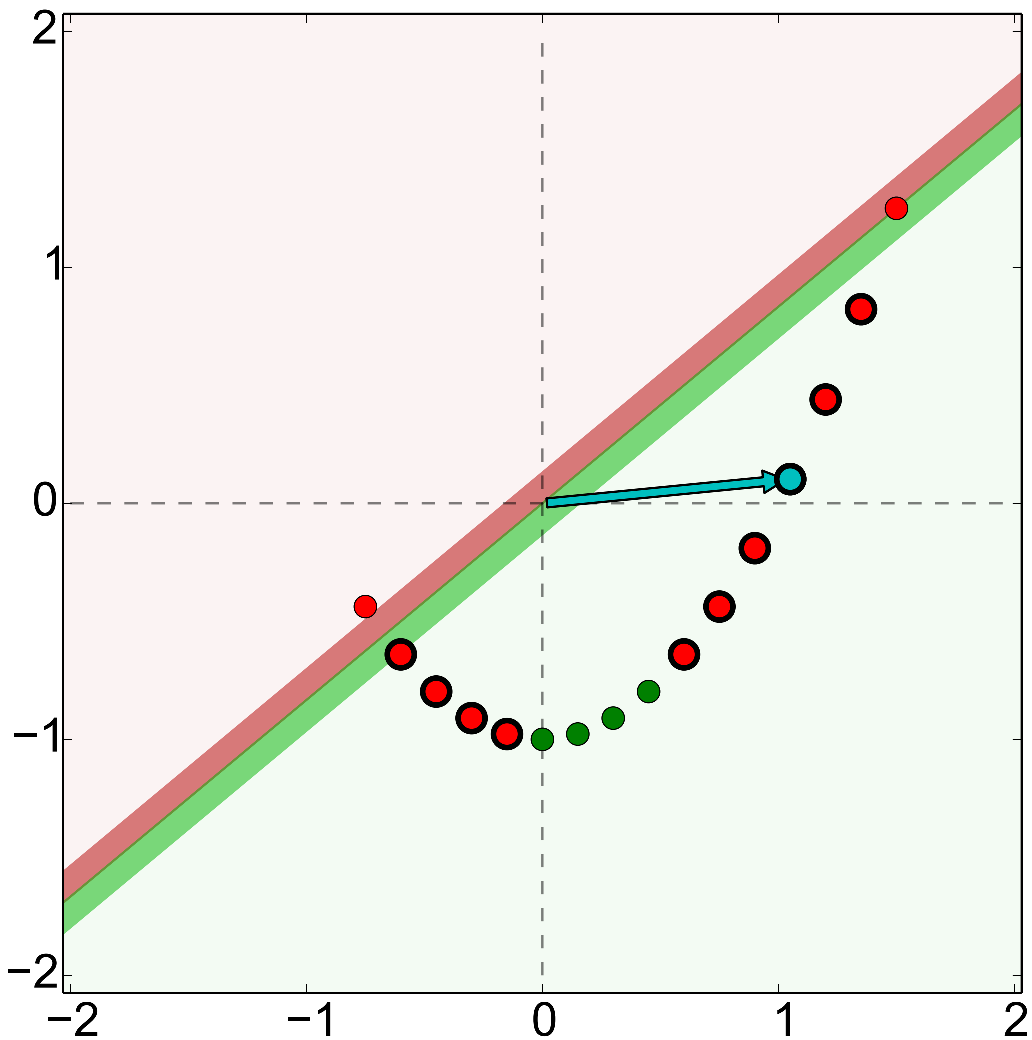


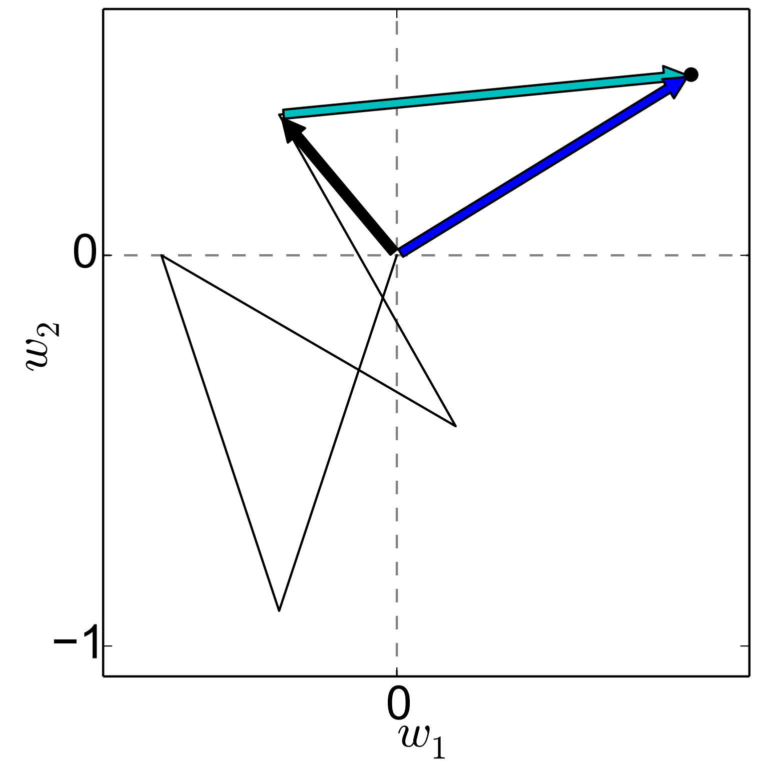
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 4)$$

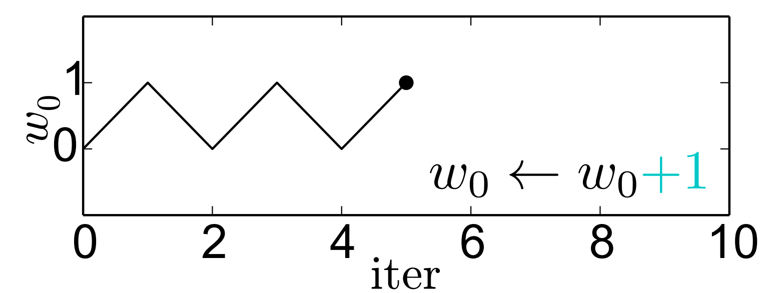
13 points visited so far.

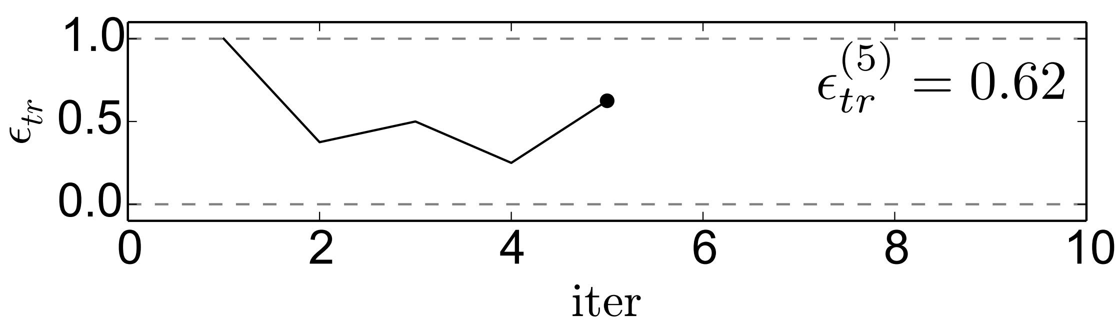
Lifting, Example 1, Iter. 5

class 1, class -1, ⬤/⬤=misclassified, ⬤ the weight updater







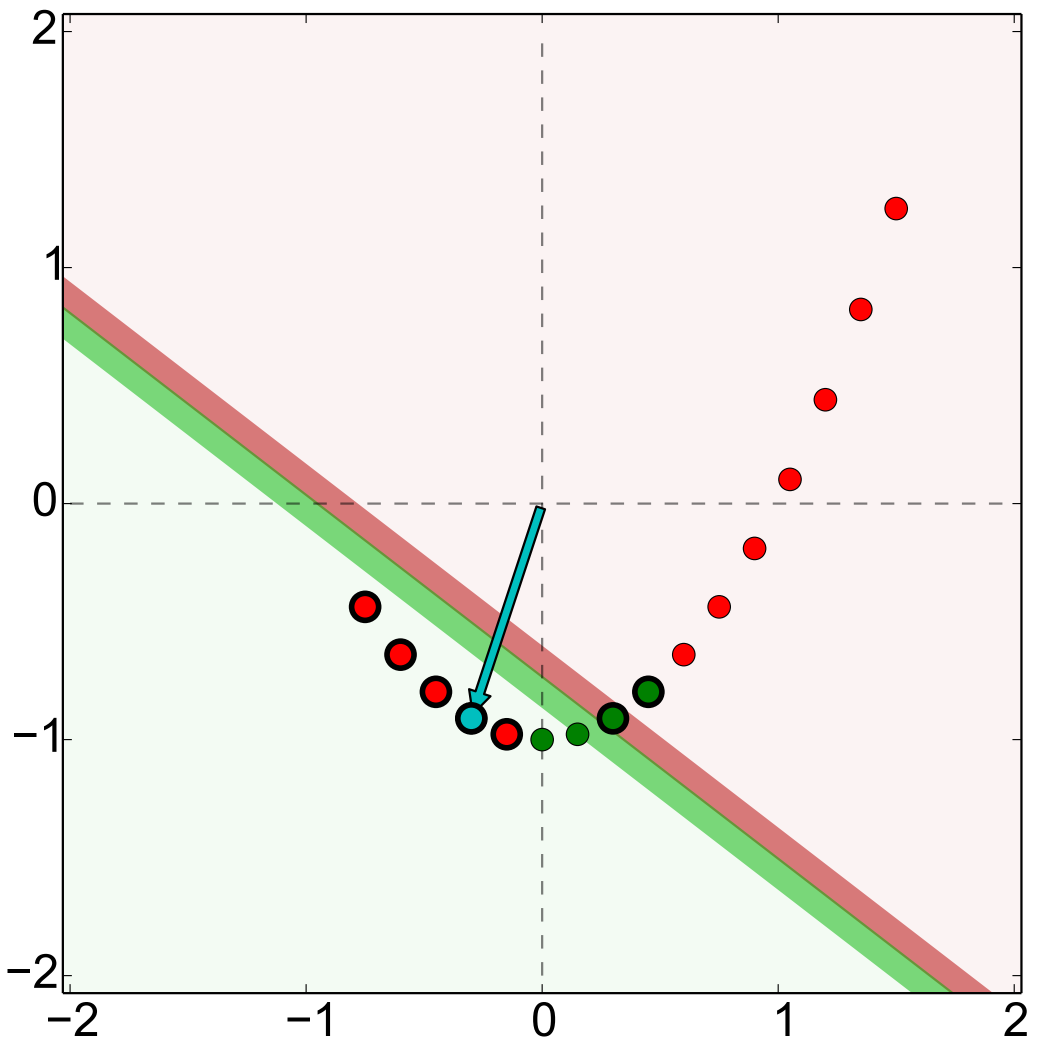


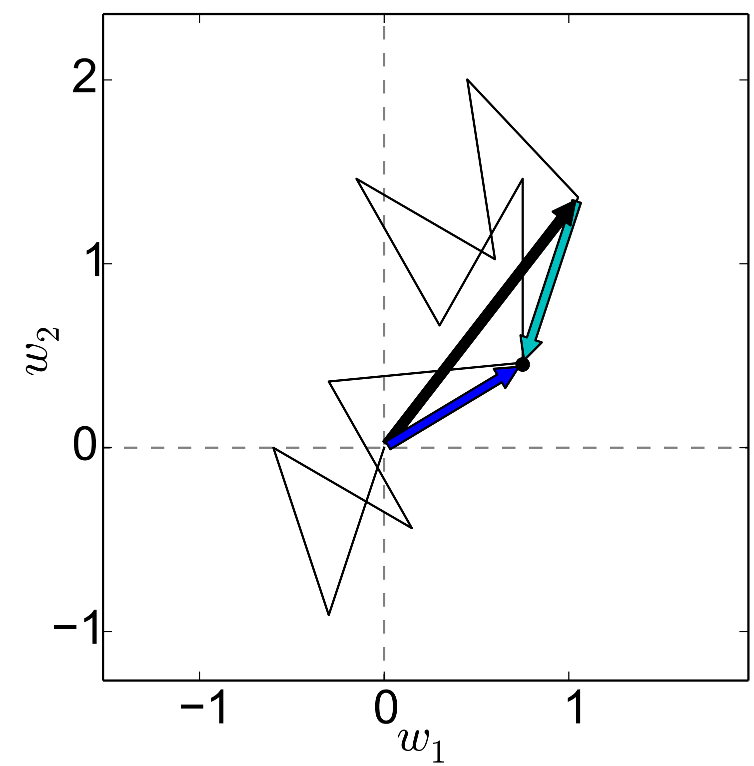
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 5)$$

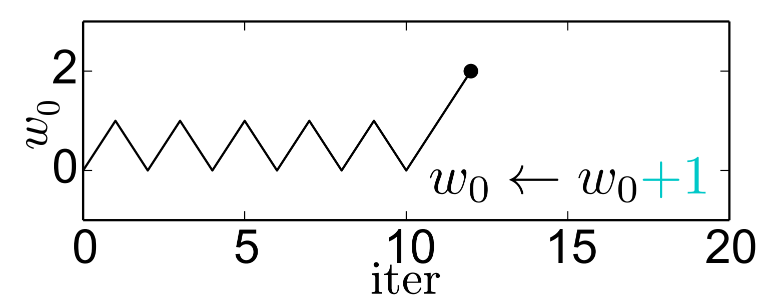
14 points visited so far.

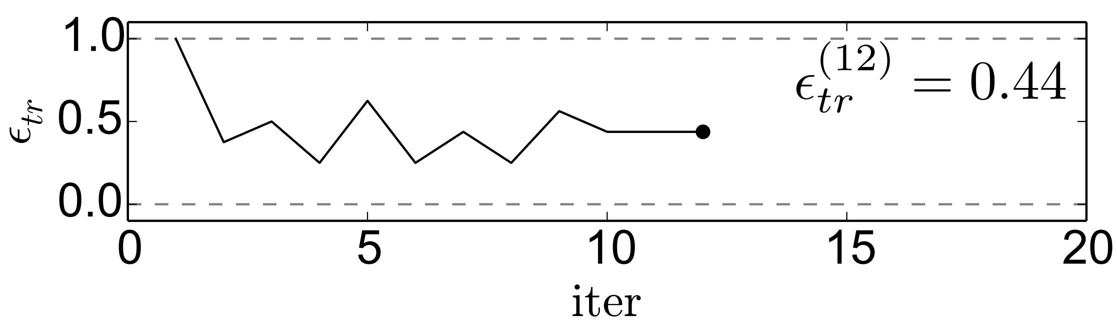
Lifting, Example 1, Iter. 12

● class 1, ● class -1, ●/●=misclassified, ● the weight updater







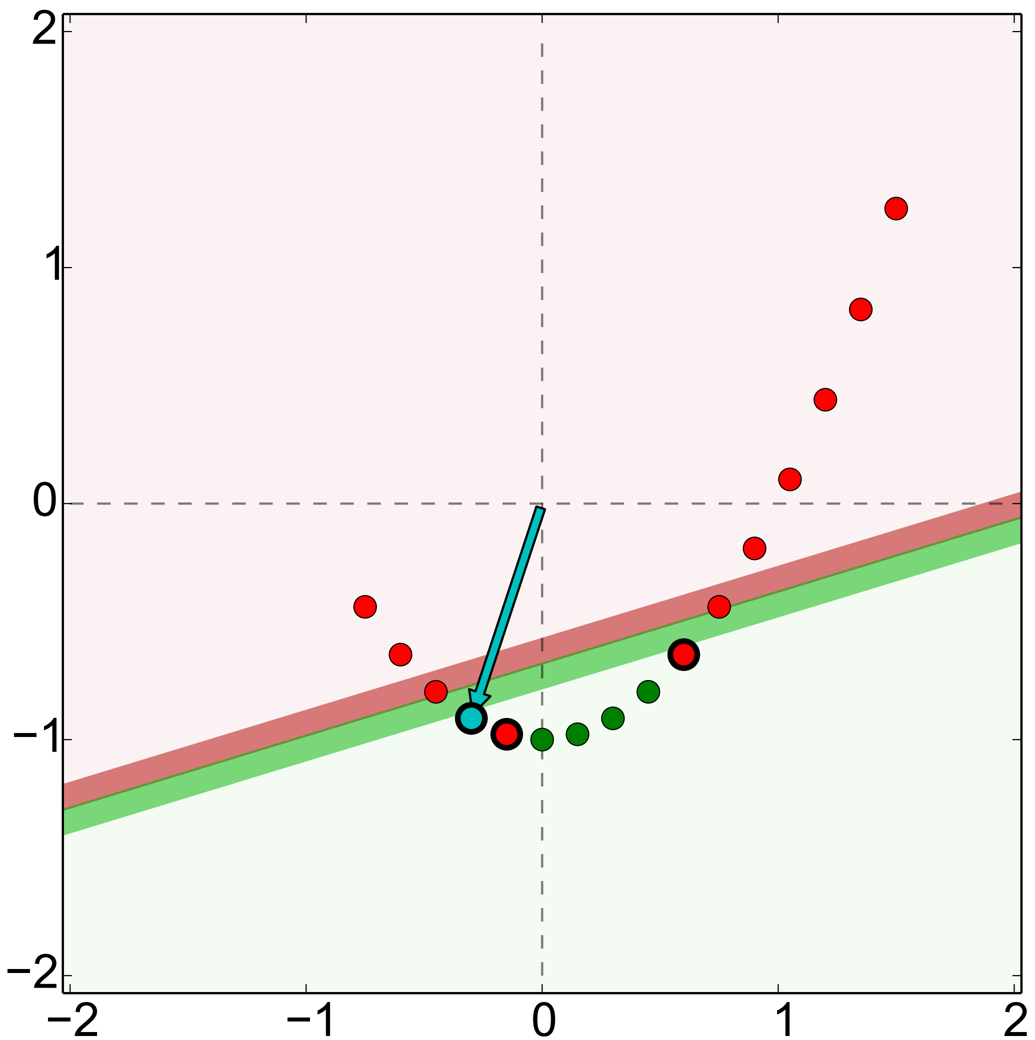


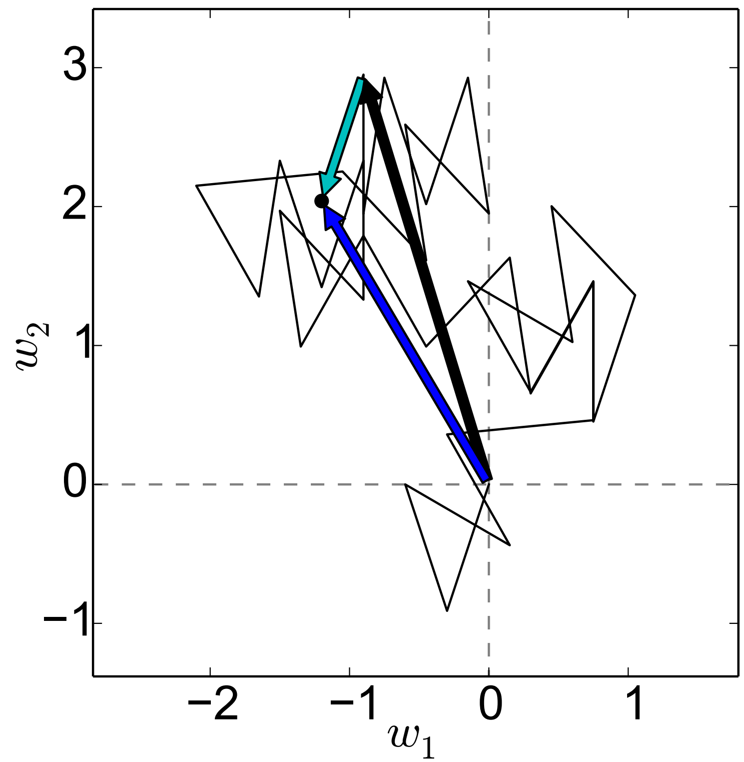
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 12)$$

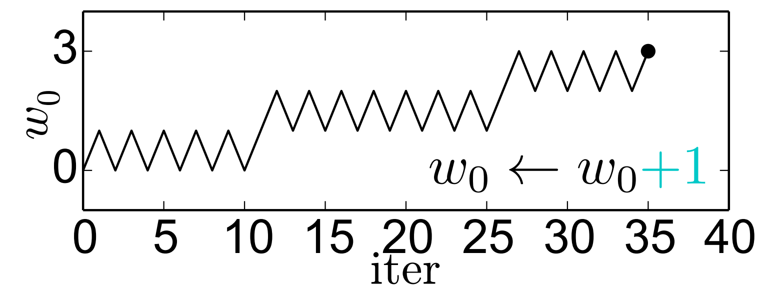
36 points visited so far.

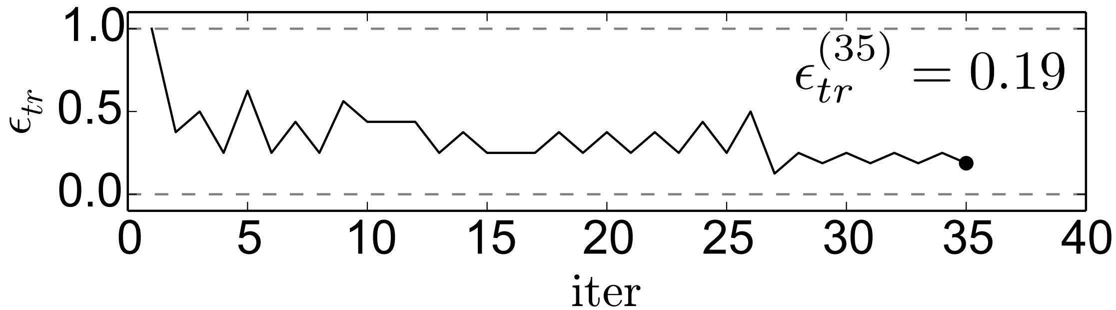
Lifting, Example 1, Iter. 35

● class 1, ● class -1, ●/●=misclassified, ● the weight updater







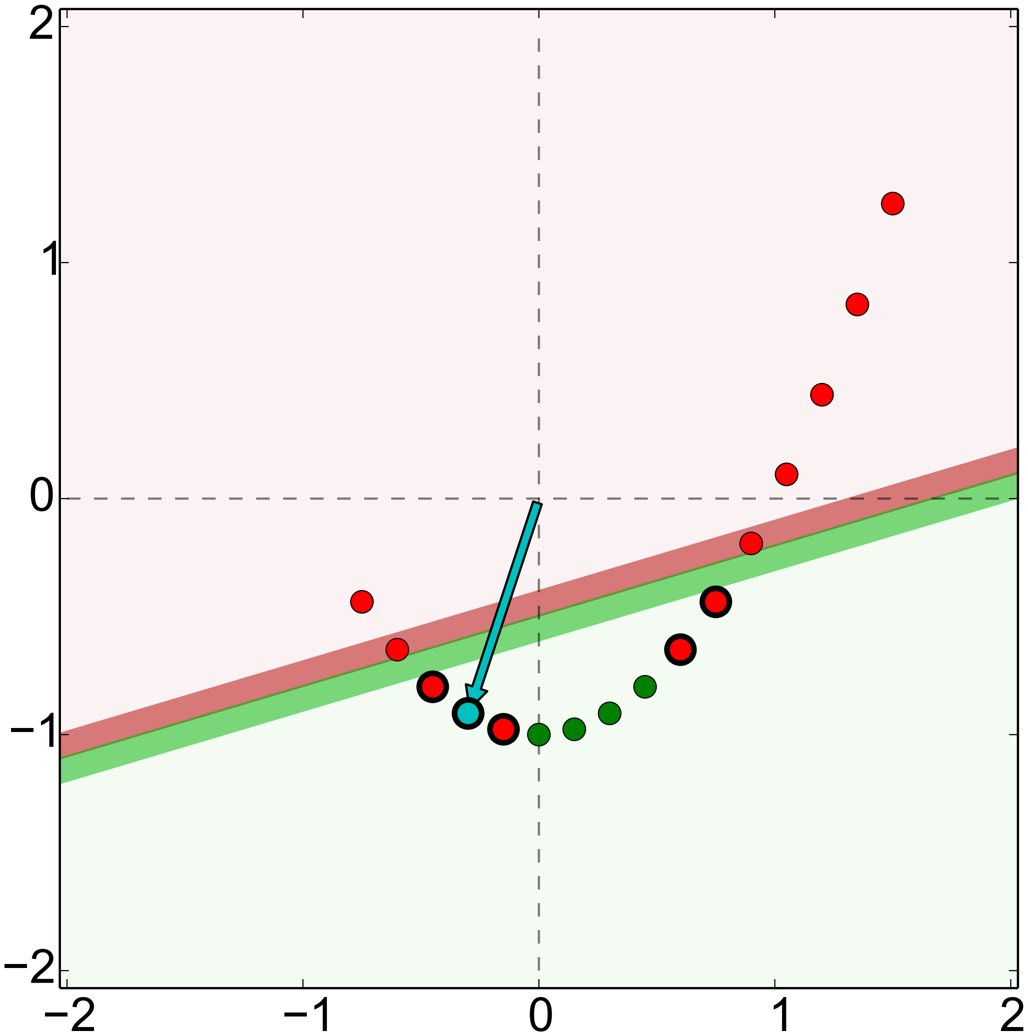


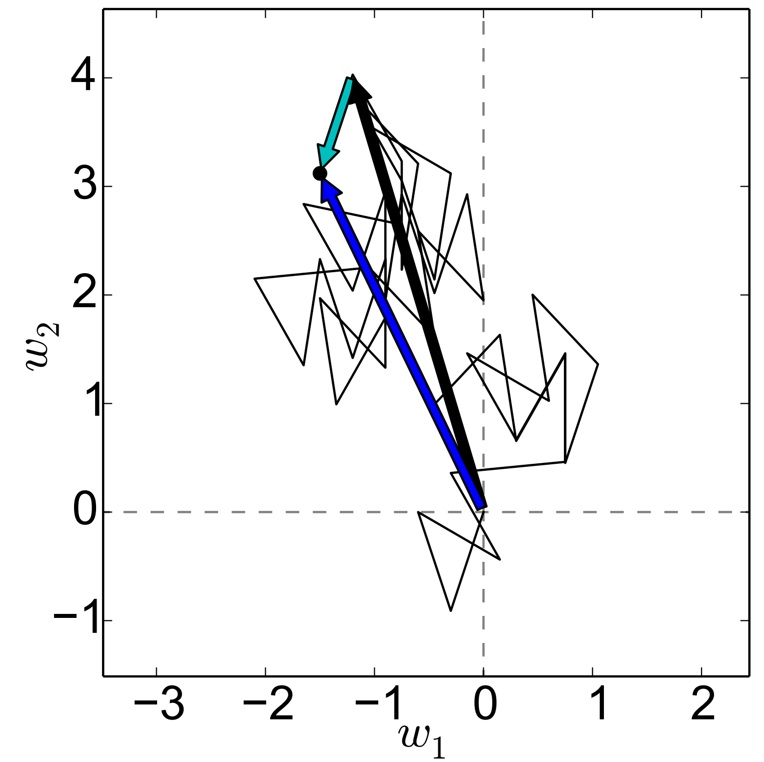
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 35)$$

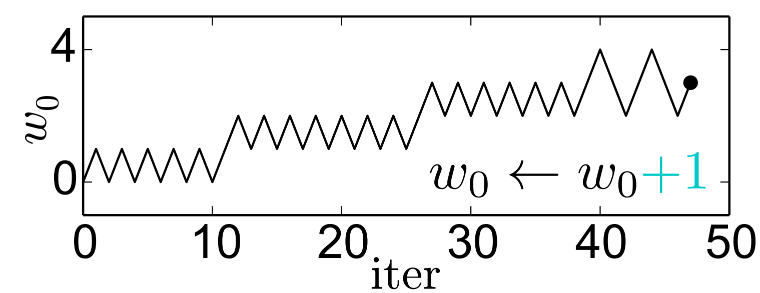
97 points visited so far.

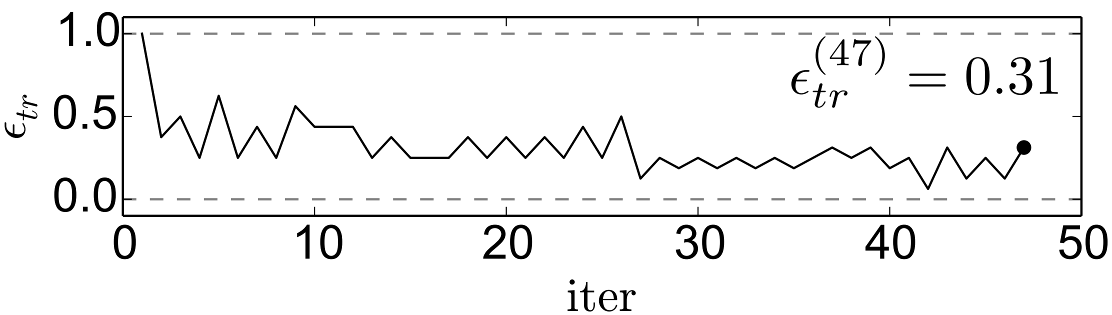
Lifting, Example 1, Iter. 47

● class 1, ● class -1, ●/●=misclassified, ● the weight updater







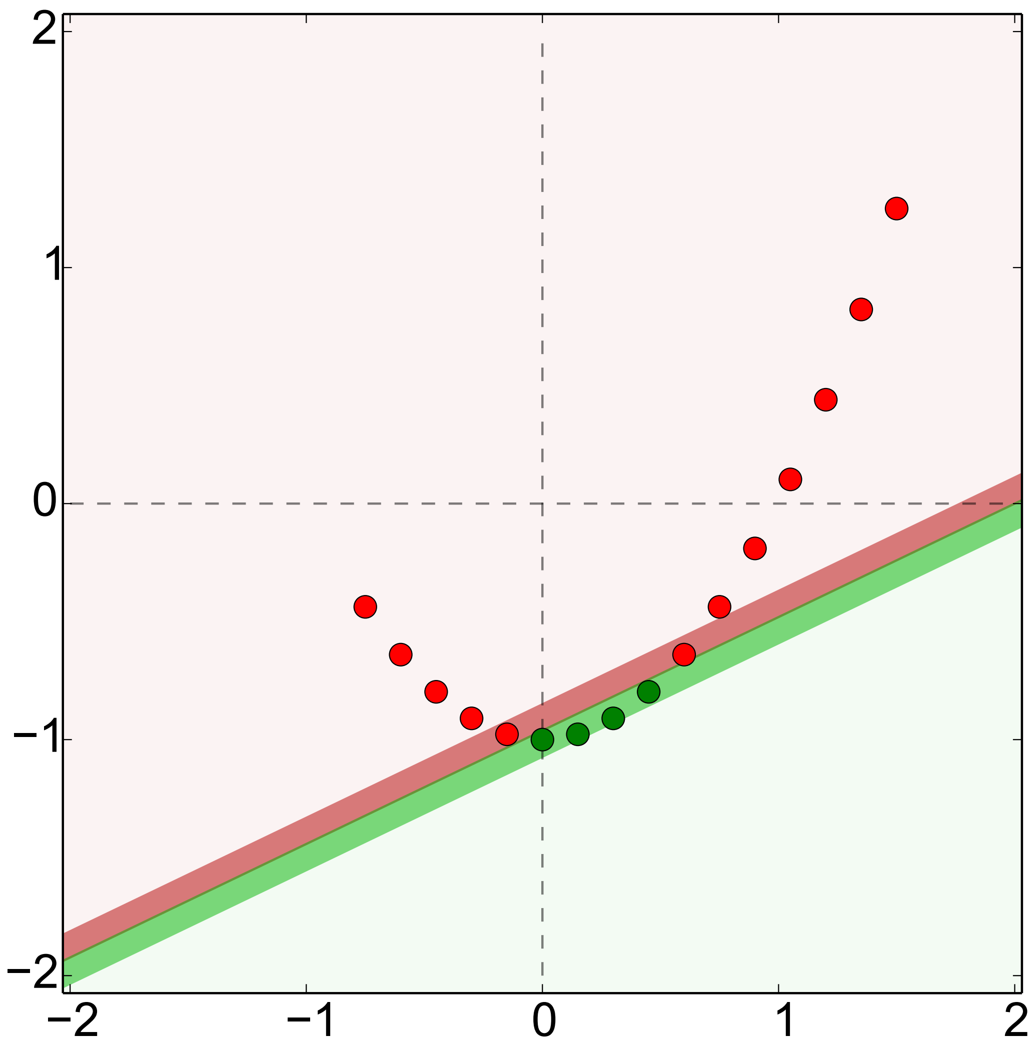


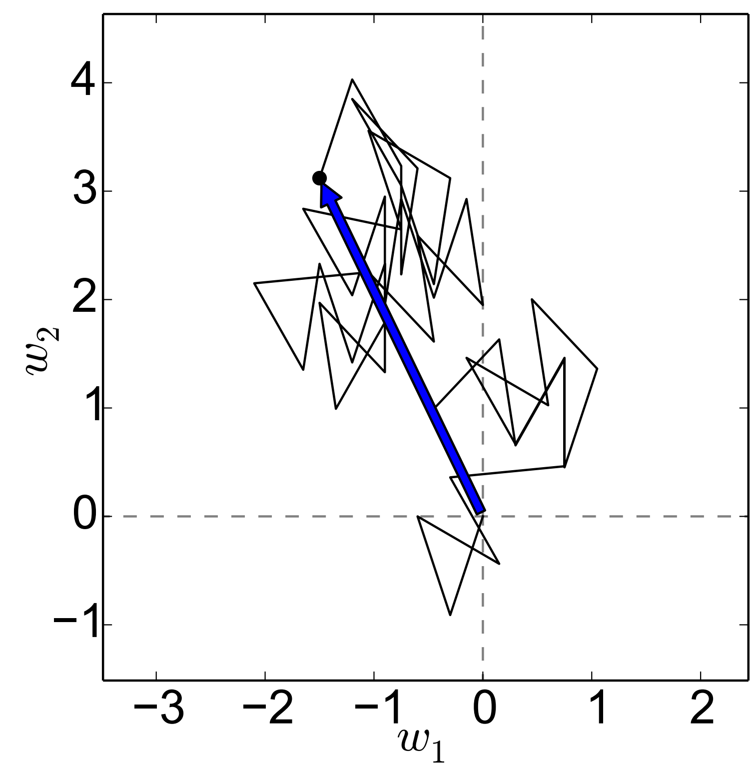
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 47)$$

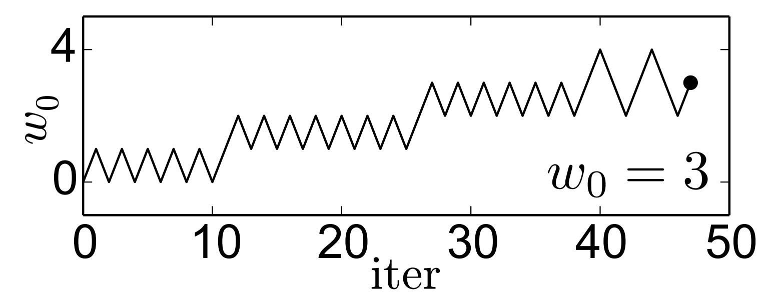
139 points visited so far.

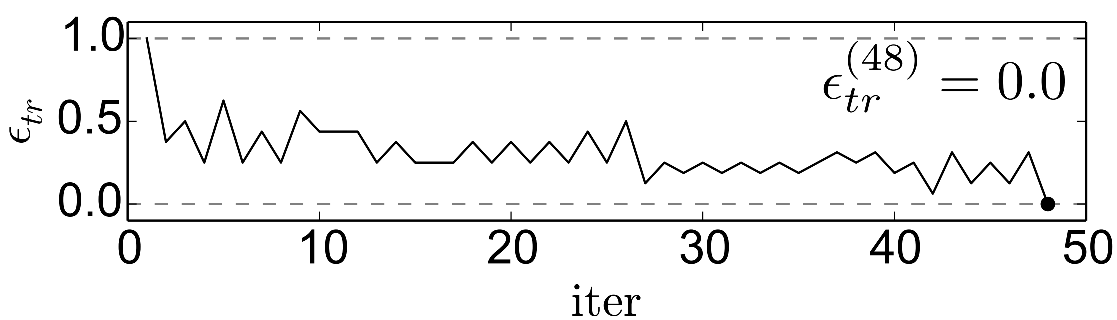
Lifting, Example 1, Iter. 48

● class 1, ● class -1, ●/●=misclassified,









160 points visited.

All data classified correctly. Done.

Final weight:
$\mathbf{w} = (3, -1.5, 3.12)^\top$

Lifting, Example 1, Result

Note that we have used the mapping $\mathbf{x} \leftarrow \begin{bmatrix} x \\ x^2 - 1 \end{bmatrix}$ because of faster perceptron convergence (w.r.t. using just $\begin{bmatrix} x \\ x^2 \end{bmatrix}$).

The final weight vector for the dimensionality-lifted dataset is $\mathbf{w} = (3, -1.5, 3.12)^\top$.

The resulting discriminant function is:

$$f(x) = 3 - 1.5x + 3.12(x^2 - 1) \quad (32)$$

$$= -0.12 - 1.5x + 3.12x^2 . \quad (33)$$