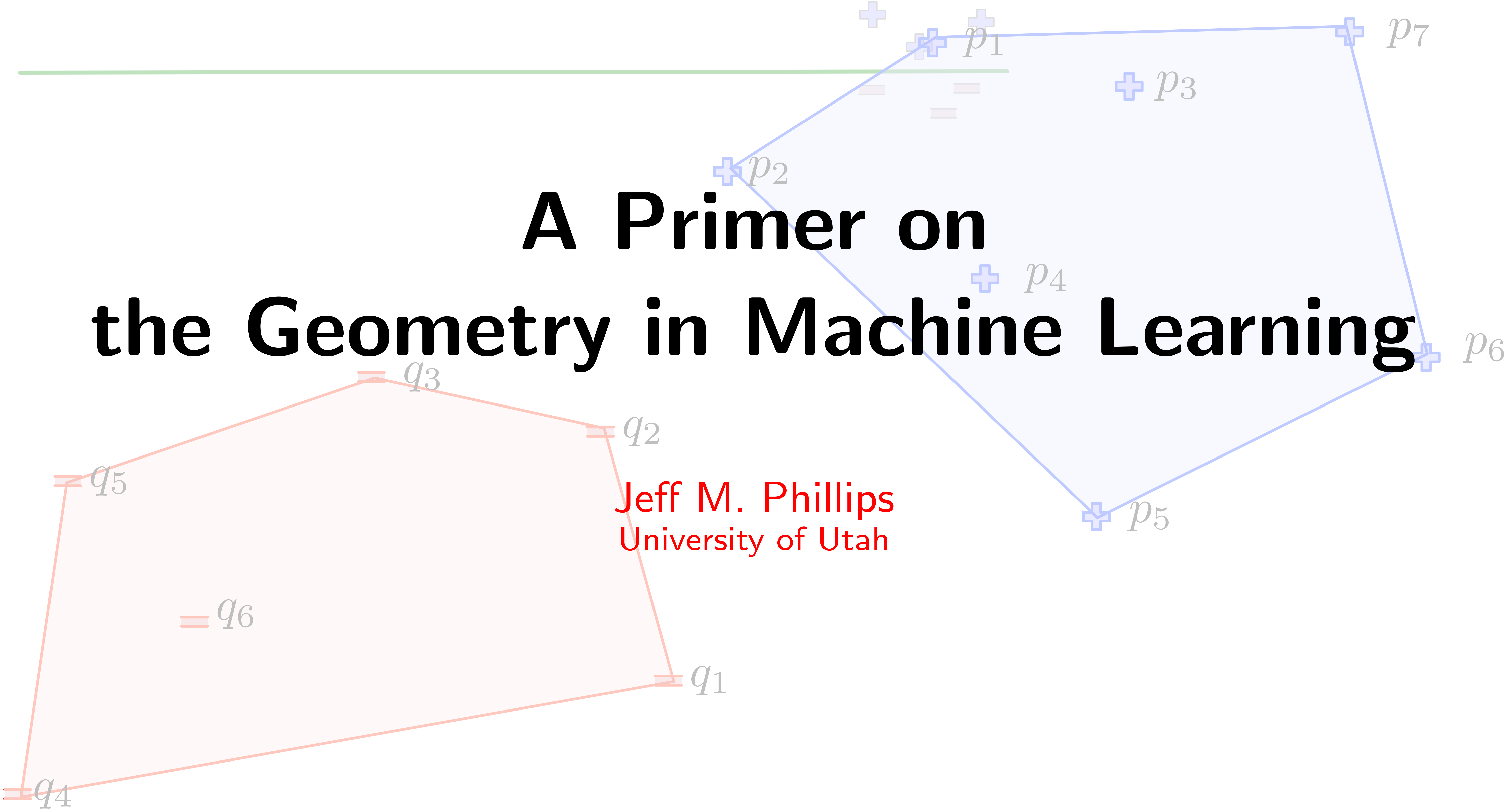
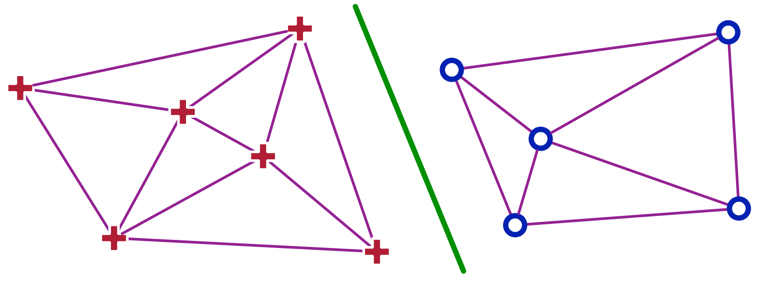
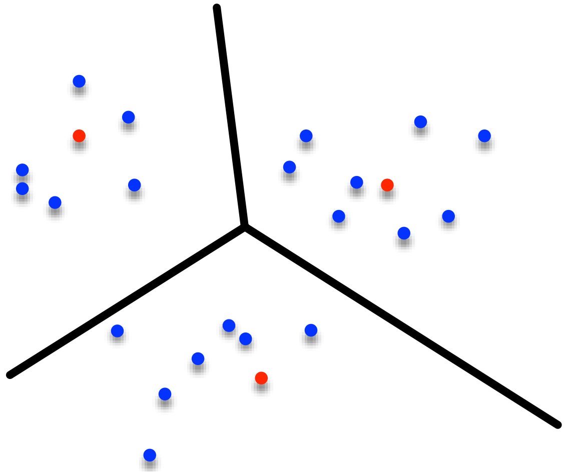
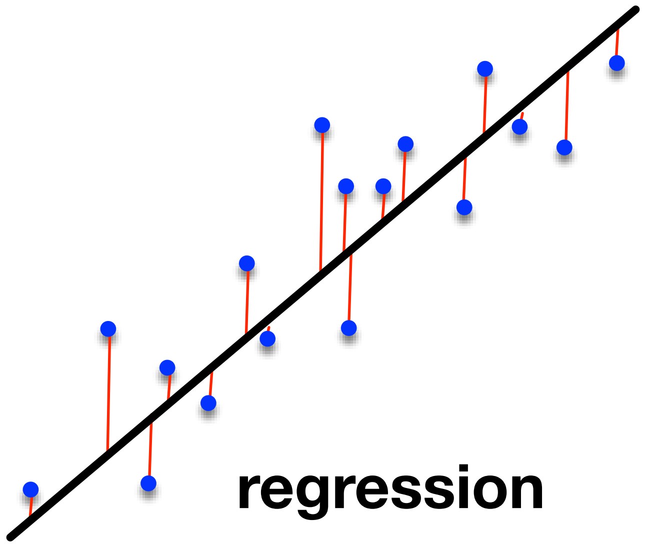
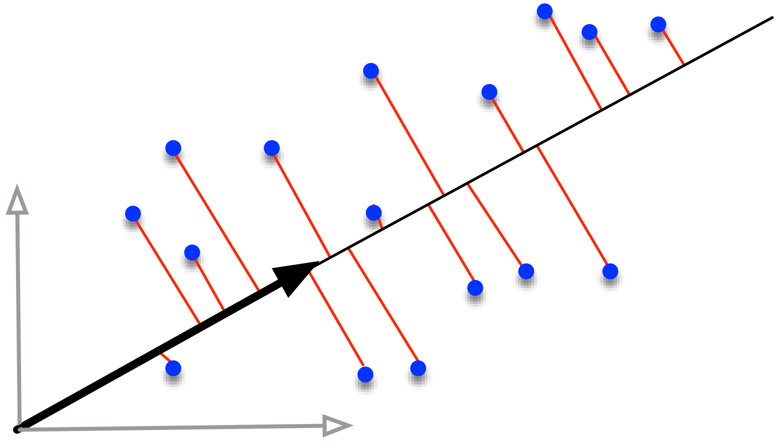
A Primer on the Geometry in Machine Learning

Jeff M. Phillips
University of Utah



What is Machine Learning?

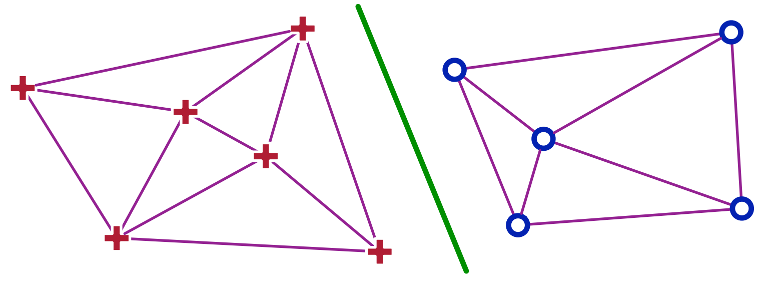
	supervised (has labels)	unsupervised (no labels)
class output	 classification	 clustering
value output	 regression	 dimensionality reduction

What is Machine Learning?

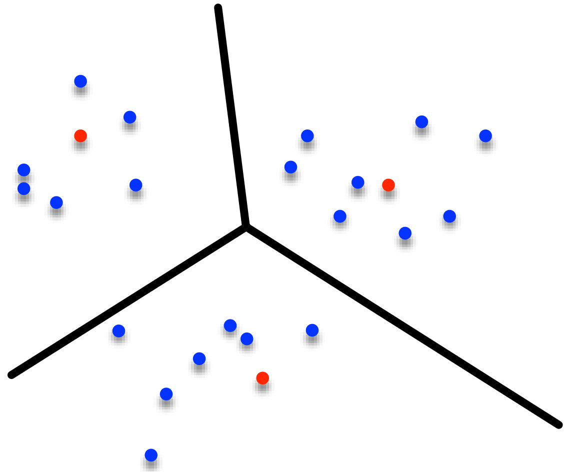
supervised (has labels)

unsupervised (no labels)

class output

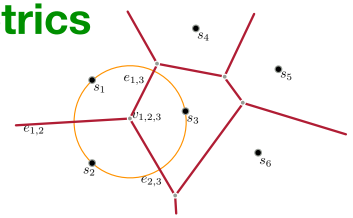


classification



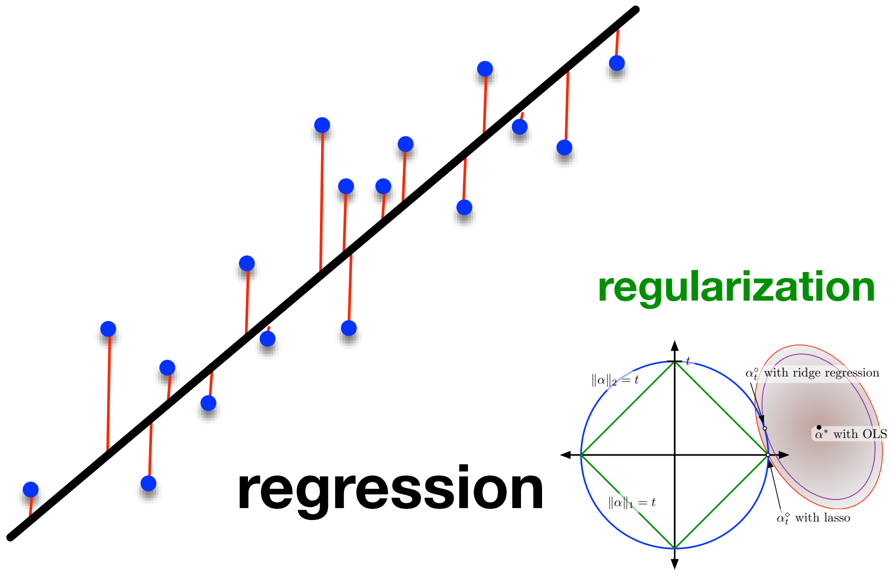
clustering

Voronoi metrics

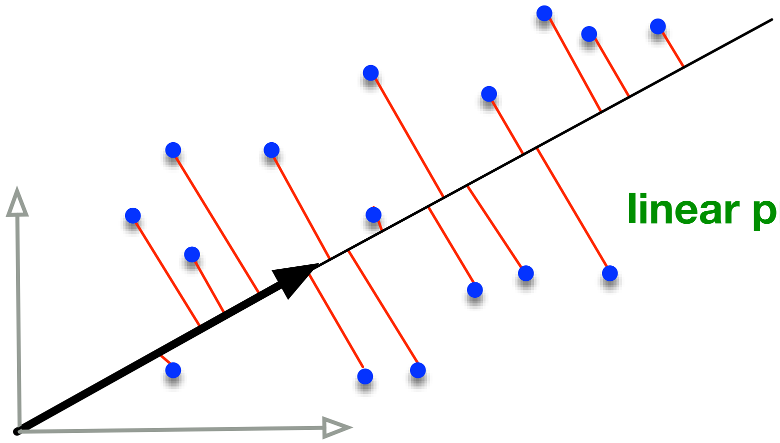


value output

regularization

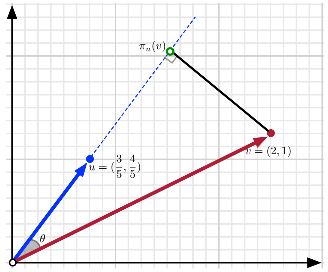

$\|\alpha\|_2 = t$
t
α_t° with ridge regression
$\hat{\alpha}^*$ with OLS
$\|\alpha\|_1 = t$
α_t° with lasso

regression



dimensionality reduction

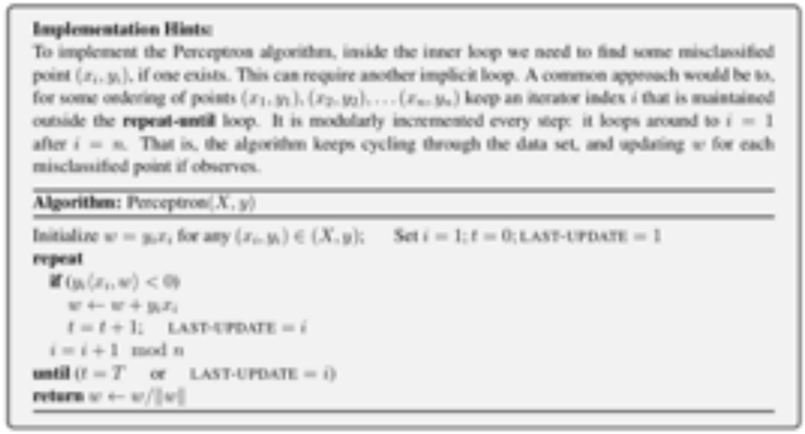
linear projections



Mathematical Foundations for Data Analysis

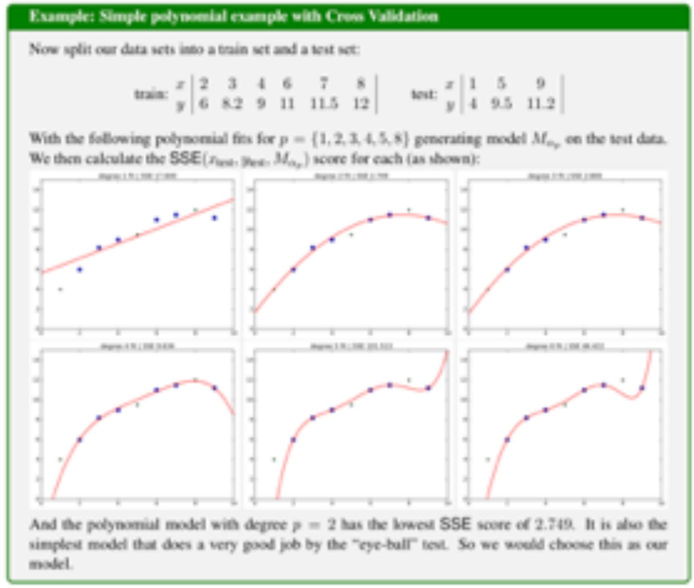
Jeff M. Phillips

http://www.cs.utah.edu/~jeffp/M4D/M4D.html



Implementation Hints:
To implement the Perceptron algorithm, inside the inner loop we need to find some misclassified point (x_i, y_i), if one exists. This can require another implicit loop. A common approach would be to, for some ordering of points $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ keep an iterator index i that is maintained outside the **repeat-until** loop. It is modularly incremented every step: it loops around to $i = 1$ after $i = n$. That is, the algorithm keeps cycling through the data set, and updating w for each misclassified point if observes.

```
Algorithm: Perceptron($X, y$)
Initialize $w = y_i x_i$ for any $(x_i, y_i) \in (X, y)$;     Set $i = 1$; $t = 0$; LAST-UPDATE $= 1$
repeat
  if $(y_i \langle x_i, w \rangle < 0)$
    $w \leftarrow w + y_i x_i$
    $t = t + 1$;    LAST-UPDATE $= i$
  $i = i + 1 \mod n$
until $(t = T$    or    LAST-UPDATE $= i)$
return $w \leftarrow w / \|w\|$
```



Example: Simple polynomial example with Cross Validation

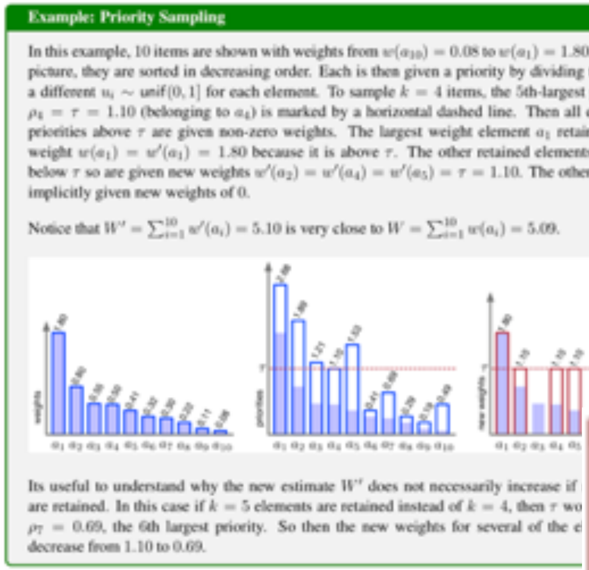
Now split our data sets into a train set and a test set:

	x	2	3	4	6	7	8
train:	y	6	8.2	9	11	11.5	12

	x	1	5	9
test:	y	4	9.5	11.2

With the following polynomial fits for $p = \{1, 2, 3, 4, 5, 8\}$ generating model M_{α_p} on the test data. We then calculate the $\mathsf{SSE}(x_{\text{test}}, y_{\text{test}}, M_{\alpha_p})$ score for each (as shown):

And the polynomial model with degree $p = 2$ has the lowest SSE score of 2.749. It is also the simplest model that does a very good job by the "eye-ball" test. So we would choose this as our model.

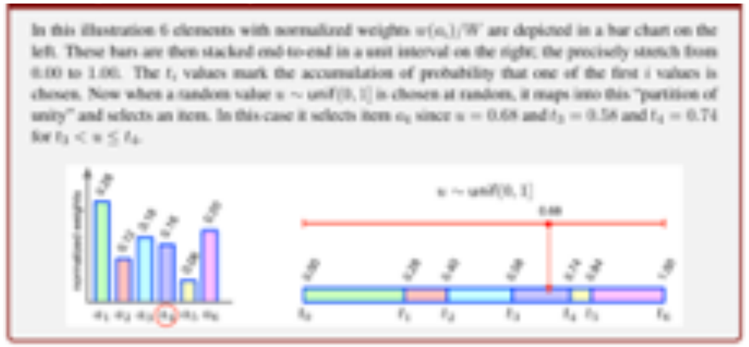


Example: Priority Sampling

In this example, 10 items are shown with weights from $w(a_{10}) = 0.08$ to $w(a_1) = 1.80$. For a clearer picture, they are sorted in decreasing order. Each is then given a priority by dividing the weight by a different $u_i \sim \mathsf{unif}(0, 1]$ for each element. To sample $k = 4$ items, the 5th-largest priority value $\rho_4 = \tau = 1.10$ (belonging to a_4) is marked by a horizontal dashed line. Then all elements with priorities above τ are given non-zero weights. The largest weight element a_1 retains its original weight $w(a_1) = w'(a_1) = 1.80$ because it is above τ. The other retained elements have weight below τ so are given new weights $w'(a_2) = w'(a_4) = w'(a_5) = \tau = 1.10$. The other elements are implicitly given new weights of 0.

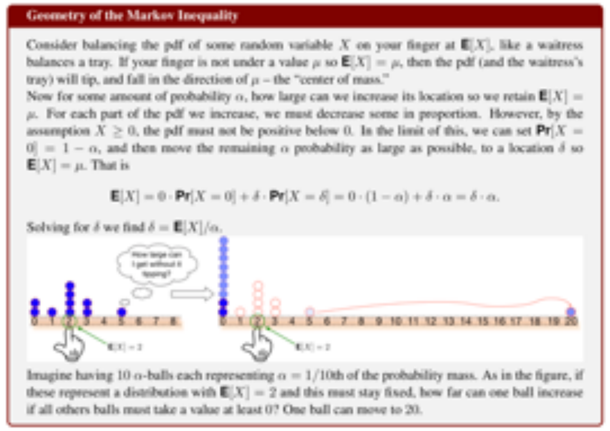
Notice that $W' = \sum_{i=1}^{10} w'(a_i) = 5.10$ is very close to $W = \sum_{i=1}^{10} w(a_i) = 5.09$.

Its useful to understand why the new estimate W' does not necessarily increase if
are retained. In this case if $k = 5$ elements are retained instead of $k = 4$, then τ wo
$\rho_7 = 0.69$, the 6th largest priority. So then the new weights for several of the e
decrease from 1.10 to 0.69.



In this illustration 6 elements with normalized weights $w(a_i)/W$ are depicted in a bar chart on the left. These bars are then stacked end-to-end in a unit interval on the right; the precisely stretch from 0.00 to 1.00. The t_i values mark the accumulation of probability that one of the first i values is chosen. Now when a random value $u \sim \mathsf{unif}(0, 1]$ is chosen at random, it maps into this "partition of unity" and selects an item. In this case it selects item a_4 since $u = 0.68$ and $t_3 = 0.58$ and $t_4 = 0.74$ for $t_3 < u \leq t_4$.

normalized weights
0.28
0.12
0.18
0.16
0.06
0.20
a_1 a_2 a_3 a_4 a_5 a_6
$u \sim \mathsf{unif}(0, 1]$
0.68
0.00
0.28
0.40
0.58
0.74
0.80
1.00
t_0 t_1 t_2 t_3 t_4 t_5 t_6



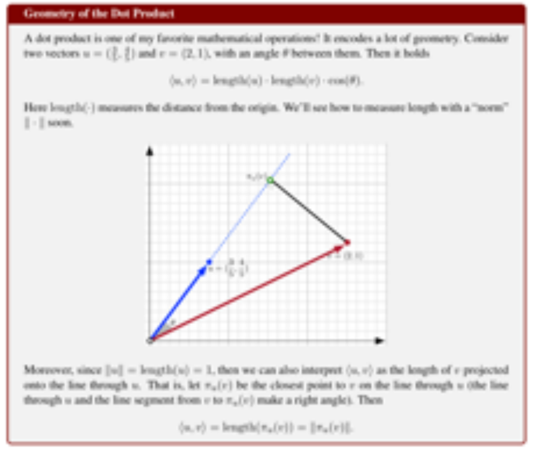
Geometry of the Markov Inequality

Consider balancing the pdf of some random variable X on your finger at $\mathbf{E}[X]$, like a waitress balances a tray. If your finger is not under a value μ so $\mathbf{E}[X] = \mu$, then the pdf (and the waitress's tray) will tip, and fall in the direction of μ – the "center of mass."
Now for some amount of probability α, how large can we increase its location so we retain $\mathbf{E}[X] = \mu$. For each part of the pdf we increase, we must decrease some in proportion. However, by the assumption $X \geq 0$, the pdf must not be positive below 0. In the limit of this, we can set $\mathbf{Pr}[X = 0] = 1 - \alpha$, and then move the remaining α probability as large as possible, to a location δ so $\mathbf{E}[X] = \mu$. That is

$$\mathbf{E}[X] = 0 \cdot \mathbf{Pr}[X = 0] + \delta \cdot \mathbf{Pr}[X = \delta] = 0 \cdot (1 - \alpha) + \delta \cdot \alpha = \delta \cdot \alpha.$$

Solving for δ we find $\delta = \mathbf{E}[X]/\alpha$.

Imagine having 10 α-balls each representing $\alpha = 1/10$th of the probability mass. As in the figure, if these represent a distribution with $\mathbf{E}[X] = 2$ and this must stay fixed, how far can one ball increase if all others balls must take a value at least 0? One ball can move to 20.



Geometry of the Dot Product

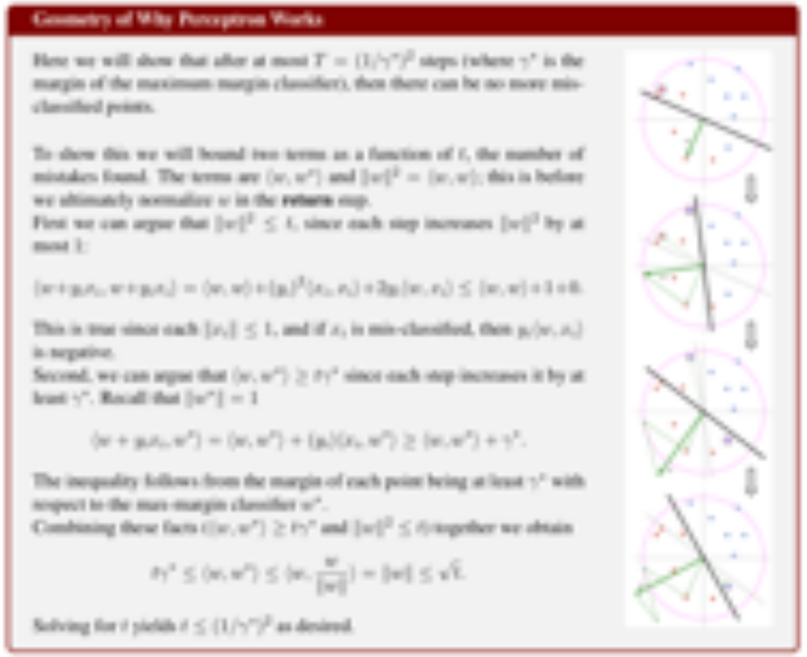
A dot product is one of my favorite mathematical operations! It encodes a lot of geometry. Consider two vectors $u = (\frac{3}{5}, \frac{4}{5})$ and $v = (2, 1)$, with an angle θ between them. Then it holds

$$\langle u, v \rangle = \mathrm{length}(u) \cdot \mathrm{length}(v) \cdot \cos(\theta).$$

Here $\mathrm{length}(\cdot)$ measures the distance from the origin. We'll see how to measure length with a "norm" $\|\cdot\|$ soon.

Moreover, since $\|u\| = \mathrm{length}(u) = 1$, then we can also interpret $\langle u, v \rangle$ as the length of v projected onto the line through u. That is, let $\pi_u(v)$ be the closest point to v on the line through u (the line through u and the line segment from v to $\pi_u(v)$ make a right angle). Then

$$\langle u, v \rangle = \mathrm{length}(\pi_u(v)) = \|\pi_u(v)\|.$$



Geometry of Why Perceptron Works

Here we will show that after at most $T = (1/\gamma^*)^2$ steps (where γ^* is the margin of the maximum margin classifier), then there can be no more mis-classified points.

To show this we will bound two terms as a function of t, the number of mistakes found. The terms are $\langle w, w^* \rangle$ and $\|w\|^2 = \langle w, w \rangle$; this is before we ultimately normalize w in the **return** step.
First we can argue that $\|w\|^2 \leq t$, since each step increases $\|w\|^2$ by at most 1:

$$\langle w + y_i x_i, w + y_i x_i \rangle = \langle w, w \rangle + (y_i)^2 \langle x_i, x_i \rangle + 2 y_i \langle w, x_i \rangle \leq \langle w, w \rangle + 1 + 0.$$

This is true since each $\|x_i\| \leq 1$, and if x_i is mis-classified, then $y_i \langle w, x_i \rangle$ is negative.
Second, we can argue that $\langle w, w^* \rangle \geq t\gamma^*$ since each step increases it by at least γ^*. Recall that $\|w^*\| = 1$

$$\langle w + y_i x_i, w^* \rangle = \langle w, w^* \rangle + (y_i)\langle x_i, w^* \rangle \geq \langle w, w^* \rangle + \gamma^*.$$

The inequality follows from the margin of each point being at least γ^* with respect to the max-margin classifier w^*.
Combining these facts ($\langle w, w^* \rangle \geq t\gamma^*$ and $\|w\|^2 \leq t$) together we obtain

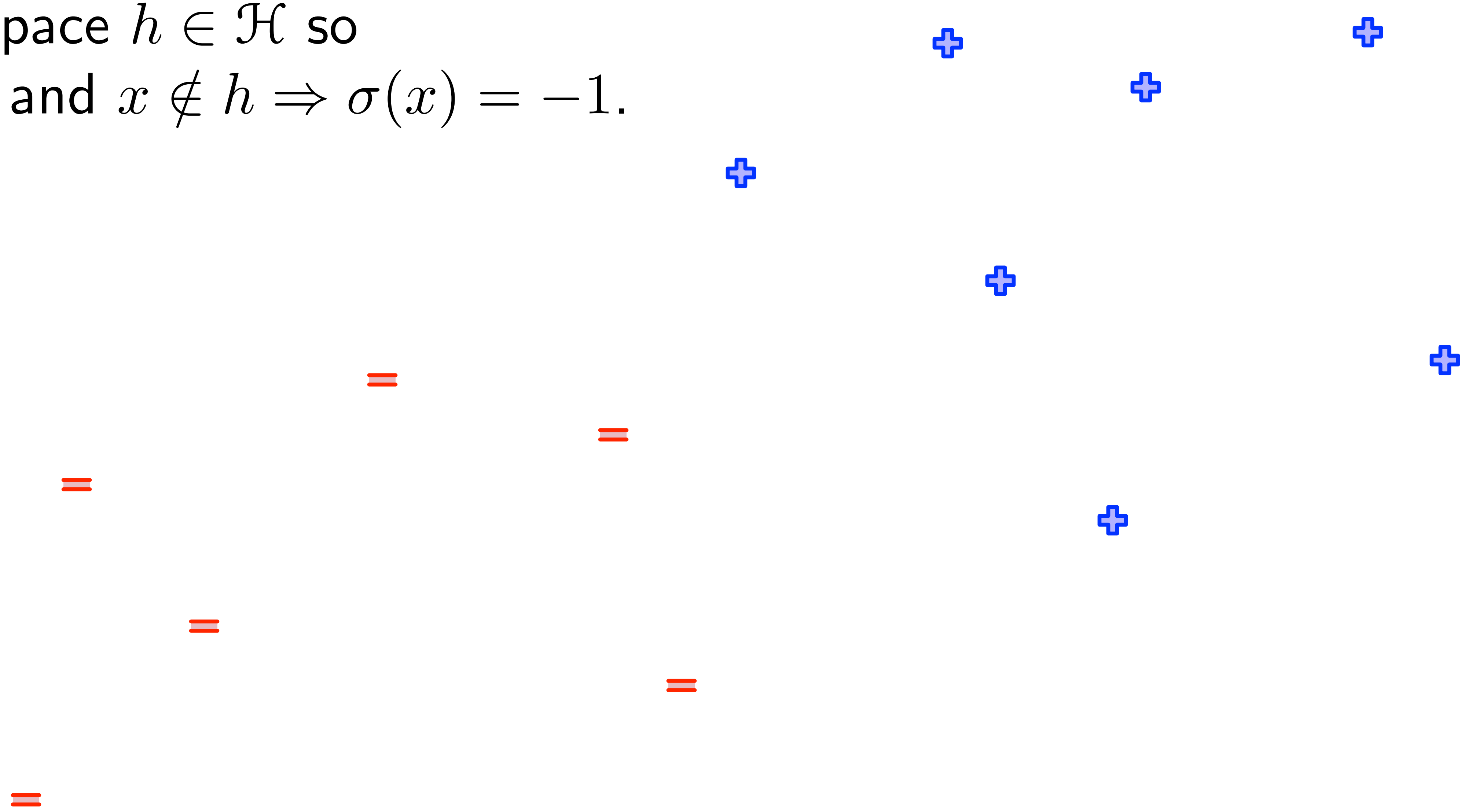
$$t\gamma^* \leq \langle w, w^* \rangle \leq \langle w, \frac{w}{\|w\|} \rangle = \|w\| \leq \sqrt{t}.$$

Solving for t yields $t \leq (1/\gamma^*)^2$ as desired.

What is Machine Learning?

Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

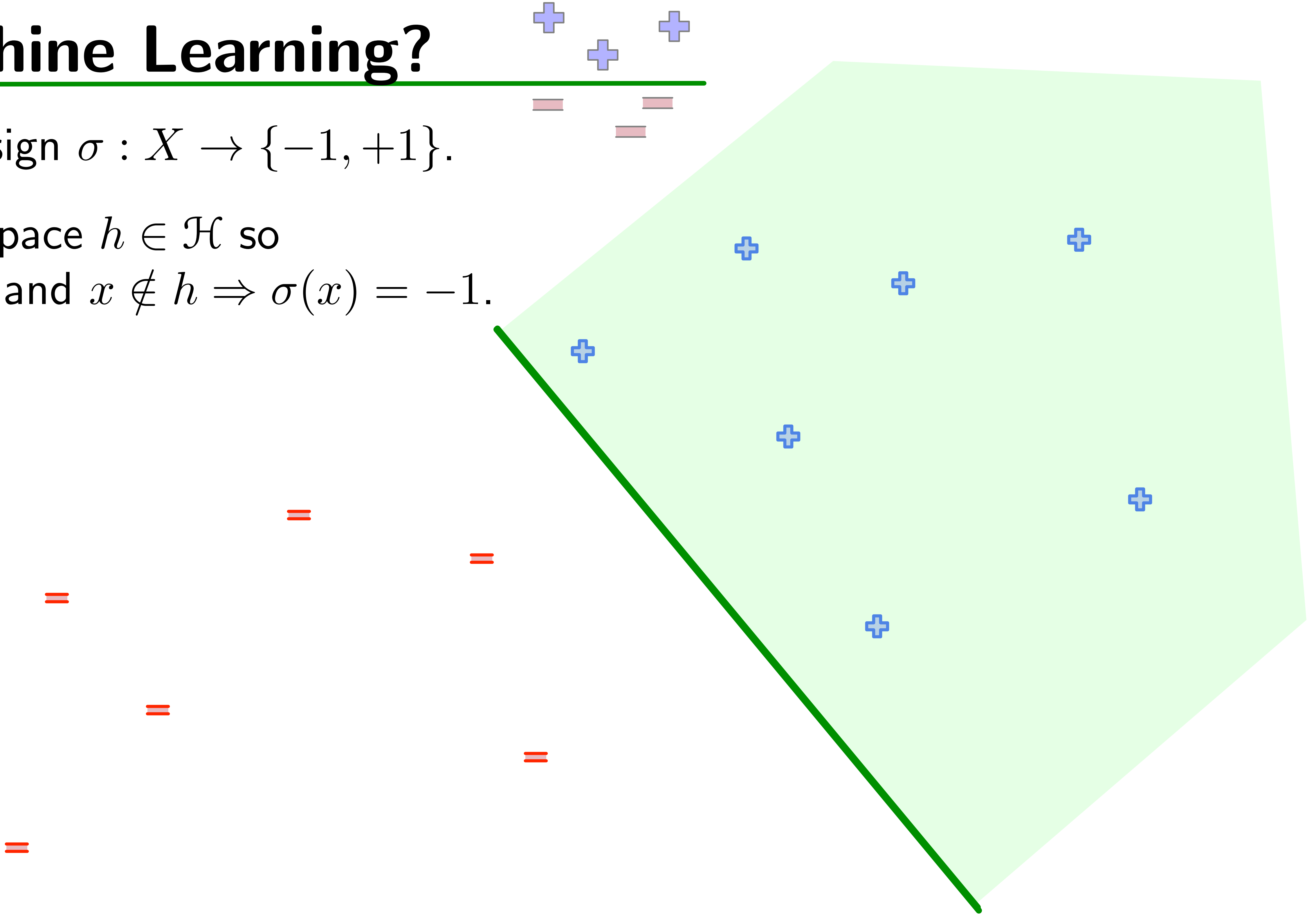
Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.



What is Machine Learning?

Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

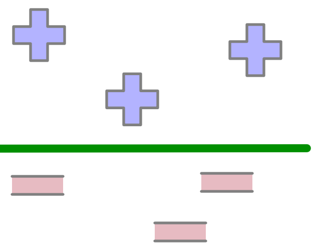
Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.

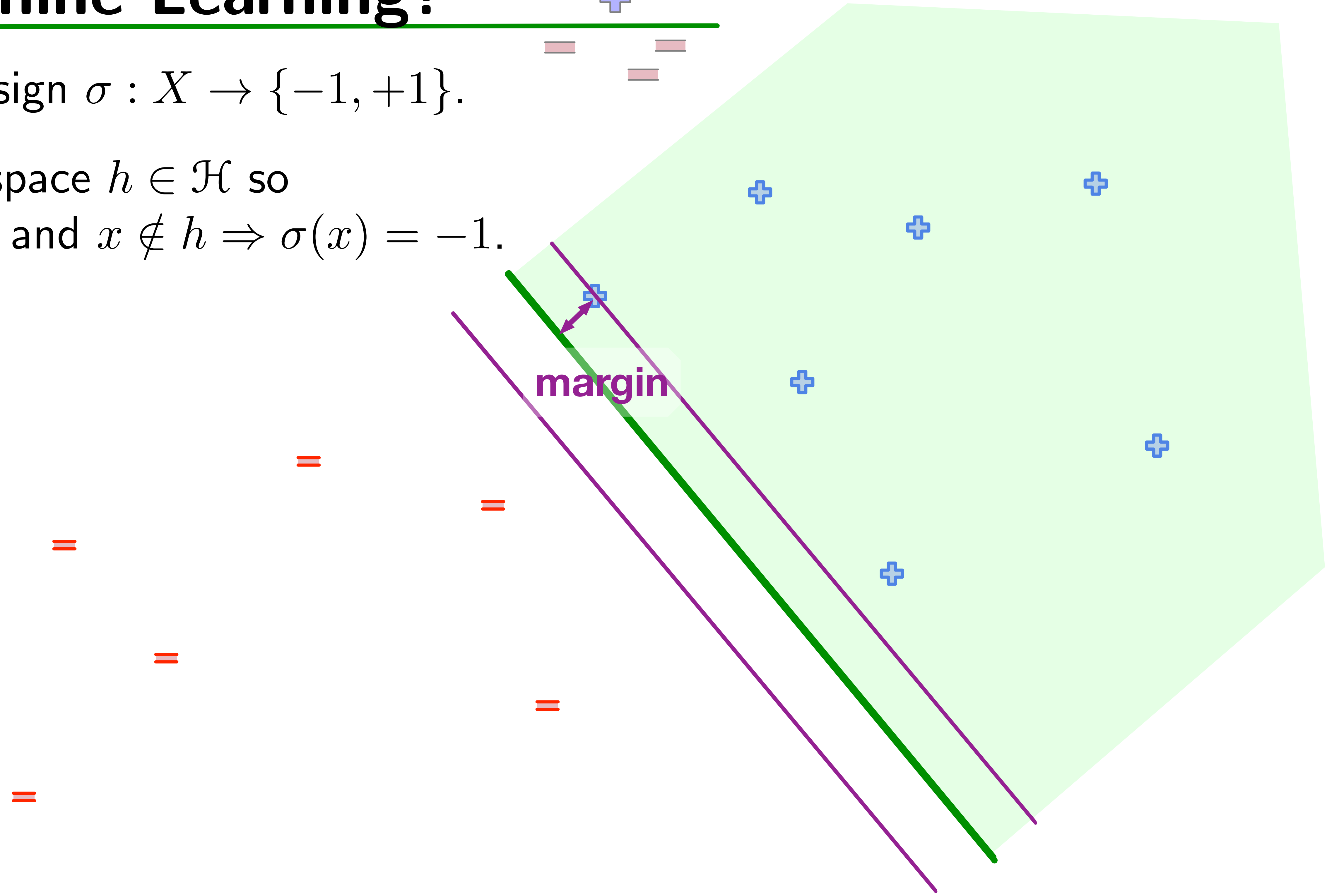


What is Machine Learning?

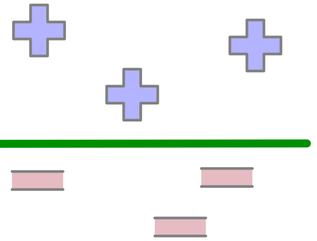
Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.



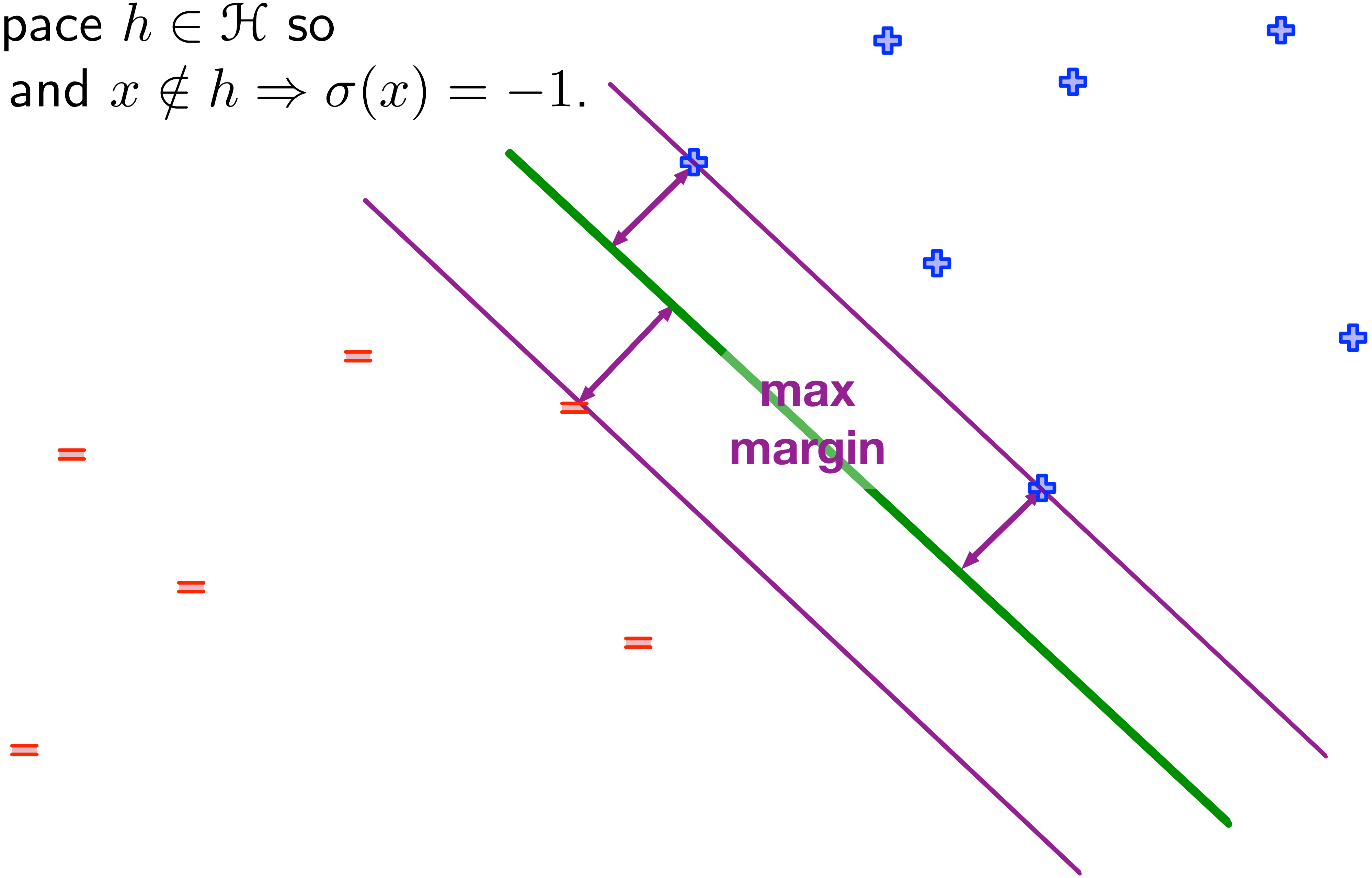


What is Machine Learning?



Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

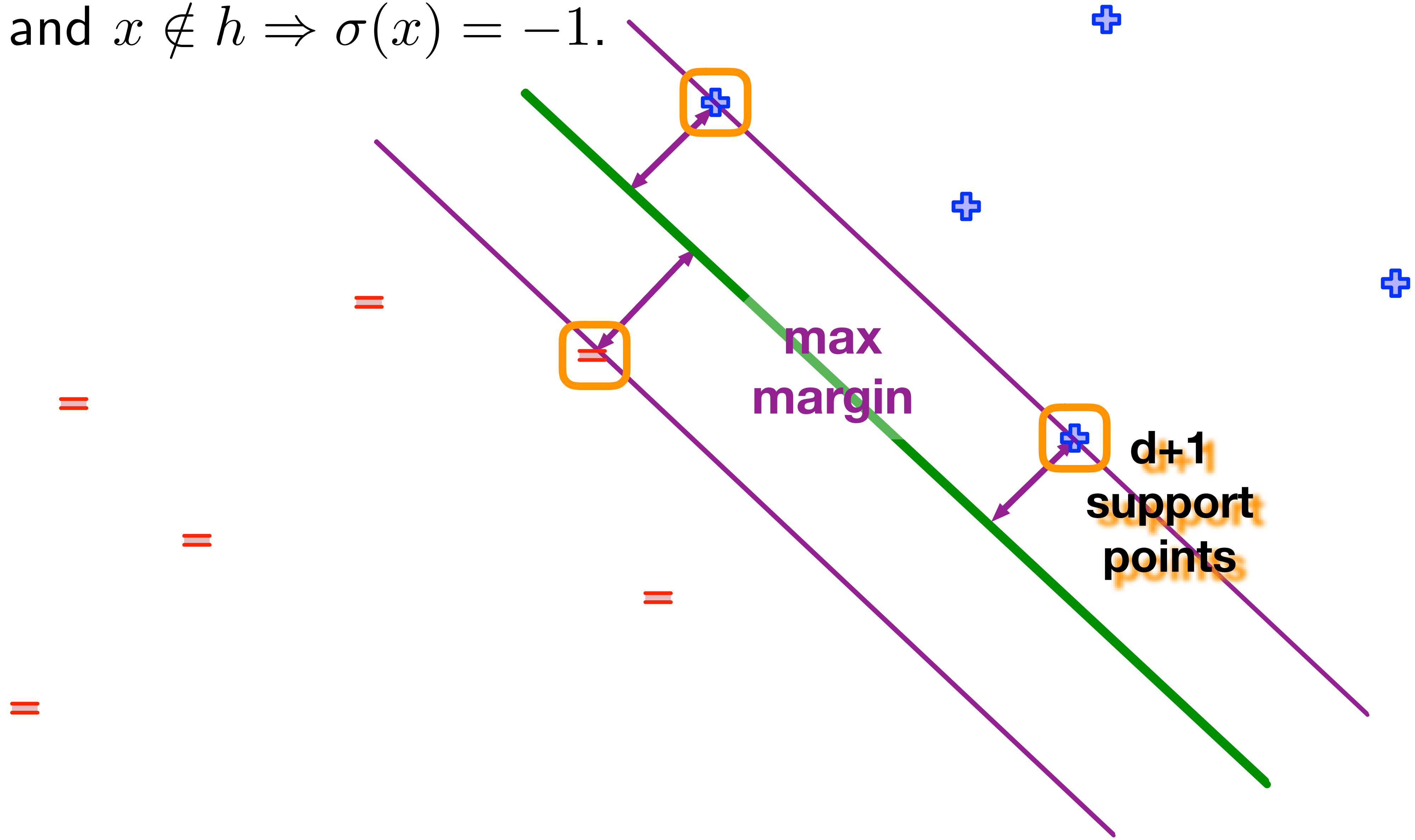
Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.



What is Machine Learning?

Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

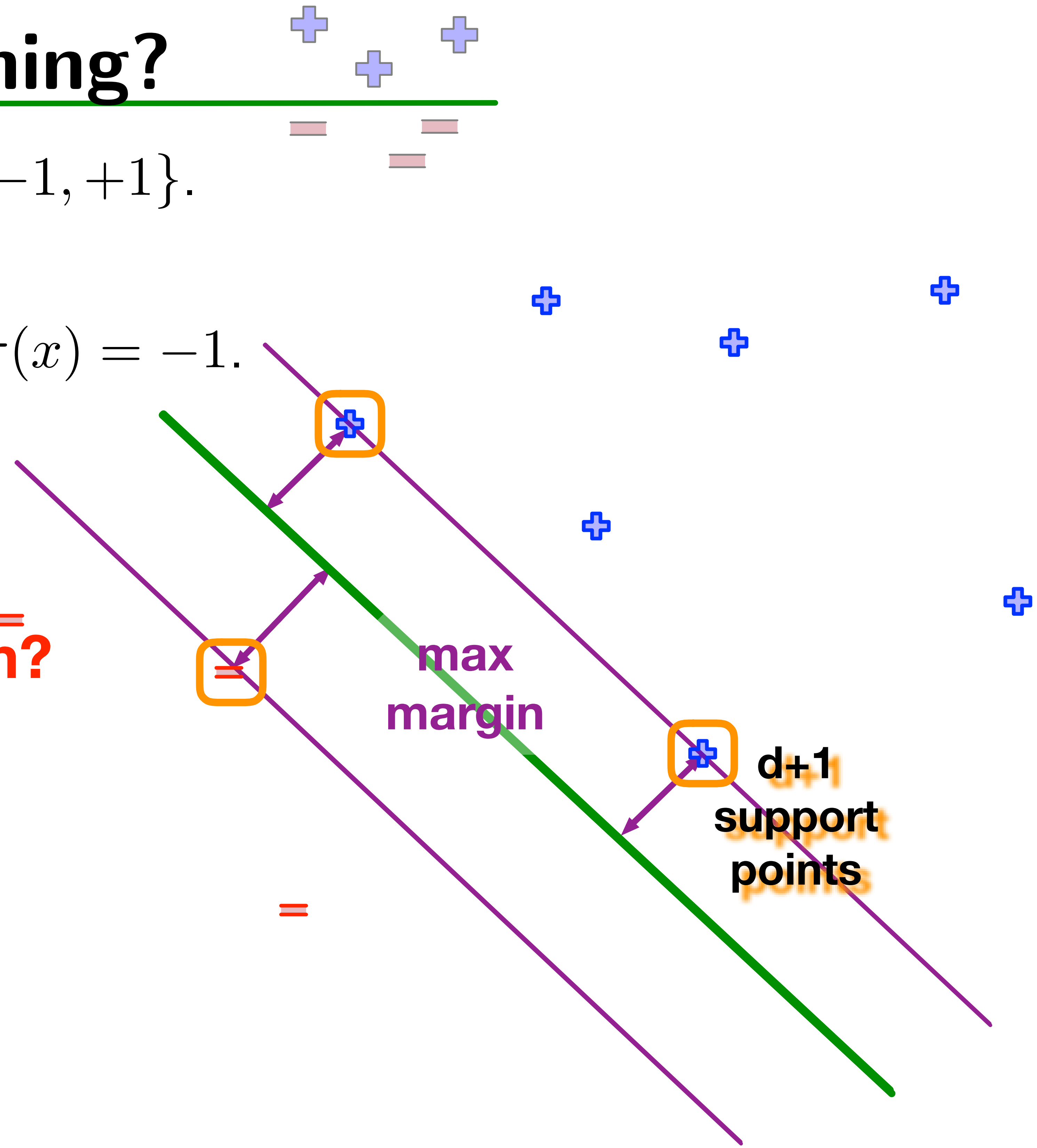
Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.



What is Machine Learning?

Given $X \in \mathbb{R}^d$ with sign $\sigma : X \to \{-1, +1\}$.

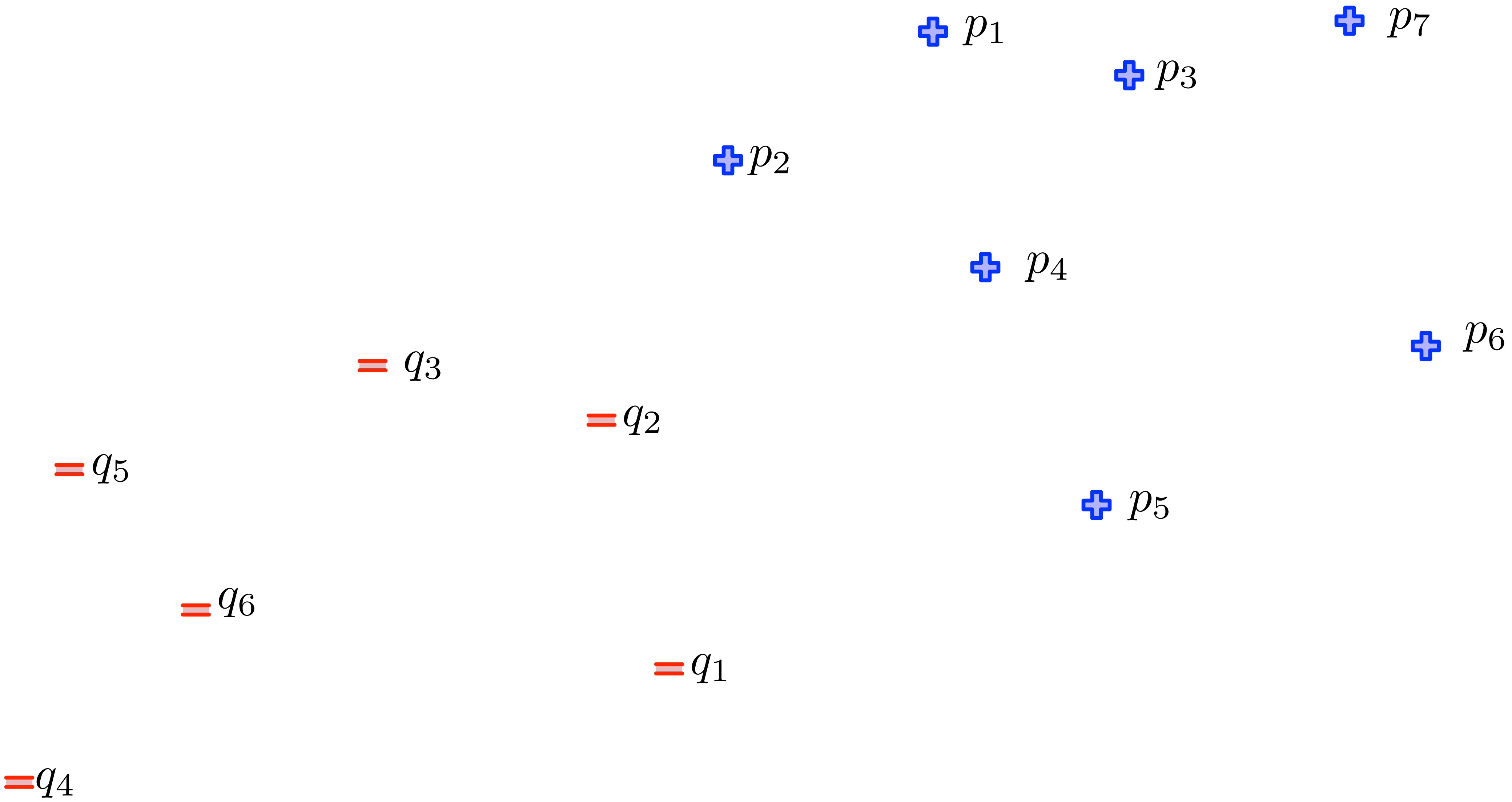
Find separating halfspace $h \in \mathcal{H}$ so
$x \in h \Rightarrow \sigma(x) = +1$ and $x \notin h \Rightarrow \sigma(x) = -1$.



How do we solve this problem?

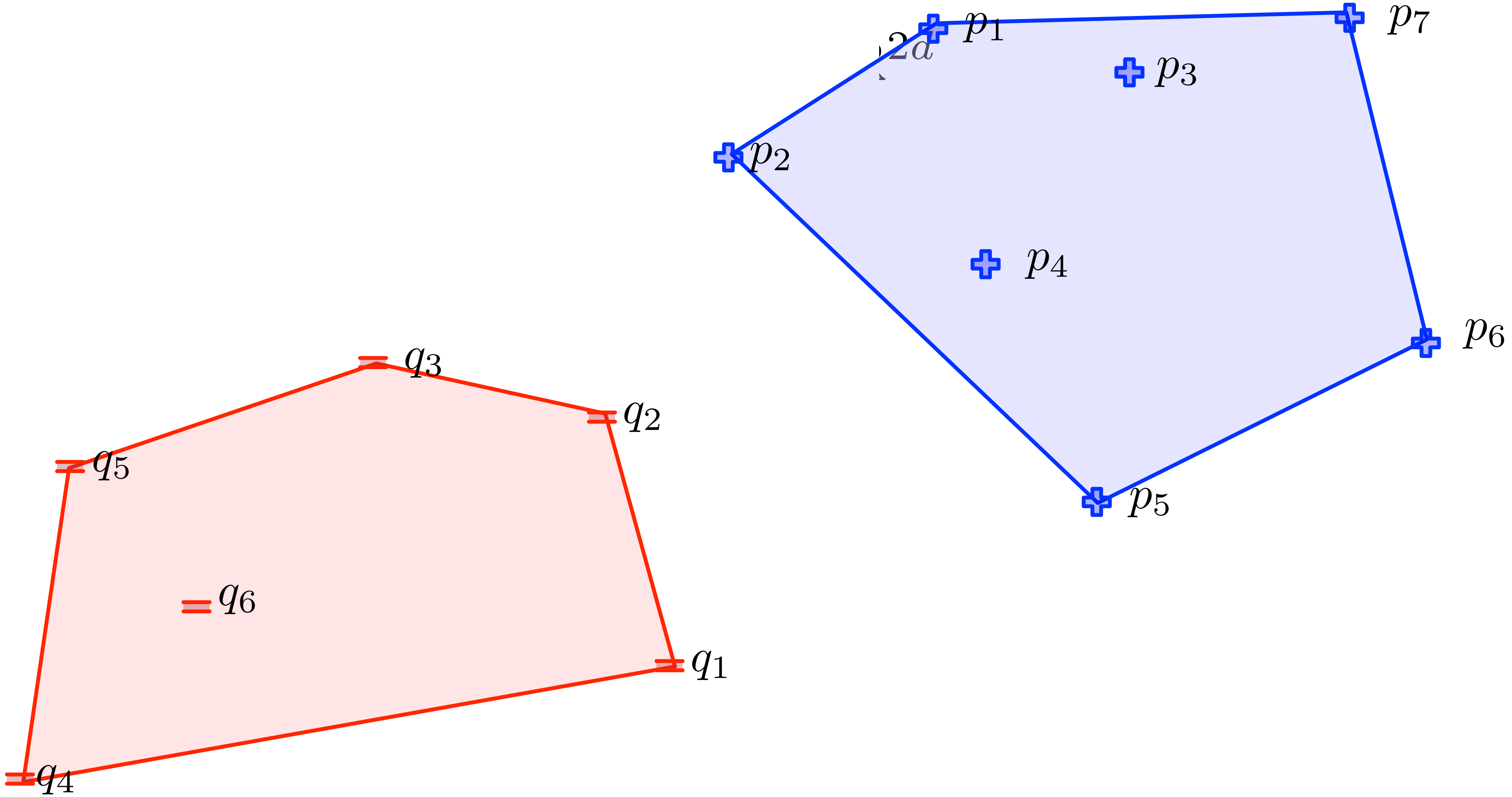
Polytope Distance

Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

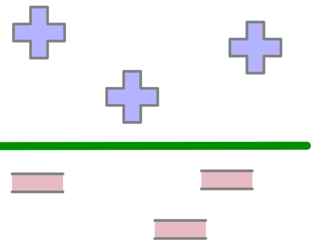


Polytope Distance

Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

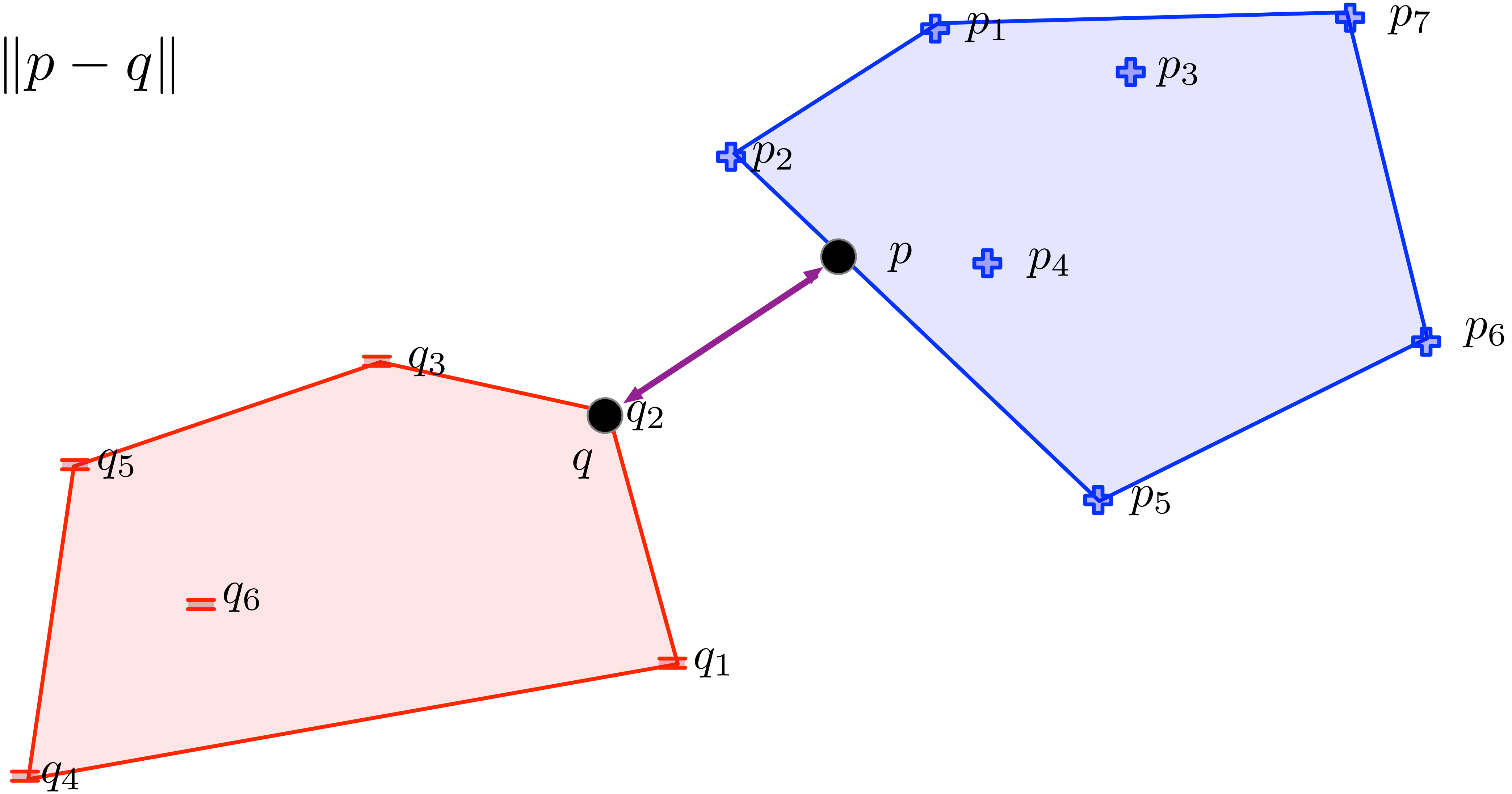


Polytope Distance



Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

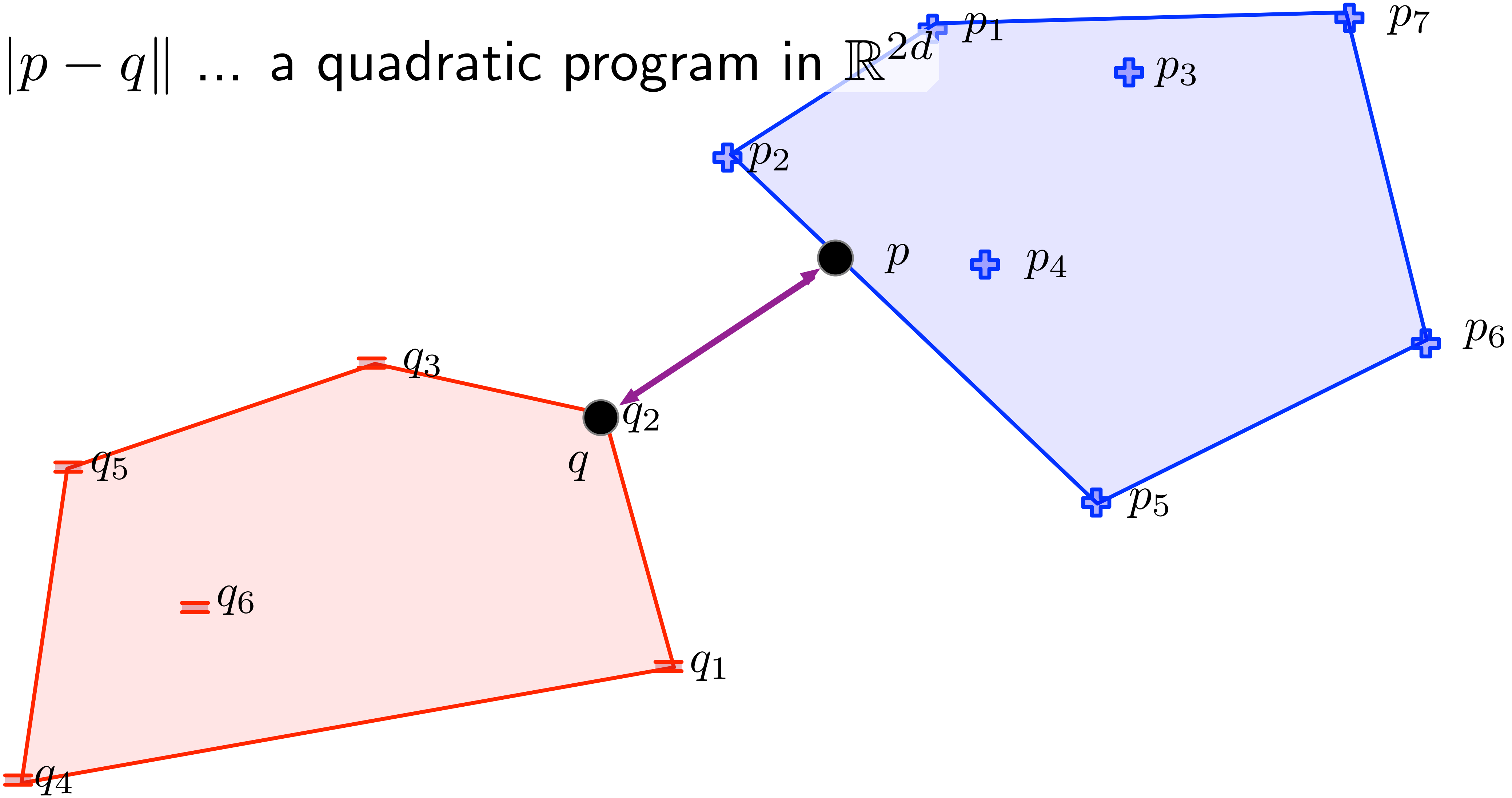
Find $\arg\min_{p \in P, q \in Q} \|p - q\|$



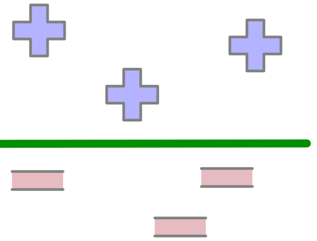
Polytope Distance

Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ ... a quadratic program in $\mathbb{R}^{2d}$



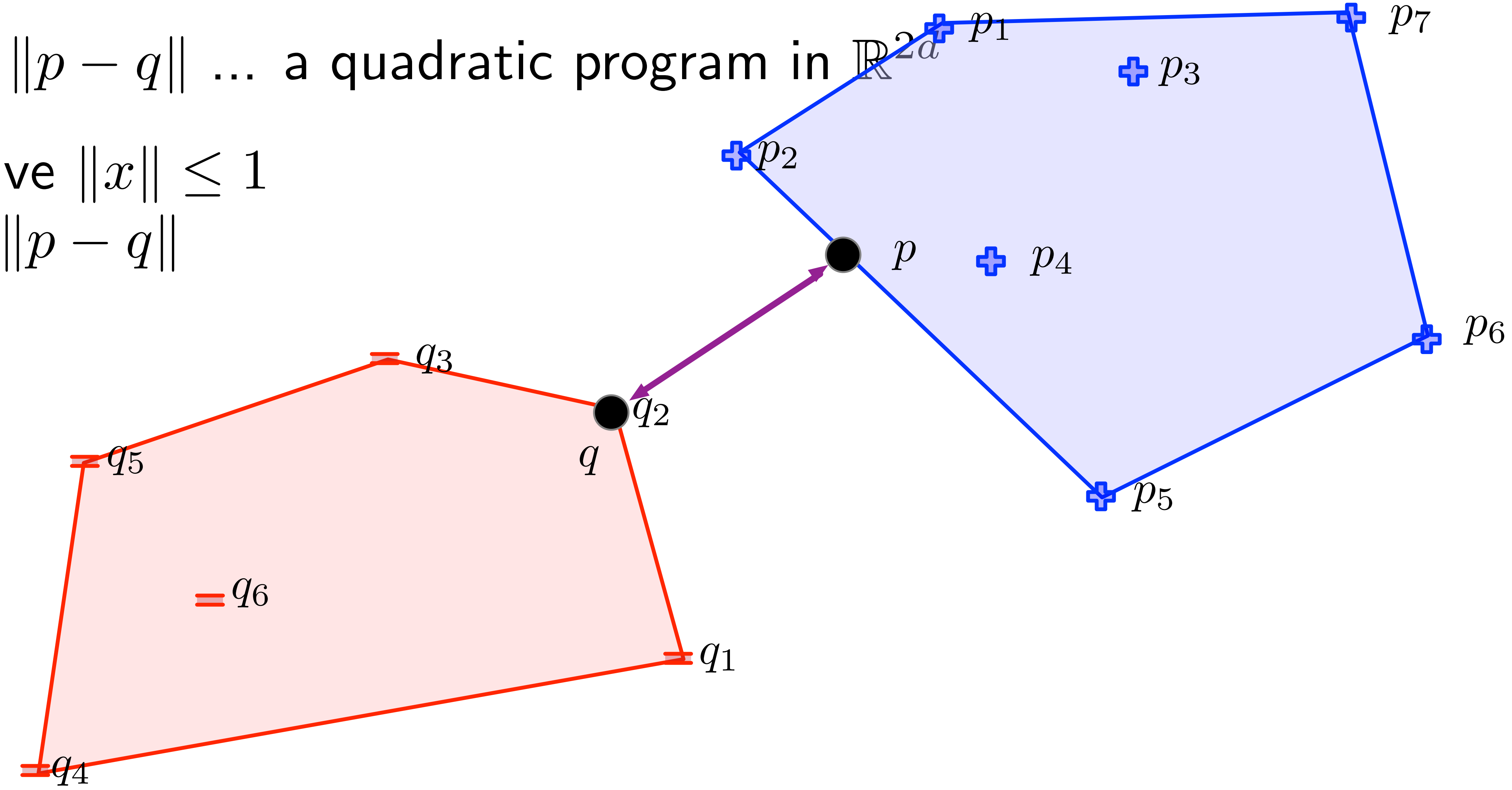
Polytope Distance



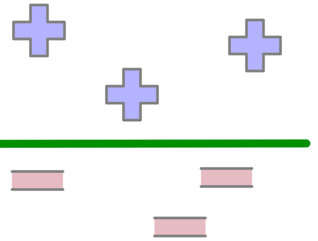
Define polytopes $P = \text{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \text{CH}(x \in X \mid \sigma(x) = -1)$

Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ ... a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$



Polytope Distance

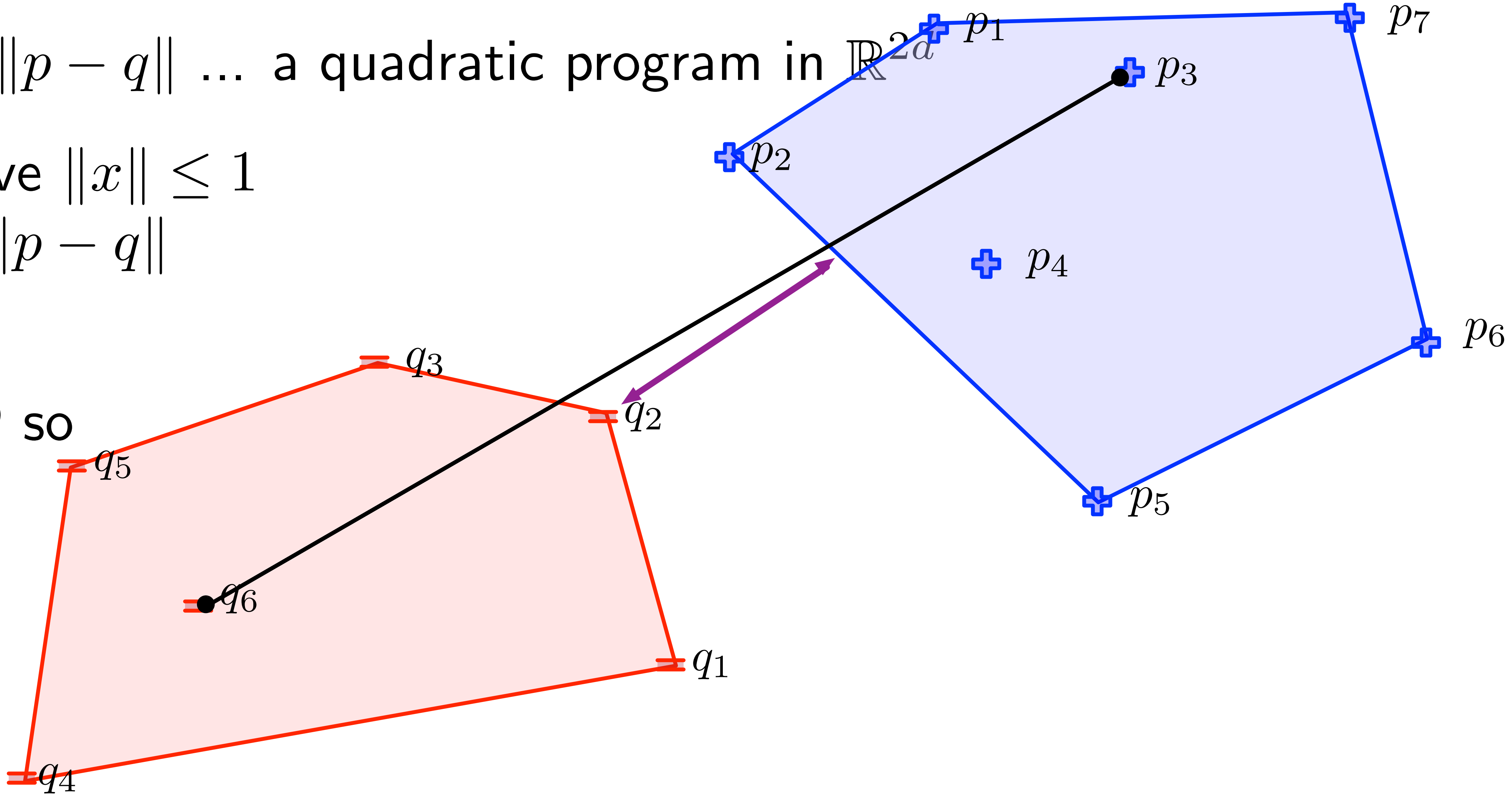


Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

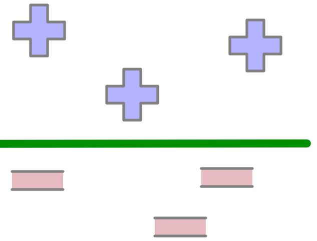
Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ … a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$

Iterate $1/(\varepsilon\gamma^2)$ steps,
find $\hat{p} \in P$ and $\hat{q} \in Q$ so
$(1 - \varepsilon)\|\hat{p} - \hat{q}\| \leq \gamma$



Polytope Distance



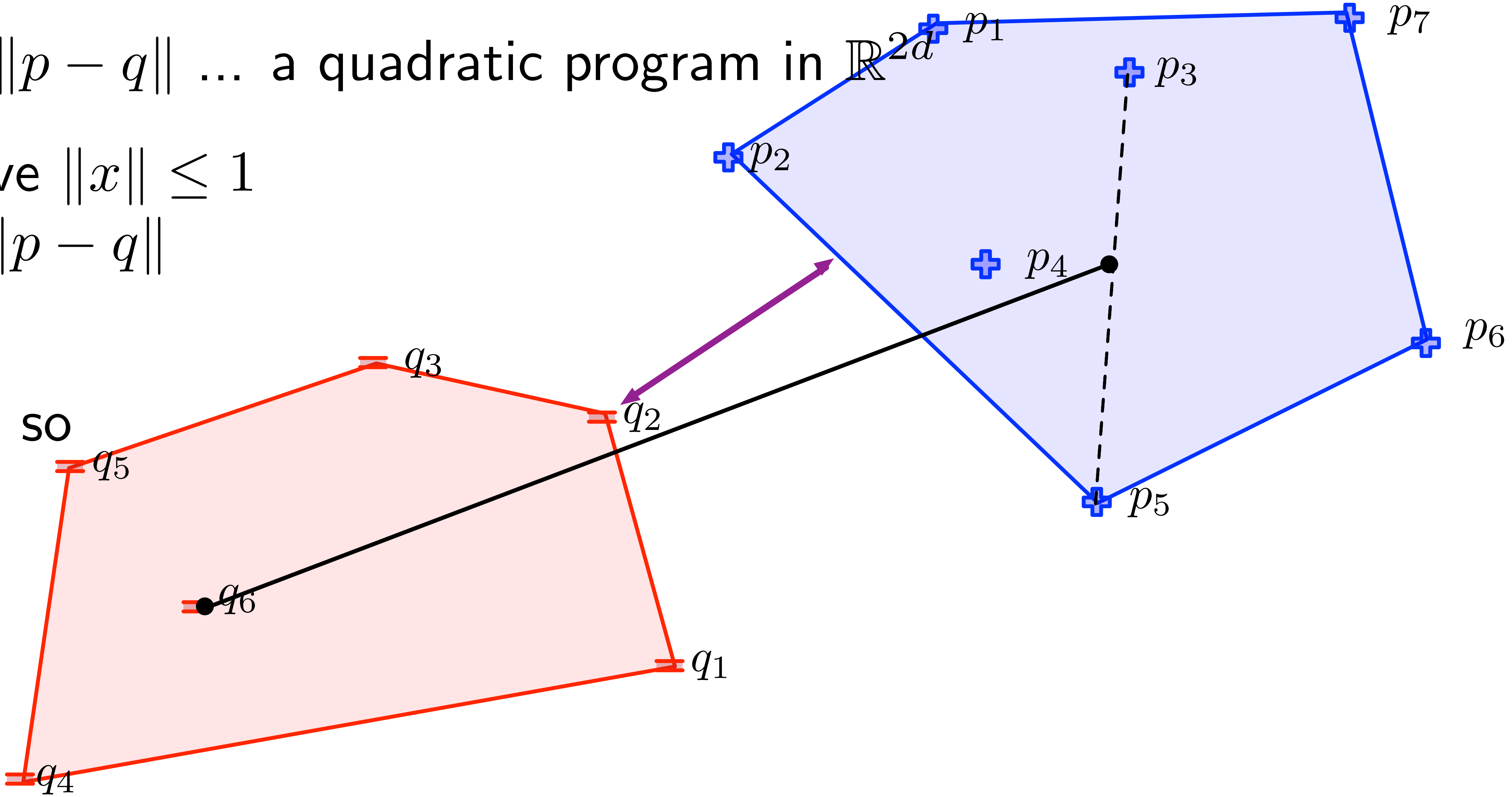
Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ ... a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$

Iterate $1/(\varepsilon\gamma^2)$ steps,
find $\hat{p} \in P$ and $\hat{q} \in Q$ so

$$(1 - \varepsilon)\|\hat{p} - \hat{q}\| \leq \gamma$$



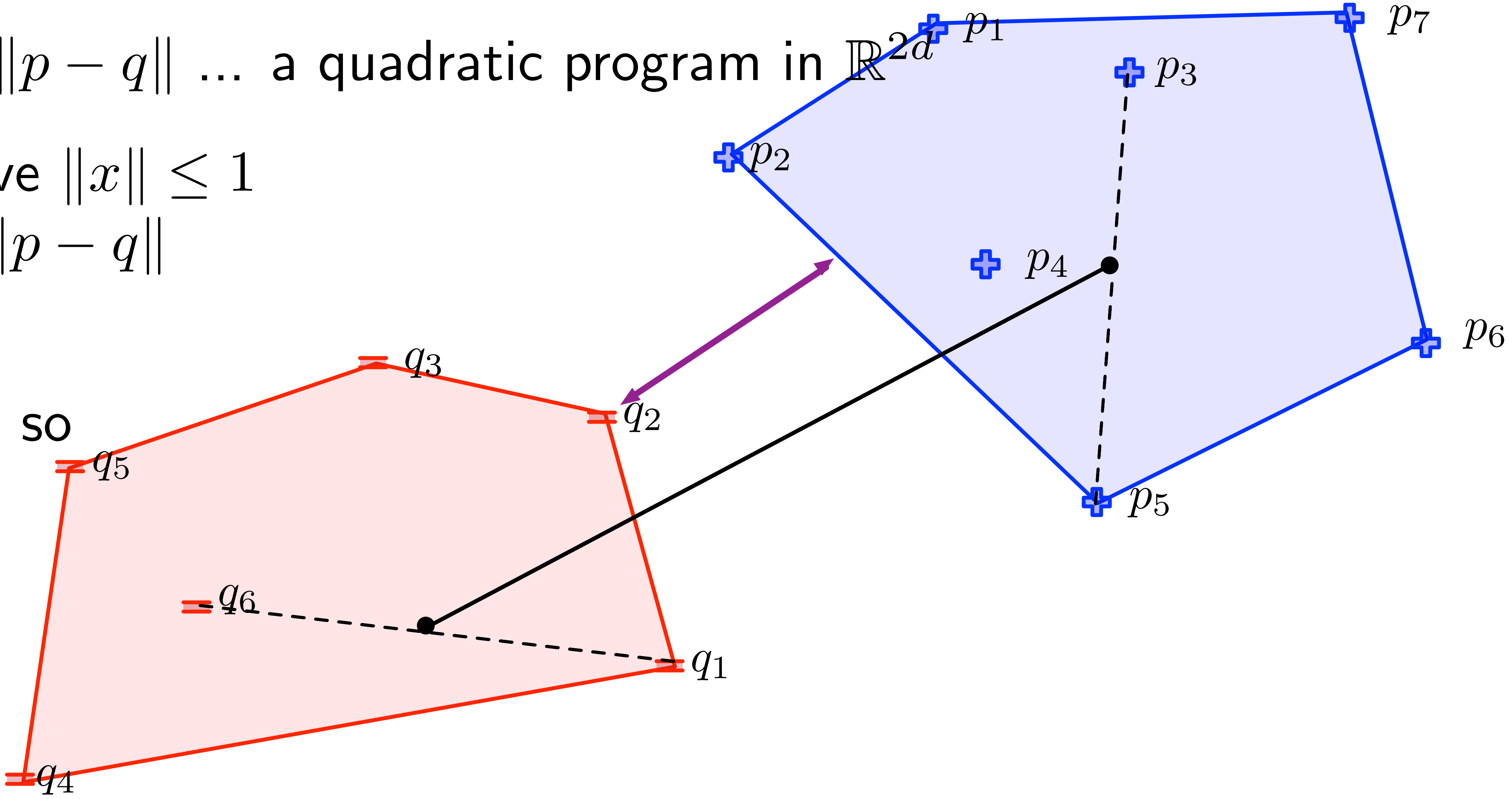
Polytope Distance

Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ ... a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$

Iterate $1/(\varepsilon\gamma^2)$ steps,
find $\hat{p} \in P$ and $\hat{q} \in Q$ so
$$(1 - \varepsilon)\|\hat{p} - \hat{q}\| \leq \gamma$$



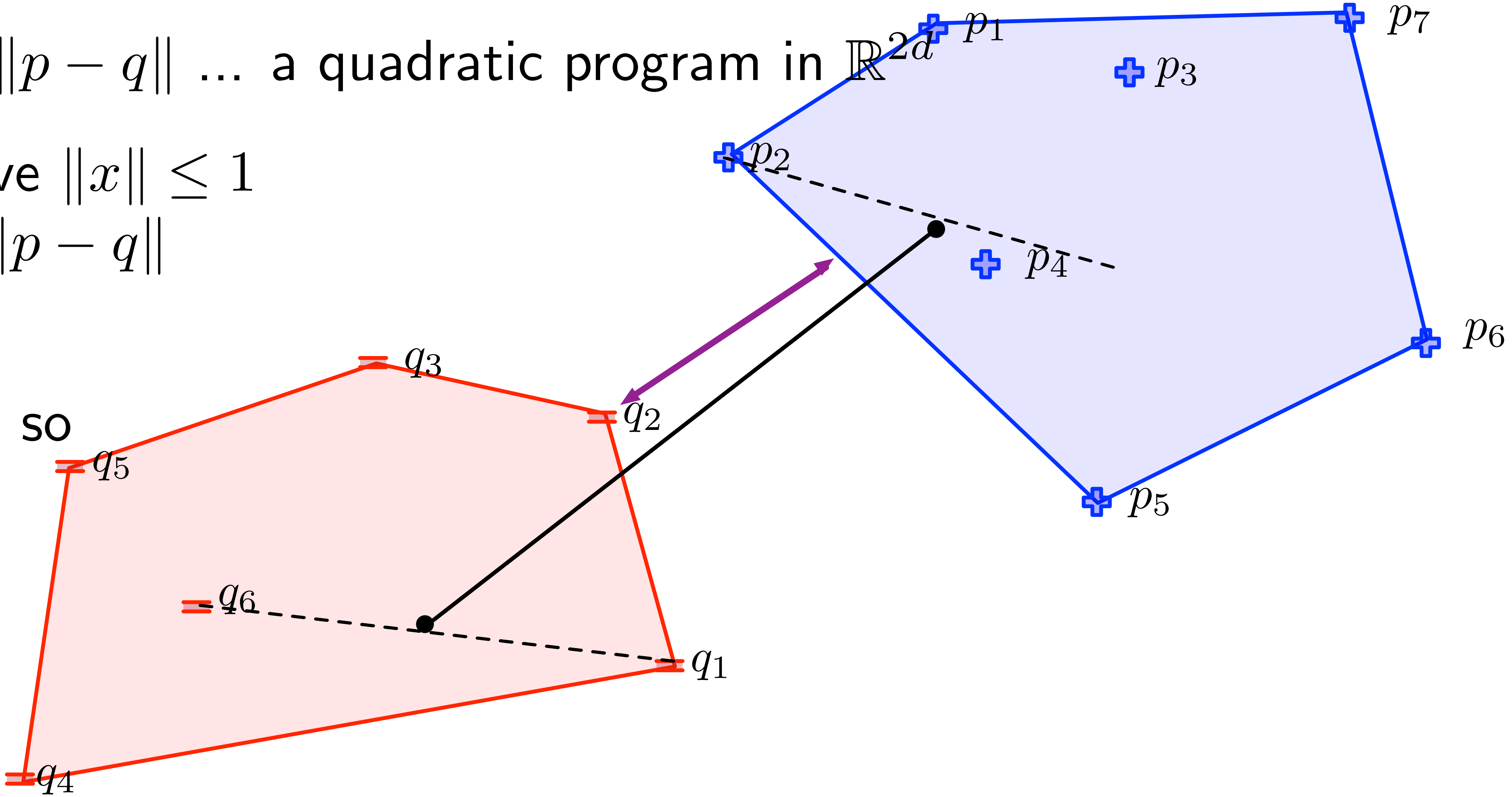
Polytope Distance

Define polytopes $P = \mathsf{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \mathsf{CH}(x \in X \mid \sigma(x) = -1)$

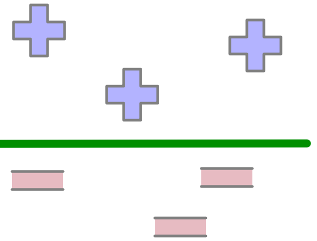
Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ … a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$

Iterate $1/(\varepsilon\gamma^2)$ steps,
find $\hat{p} \in P$ and $\hat{q} \in Q$ so
$(1 - \varepsilon)\|\hat{p} - \hat{q}\| \leq \gamma$



Polytope Distance



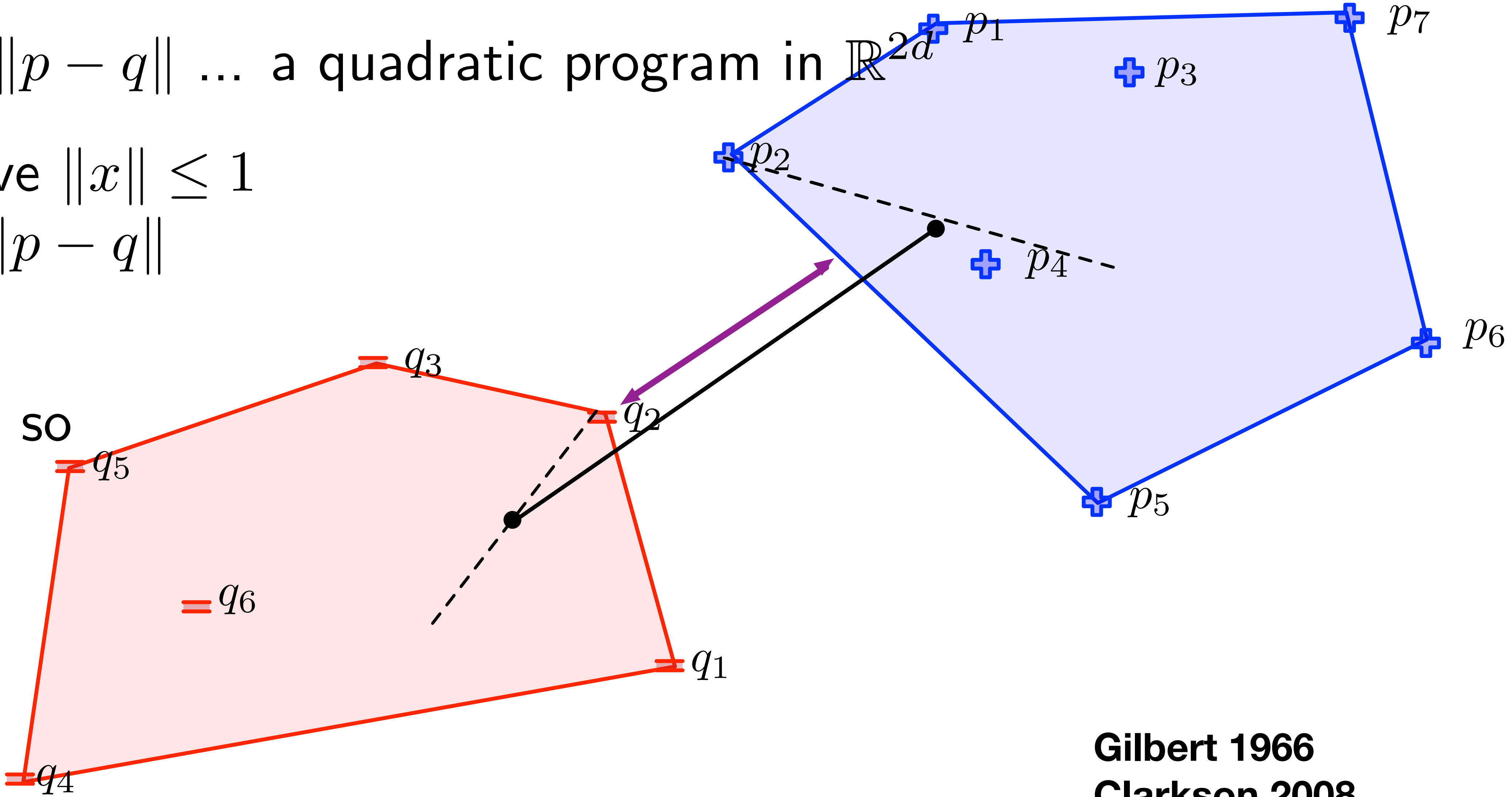
Define polytopes $P = \text{CH}(x \in X \mid \sigma(x) = +1)$
$Q = \text{CH}(x \in X \mid \sigma(x) = -1)$

Find $\arg\min_{p \in P, q \in Q} \|p - q\|$ ... a quadratic program in $\mathbb{R}^{2d}$

Assume all $x \in X$ have $\|x\| \leq 1$
Let $\gamma = \min_{p \in P, q \in Q} \|p - q\|$

Iterate $1/(\varepsilon\gamma^2)$ steps,
find $\hat{p} \in P$ and $\hat{q} \in Q$ so

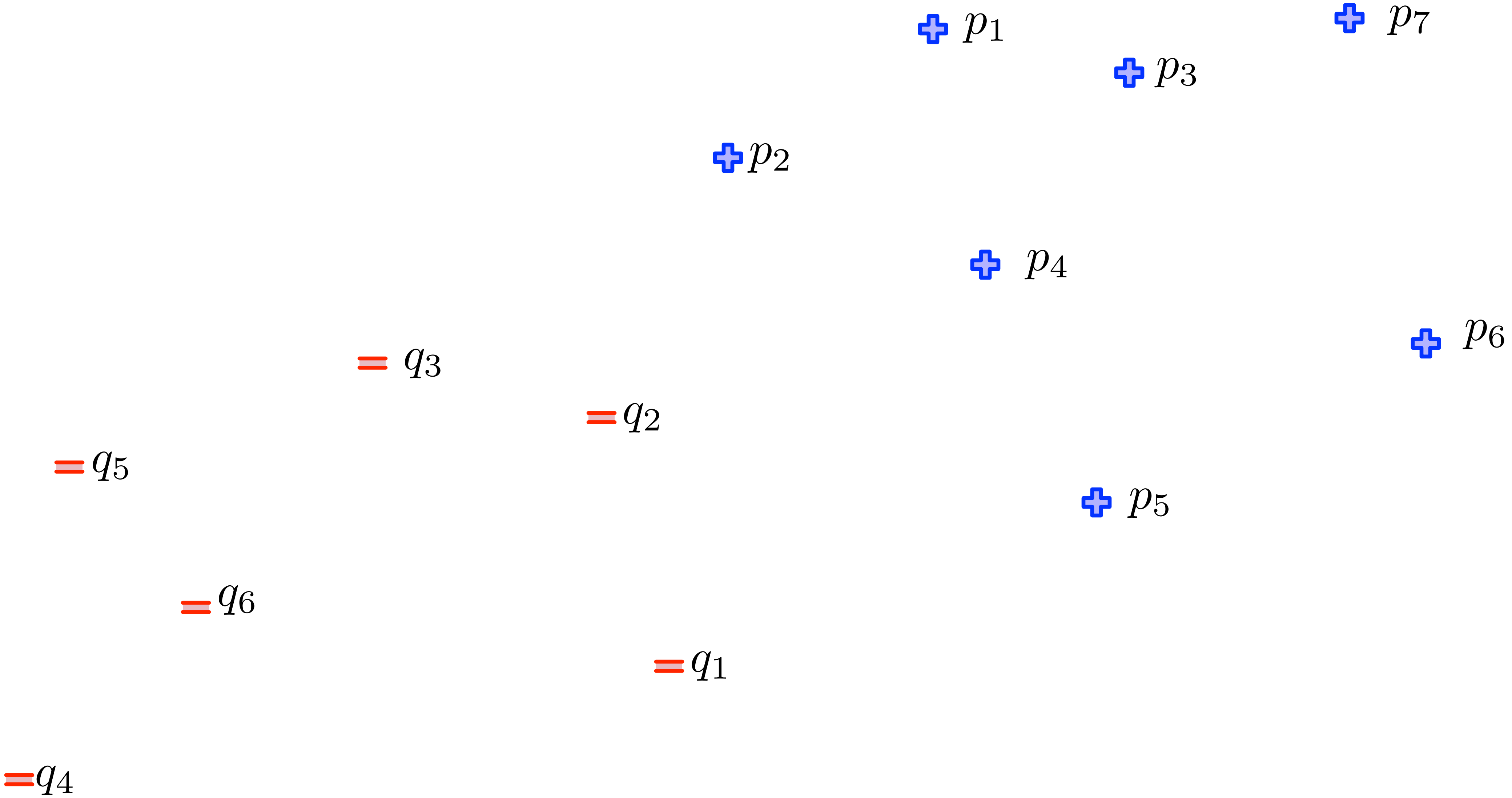
$$(1 - \varepsilon)\|\hat{p} - \hat{q}\| \leq \gamma$$



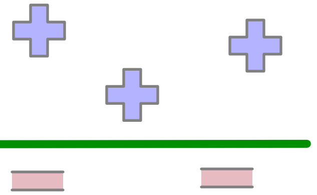
Gilbert 1966
Clarkson 2008
Gartner+Jaggi 2009

Force h through Origin

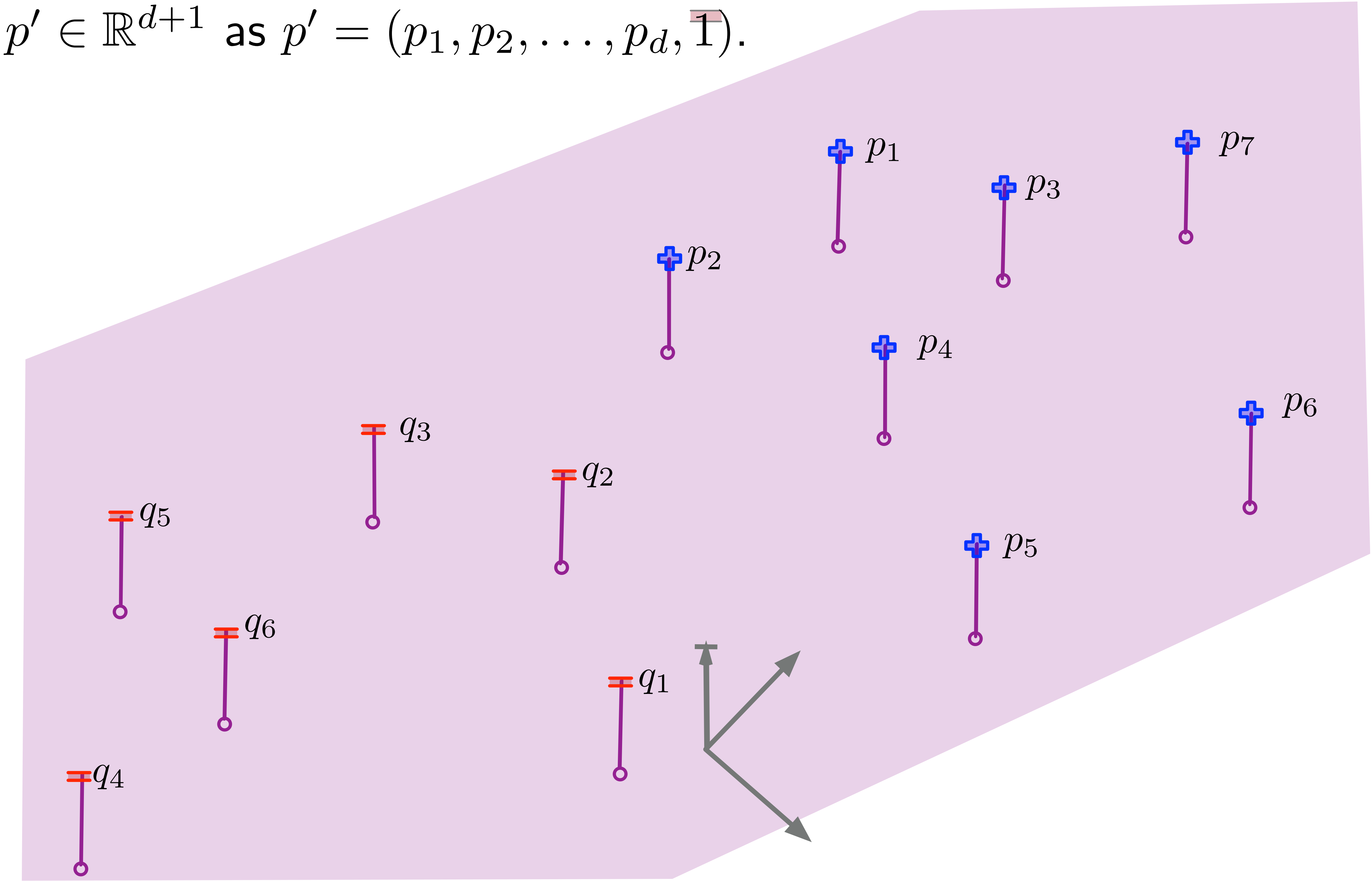
1: Map $p \in \mathbb{R}^d$ to $p' \in \mathbb{R}^{d+1}$ as $p' = (p_1, p_2, \ldots, p_d, 1)$.



Force h through Origin



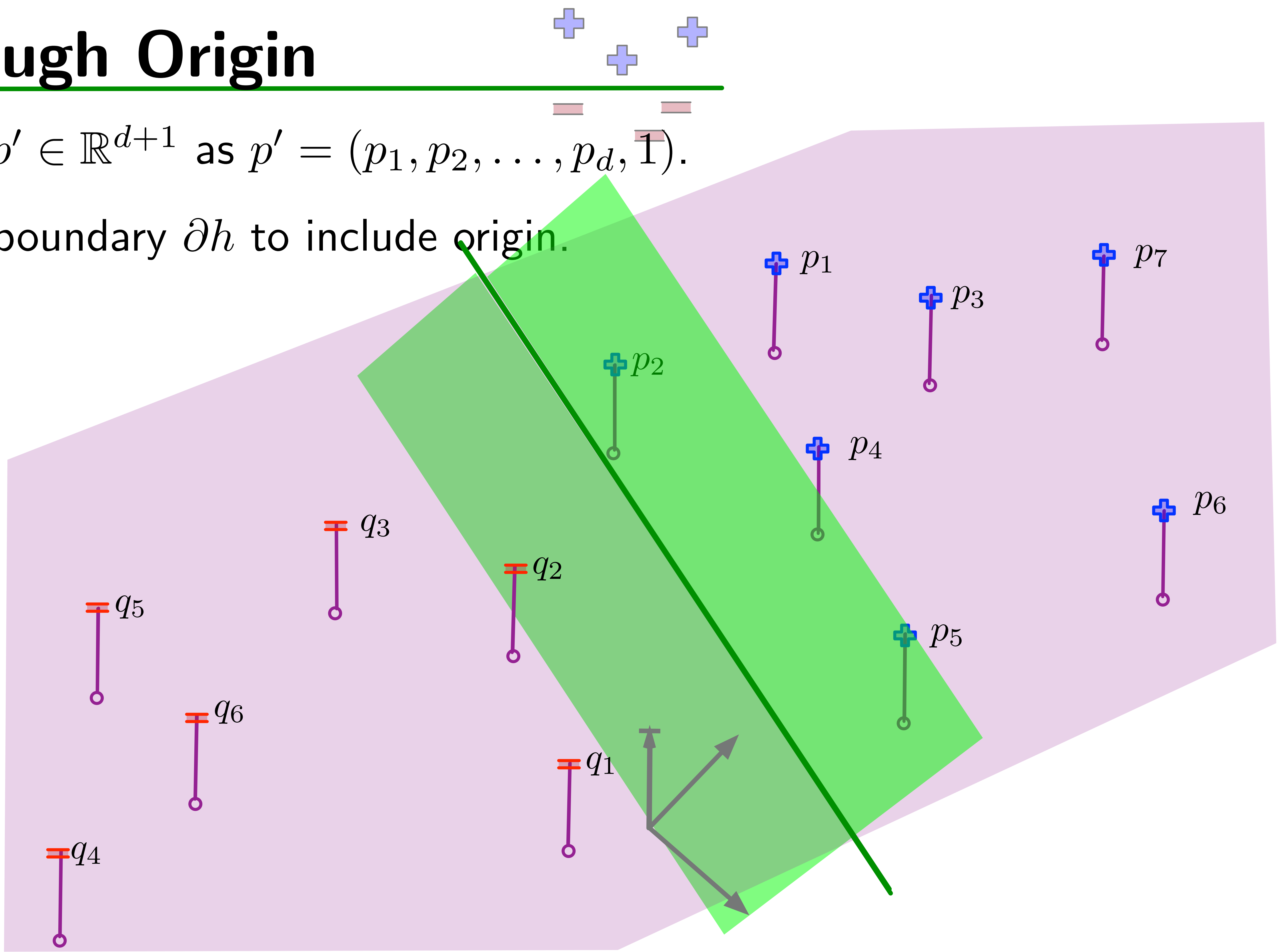
1: Map $p \in \mathbb{R}^d$ to $p' \in \mathbb{R}^{d+1}$ as $p' = (p_1, p_2, \ldots, p_d, 1)$.



Force h through Origin

1: Map $p \in \mathbb{R}^d$ to $p' \in \mathbb{R}^{d+1}$ as $p' = (p_1, p_2, \ldots, p_d, 1)$.

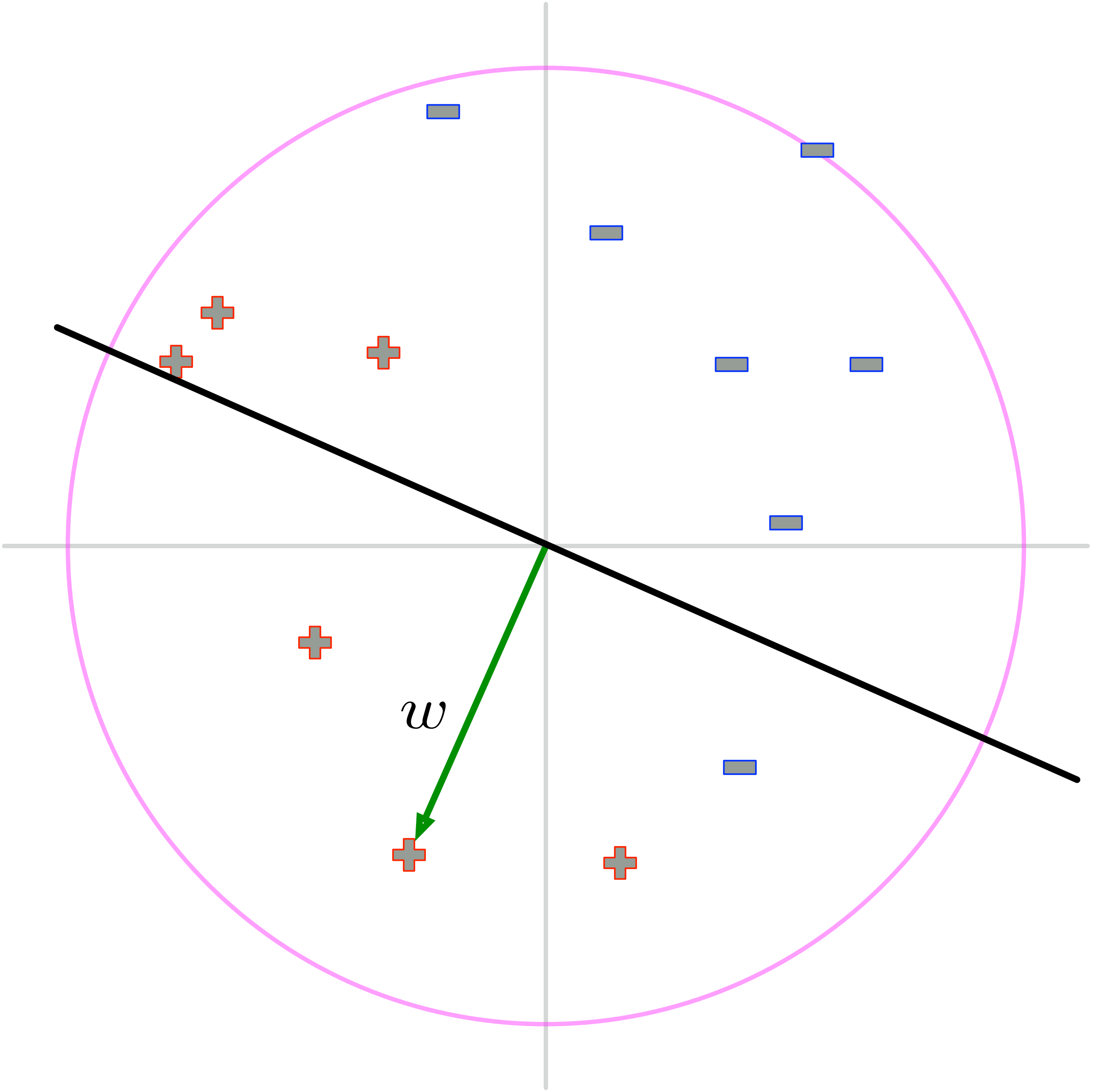
2: Force halfspace boundary ∂h to include origin.



Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$

(2) halfspace $h \in \mathcal{H}_0$ goes through origin

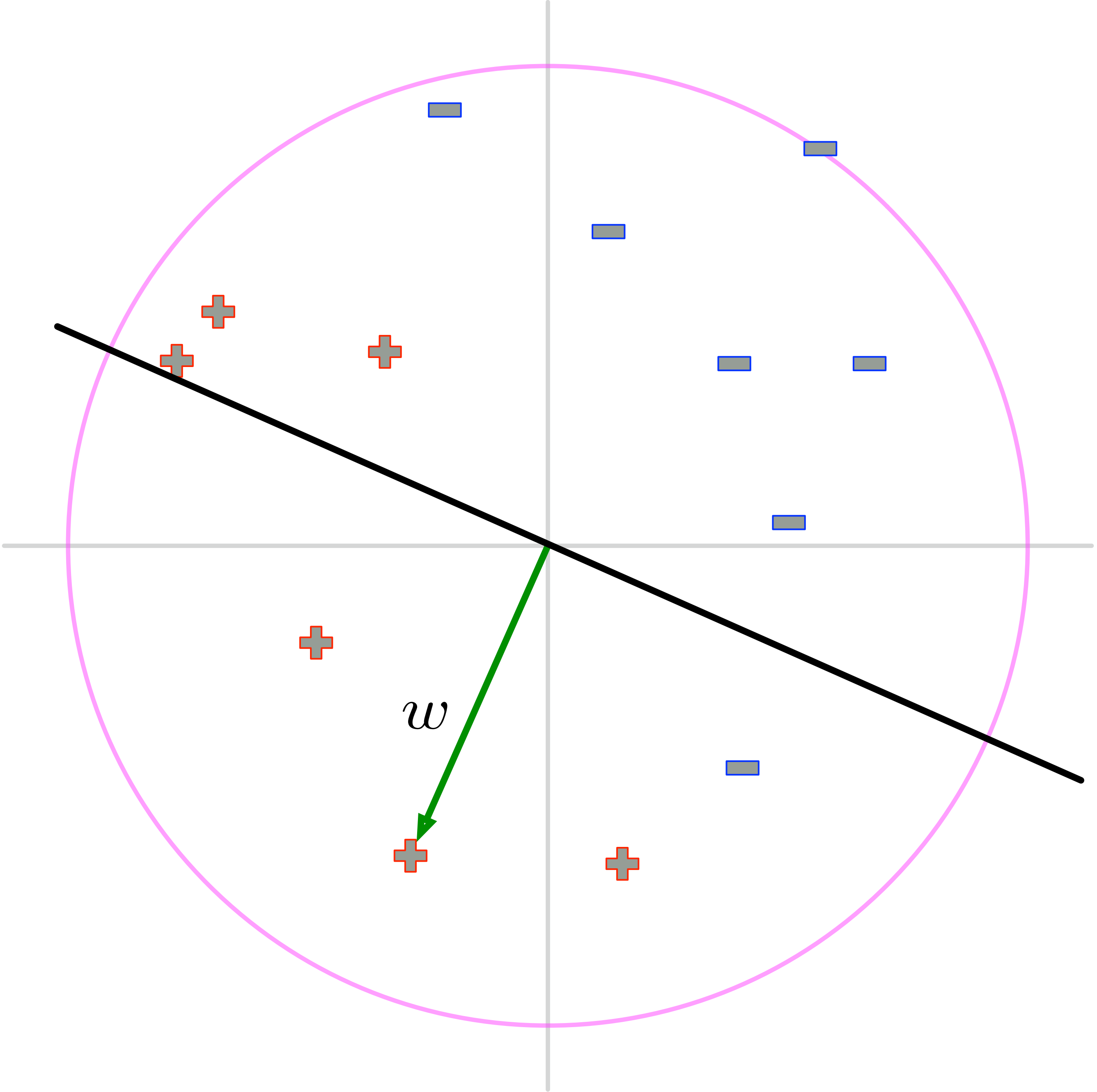


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm: **(Rosenblatt 1958)**

```
choose $w = \sigma(x)x$
**for** $i = 1$ **to** $1/\gamma^2$ steps
   $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
   $w = w + \sigma(x')x'$
```

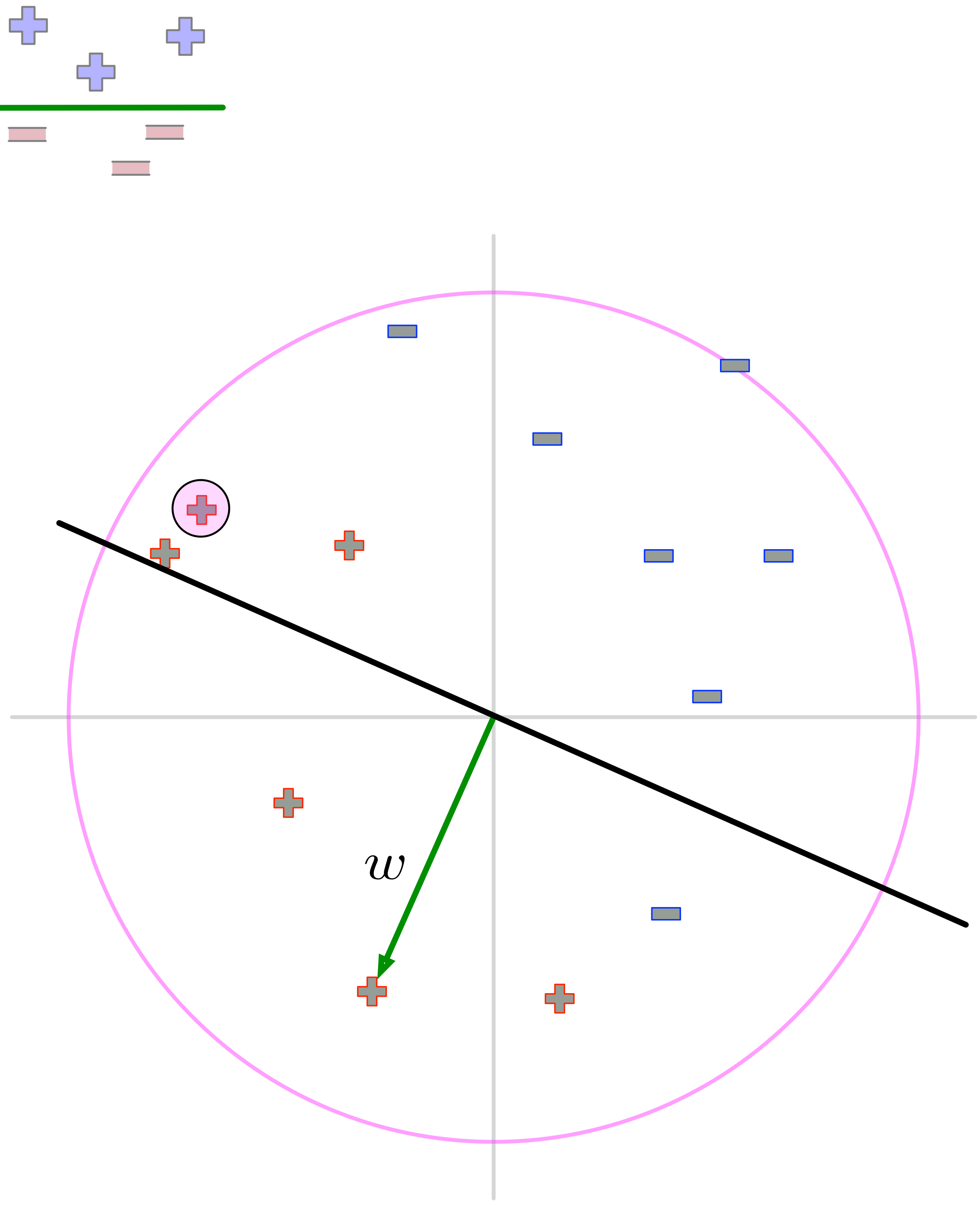


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
**for** $i = 1$ **to** $1/\gamma^2$ steps
    $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
    $w = w + \sigma(x')x'$
```

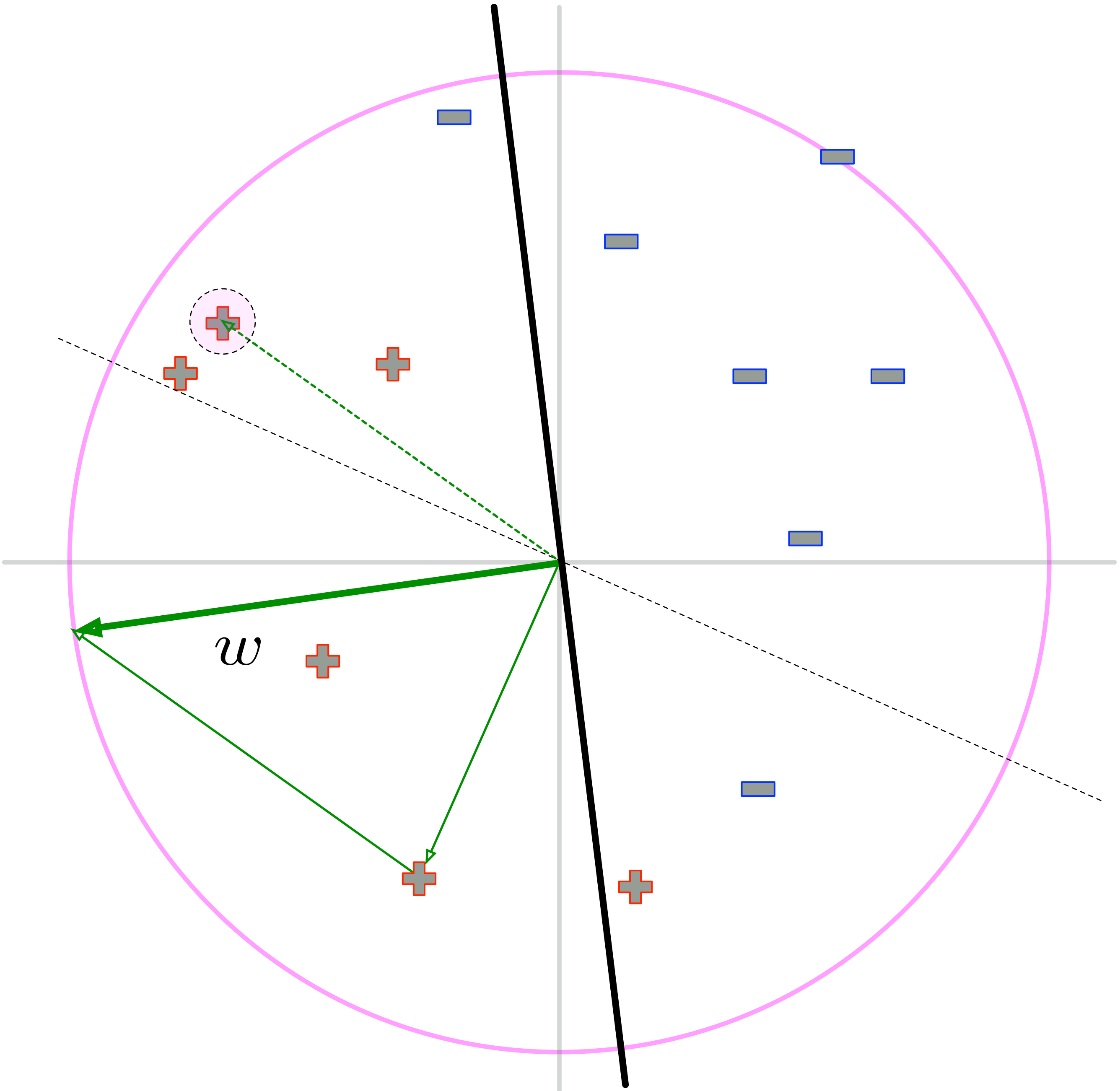


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
**for** $i = 1$ **to** $1/\gamma^2$ steps
  $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
  $w = w + \sigma(x')x'$
```

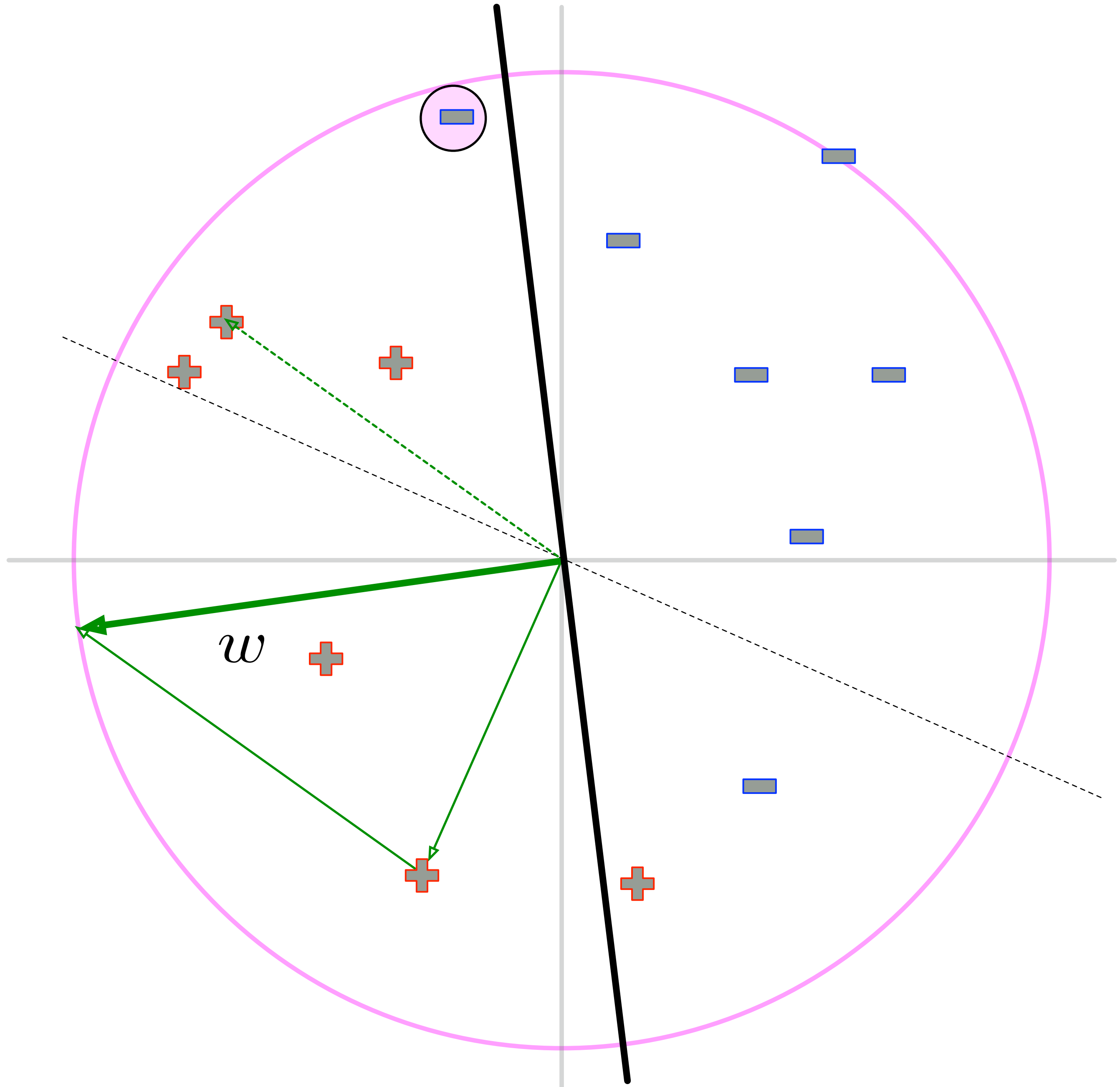


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\gamma^2$ steps
  $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
  $w = w + \sigma(x')x'$
```

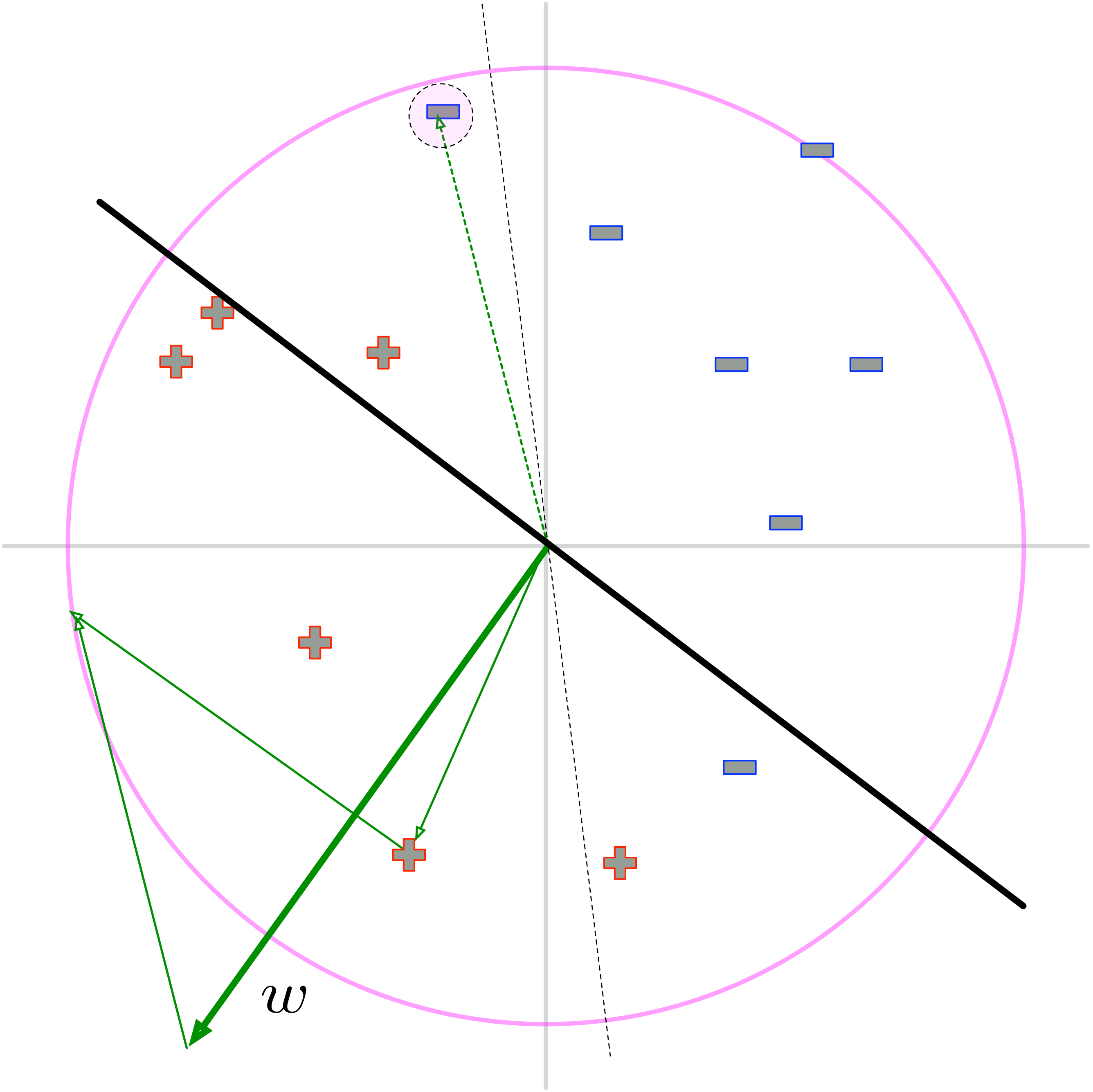


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
**for** $i = 1$ **to** $1/\gamma^2$ steps
  $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
  $w = w + \sigma(x')x'$
```

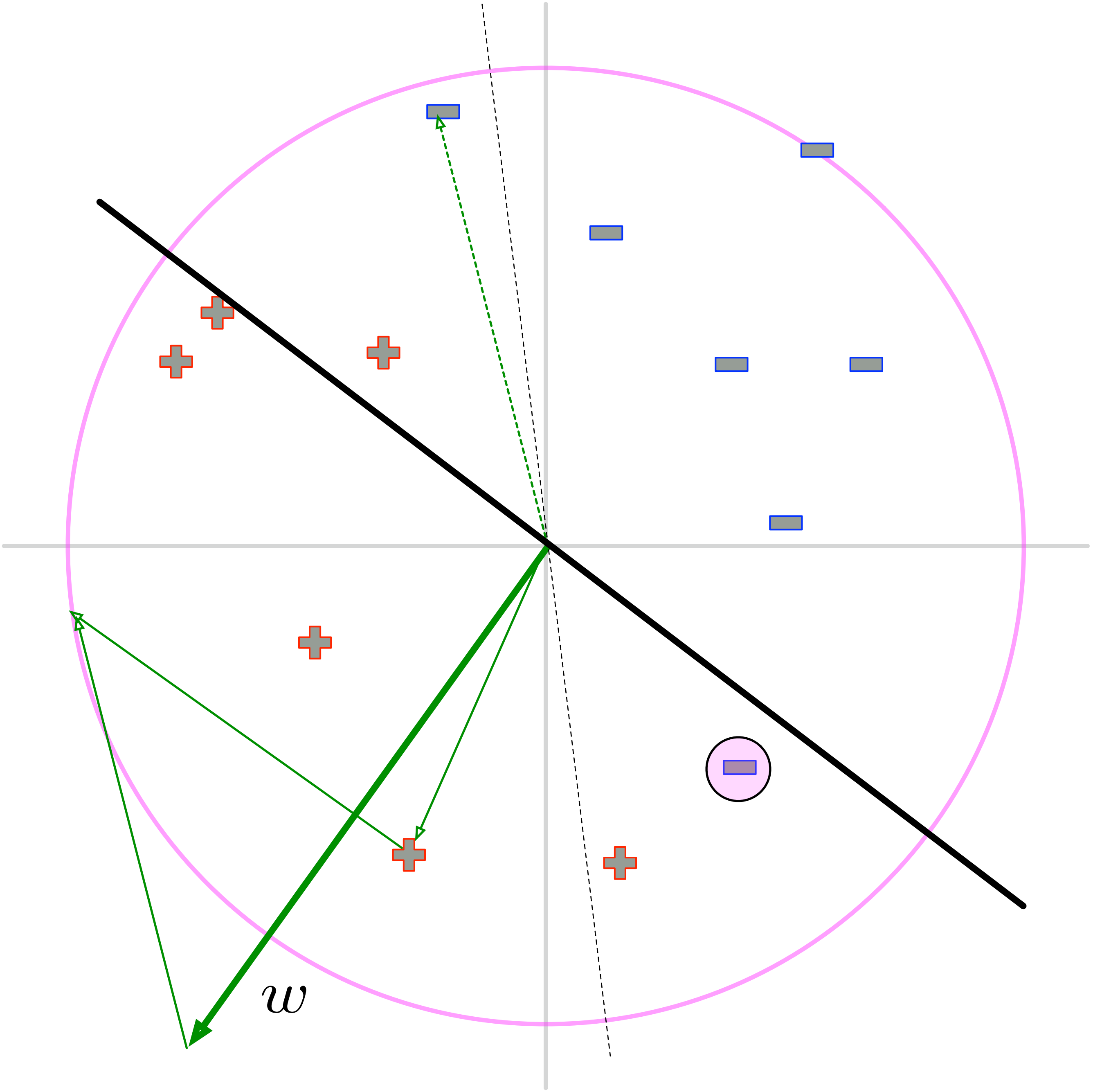


Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\gamma^2$ steps
  $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
  $w = w + \sigma(x')x'$
```

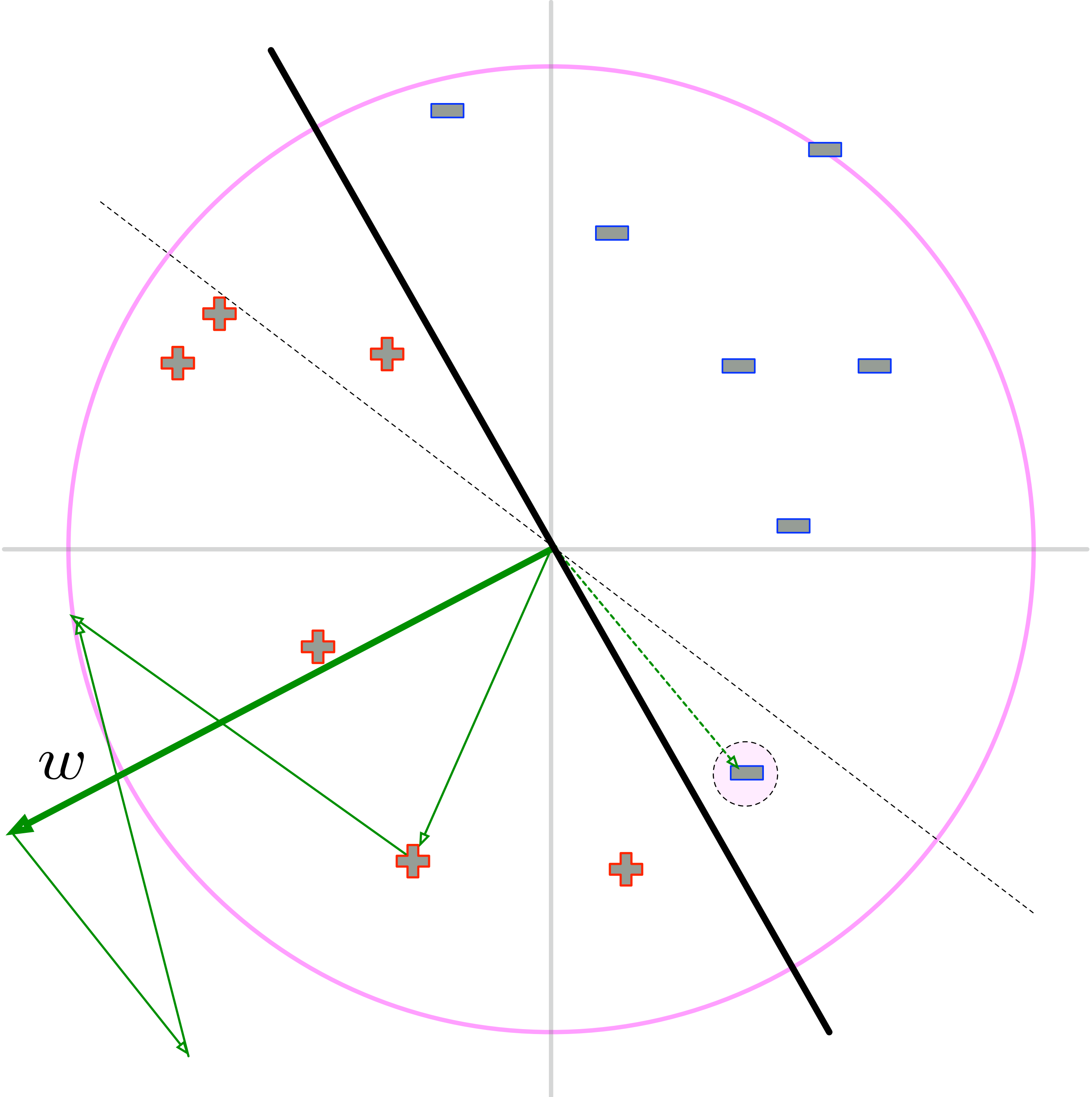


Perceptron Algorithm

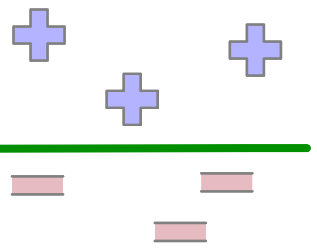
Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\gamma^2$ steps
    $x' =$ any $x \in X$ s.t. $\langle \sigma(x)x, w \rangle < 0$
    $w = w + \sigma(x')x'$
```



Perceptron Algorithm

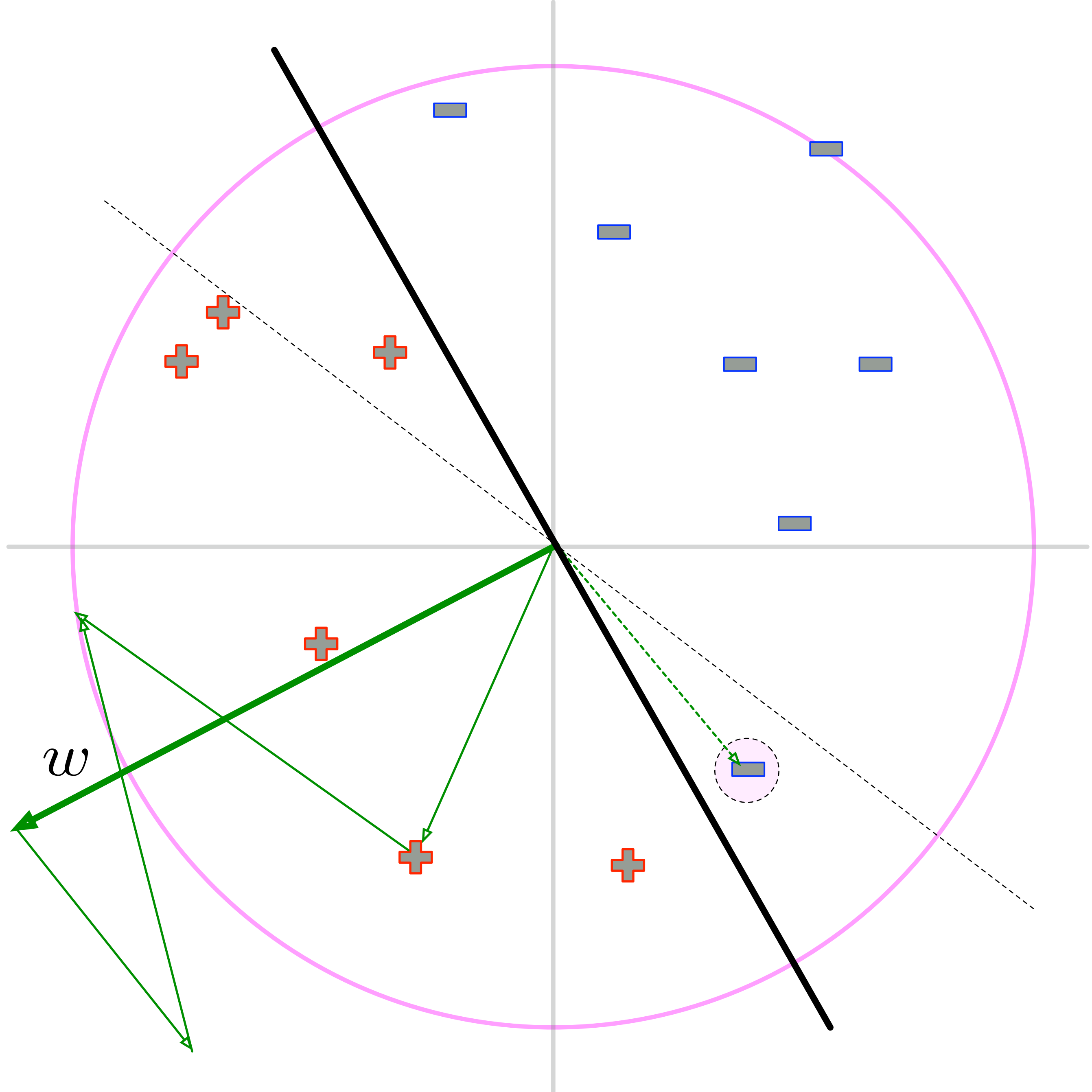


Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$

(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\varepsilon\gamma^2$ steps
  $x' = \arg\min\langle\sigma(x)x, w\rangle$
  $w = w + \sigma(x')x'$
```



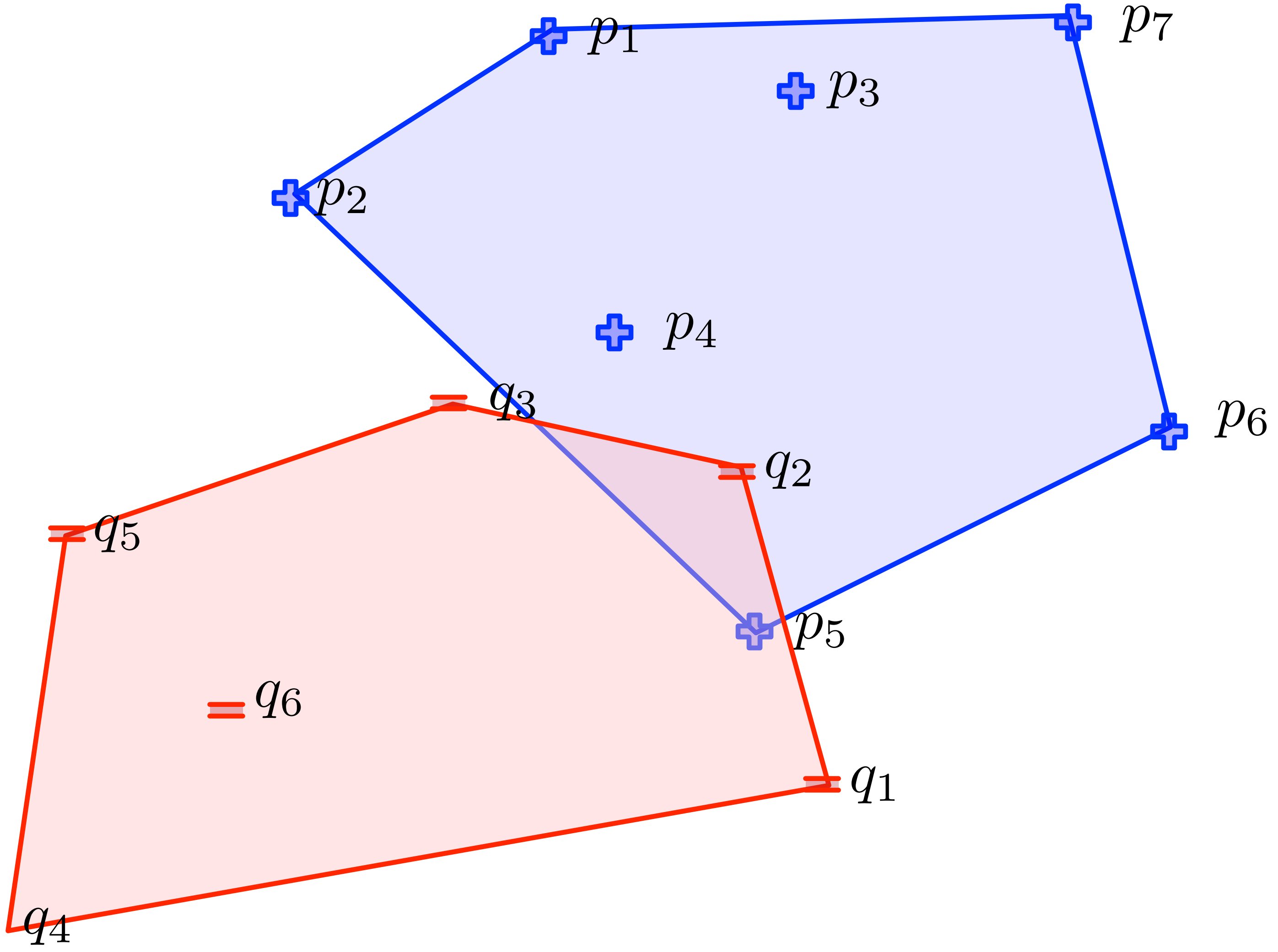
Perceptron Algorithm

Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\varepsilon\gamma^2$ steps
    $x' = \arg\min \langle \sigma(x)x, w \rangle$
    $w = w + \sigma(x')x'$
```

What to do if the polytopes intersect?
it is ``non-separable" ?



Perceptron Algorithm

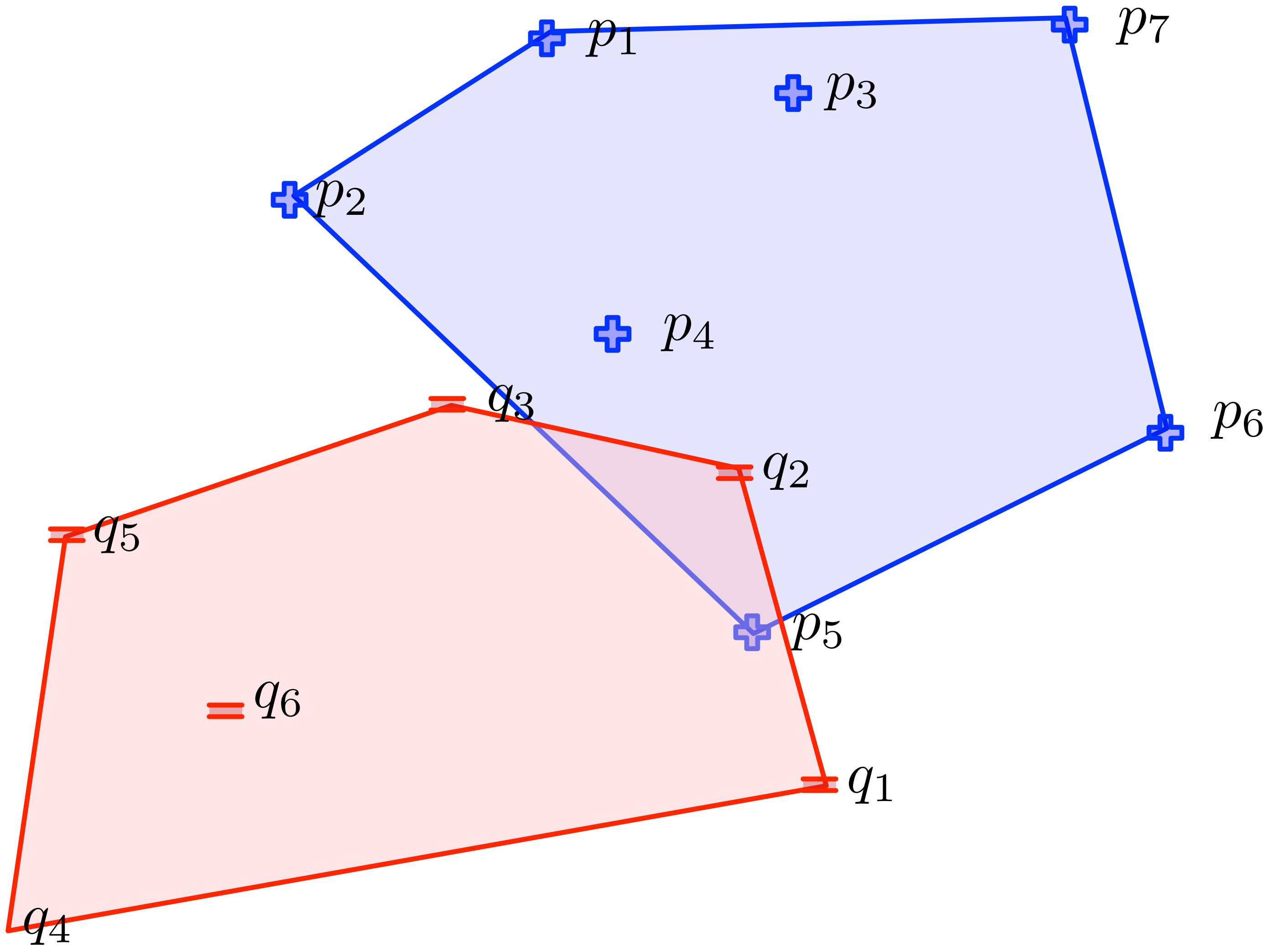
Assume (1) $x \in X \subset \mathbb{R}^d$ has $\|x\| \leq 1$
(2) halfspace $h \in \mathcal{H}_0$ goes through origin

Algorithm:

```
choose $w = \sigma(x)x$
for $i = 1$ to $1/\varepsilon\gamma^2$ steps
  $x' = \arg\min \langle \sigma(x)x, w \rangle$
  $w = w + \sigma(x')x'$
```

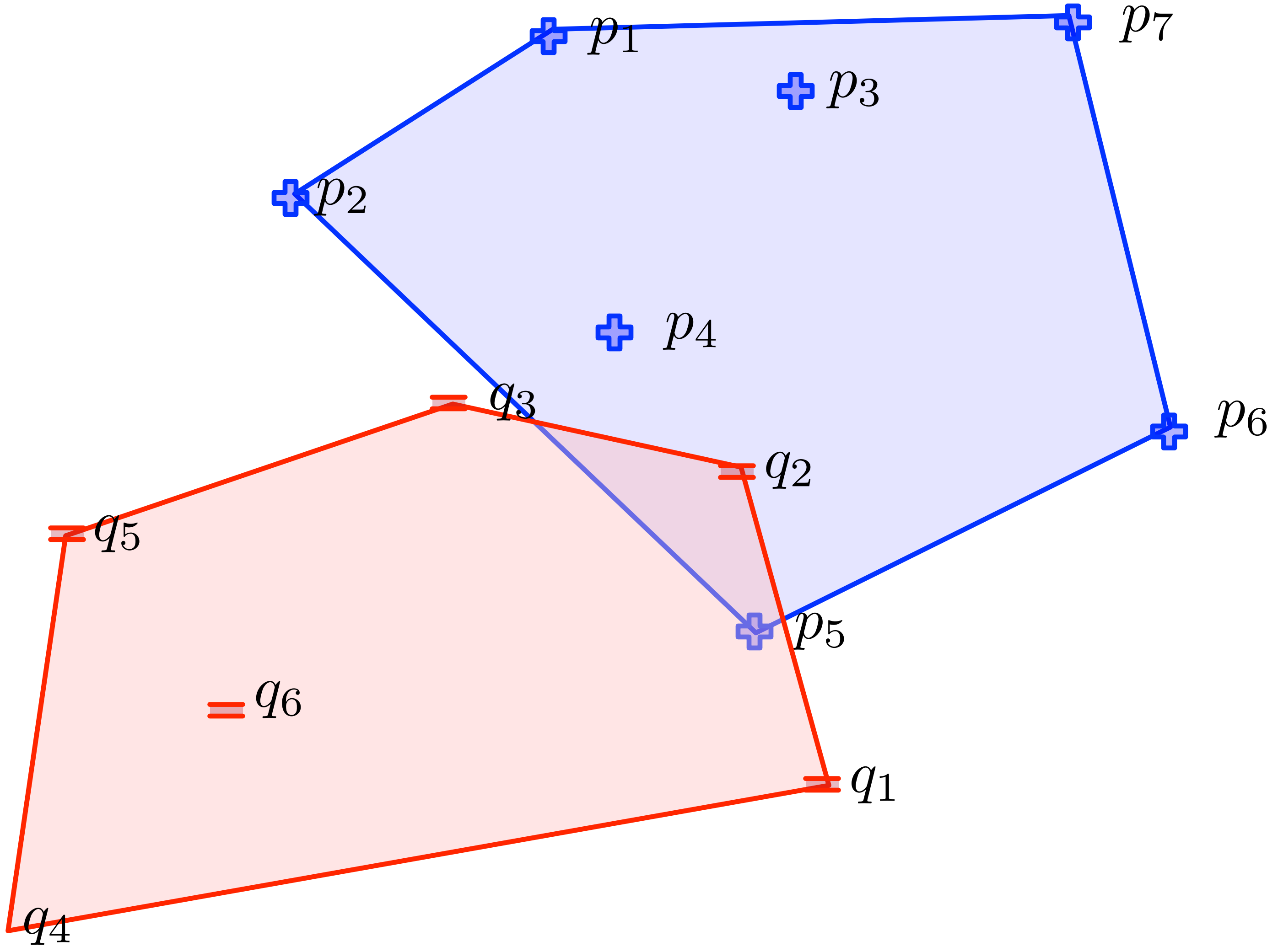
What to do if the polytopes intersect?
it is ``non-separable" ?

(1) kernels
(2) penalize it



Kernel Perceptron Algorithm

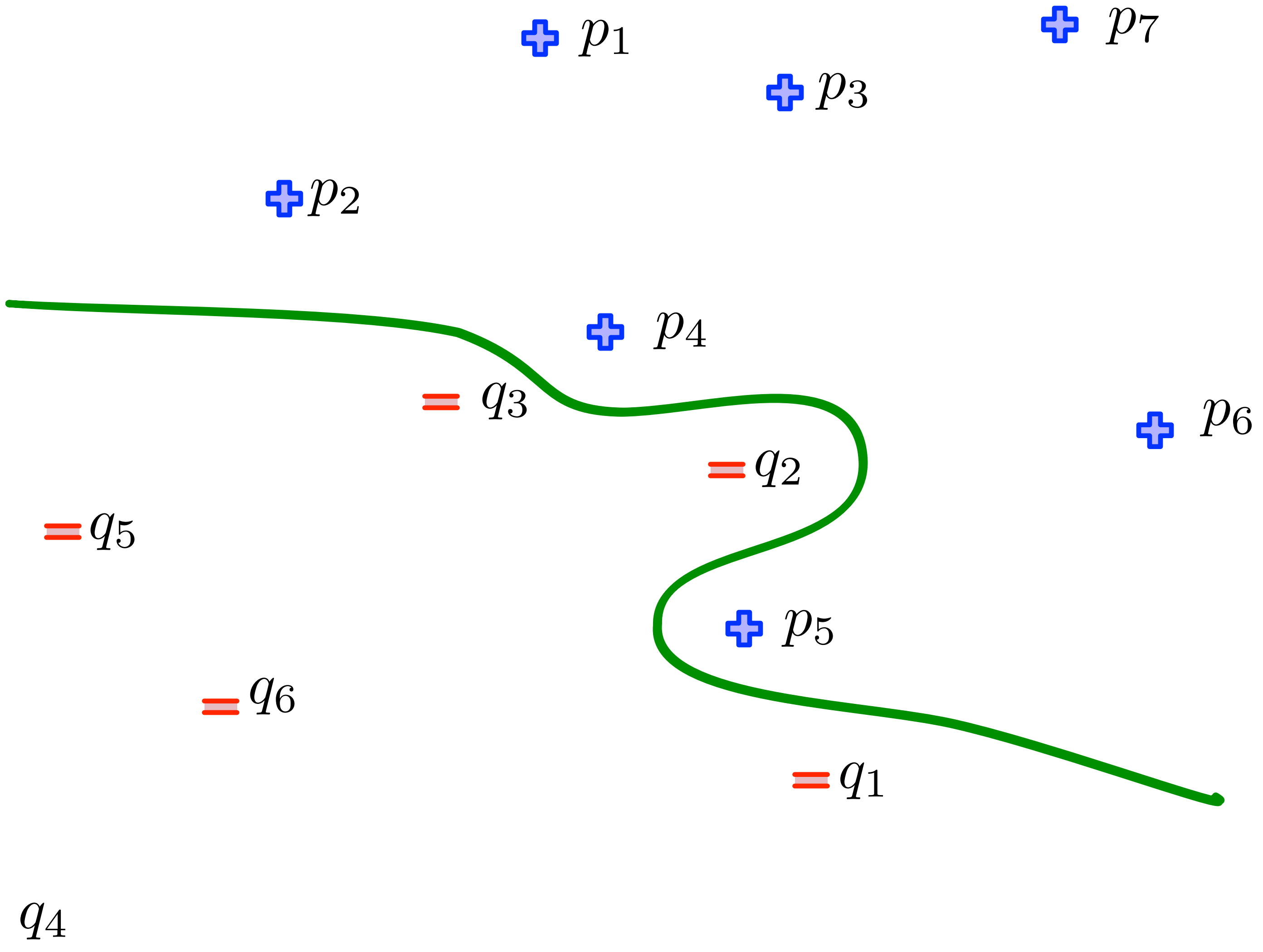
```
Algorithm: (Kernel SVM)
    choose $w = \sigma(x)x$
    **for** $i = 1$ **to** $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\langle x, w\rangle$
        $w = w + \sigma(x')x'$
```



Kernel Perceptron Algorithm

```
Algorithm: (Kernel SVM)
    choose $w = \sigma(x)x$
    **for** $i = 1$ **to** $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x) K(x, w)$
        $w = w \oplus \sigma(x')x'$
```

$$K(x,p) = \exp(-\|x-p\|^2)$$
$$K(x,p) = \exp(-\|x-p\|)$$
$$K(x,p) = (1 + \langle x, p\rangle)^r$$



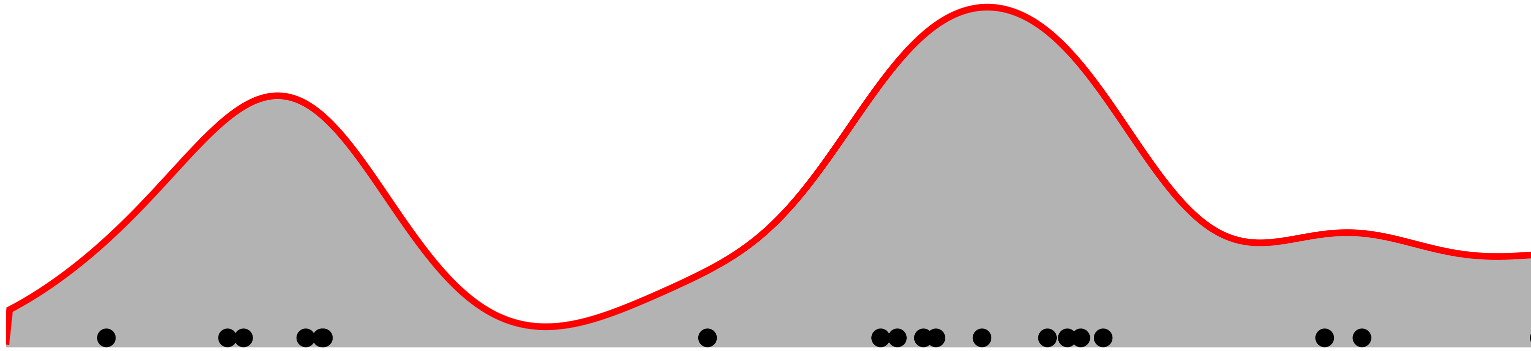
Kernel Perceptron Algorithm

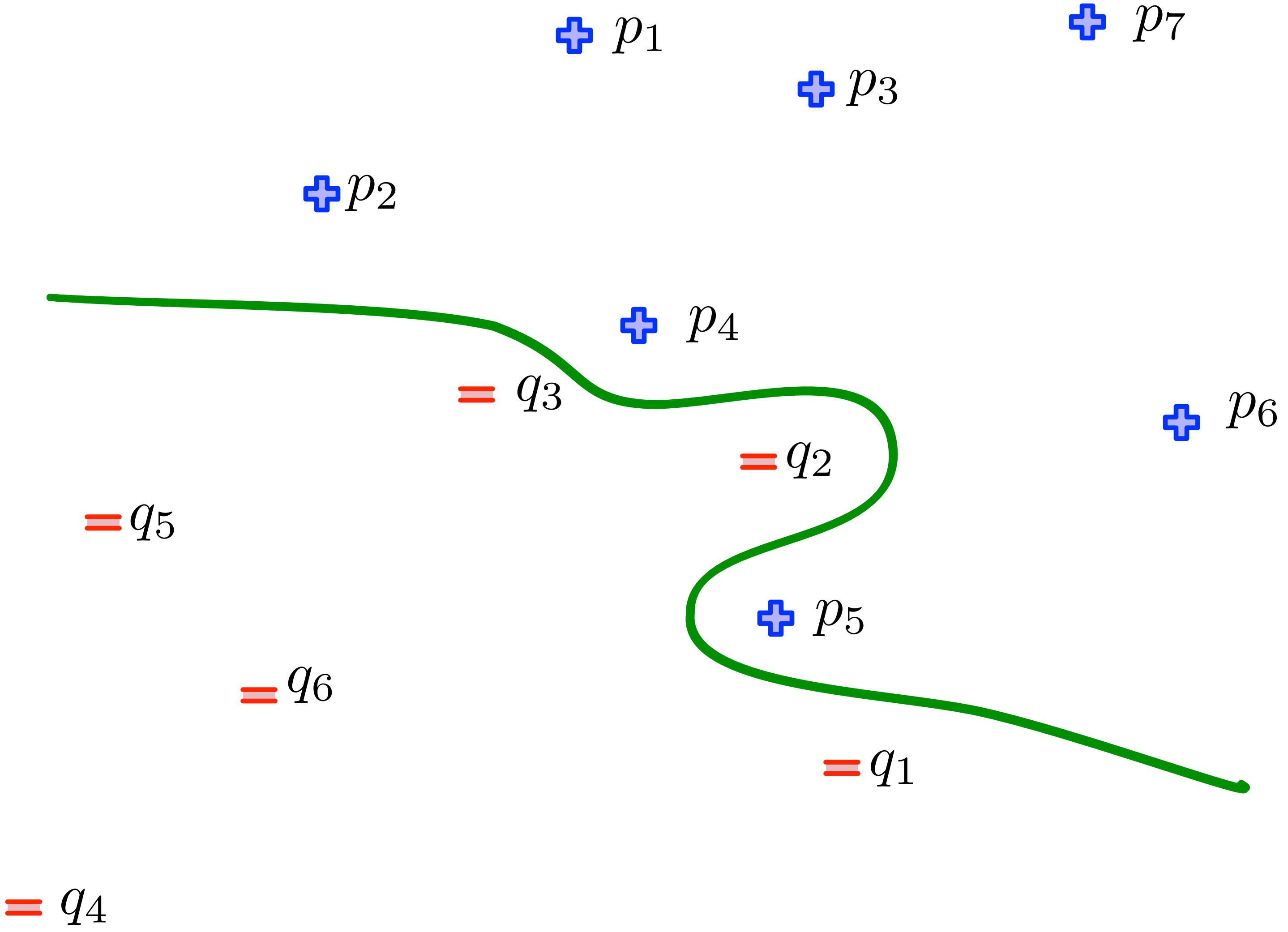
```
Algorithm: (Kernel SVM)
    choose $w = \sigma(x)x$
    for $i = 1$ to $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\text{KDE}_W(x)$
        $\text{KDE}_W = \text{KDE}_W + \sigma(x')K(x', \cdot)$
```

$$K(x,p) = \exp(-\|x-p\|^2)$$
$$K(x,p) = \exp(-\|x-p\|)$$
$$K(x,p) = (1 + \langle x, p\rangle)^r$$

kernel density estimate:

$$\text{KDE}_W(x) = \frac{1}{|W|}\sum_{w \in W} K(w,x)$$





Kernel Perceptron Algorithm

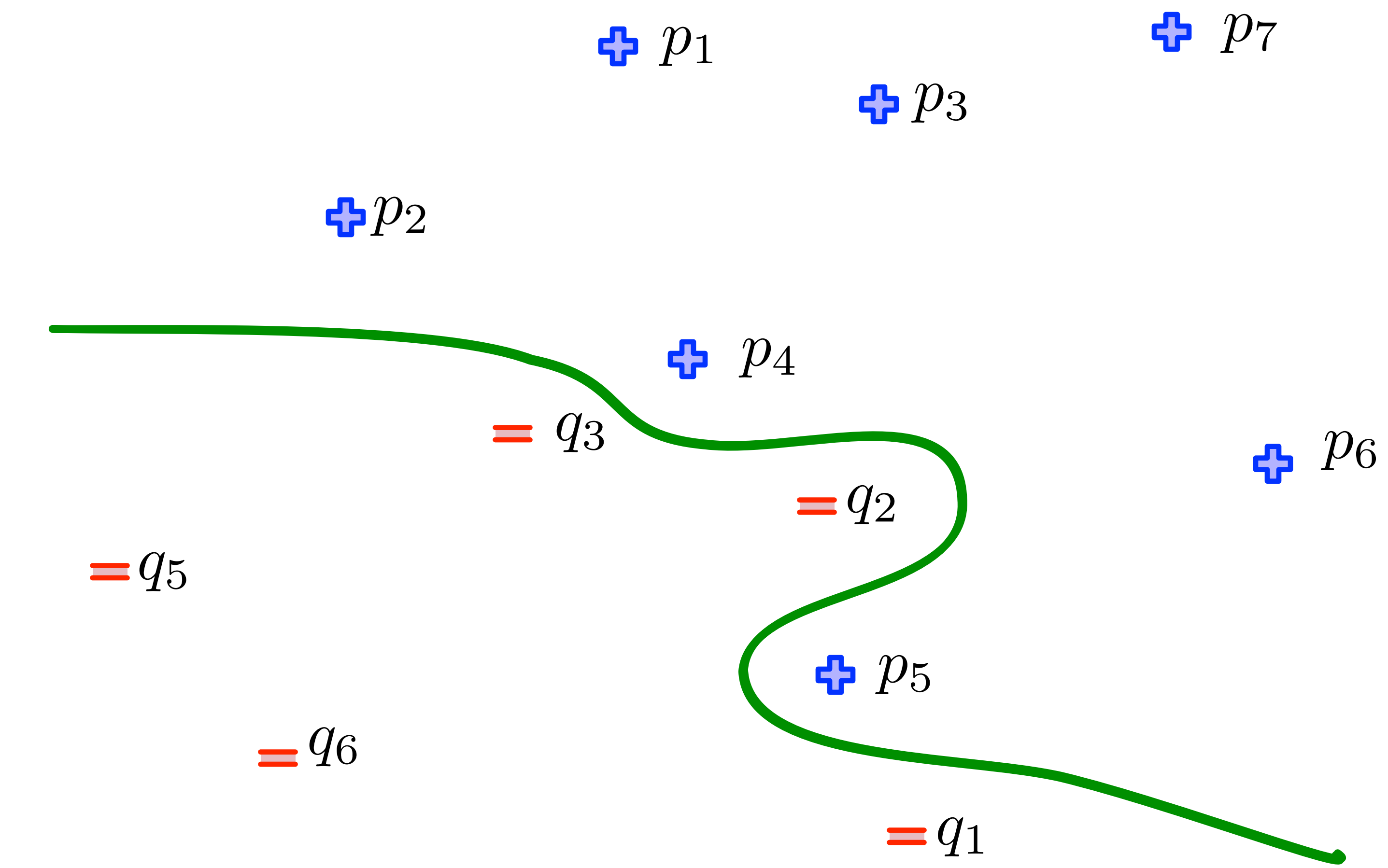
Algorithm: (Kernel SVM)

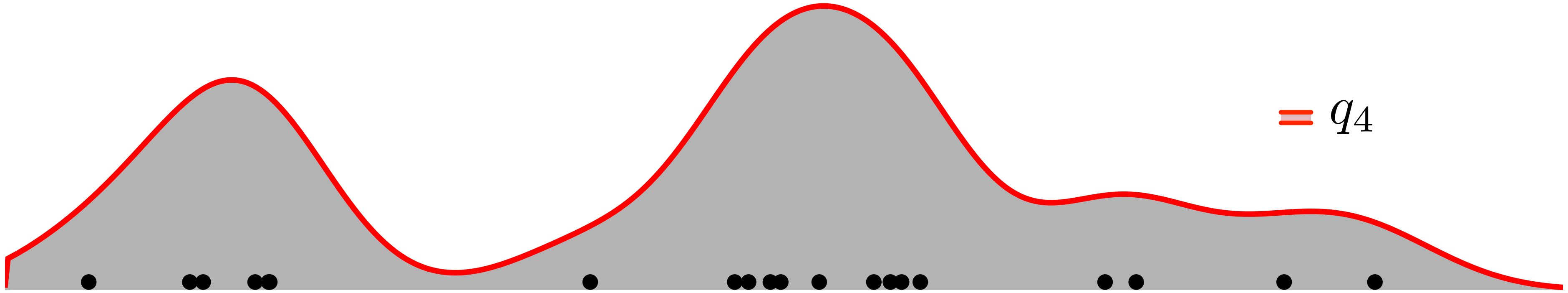
```
choose $w = \sigma(x)x$
**for** $i = 1$ **to** $1/\varepsilon\gamma^2$ steps
    $x' = \arg\min \sigma(x)\text{KDE}_W(x)$
    $\text{KDE}_W = \text{KDE}_W + \sigma(x')K(x', \cdot)$
```

$$K(x,p) = \exp(-\|x-p\|^2)$$
$$K(x,p) = \exp(-\|x-p\|)$$
$$K(x,p) = (1 + \langle x, p \rangle)^r$$

kernel density estimate:

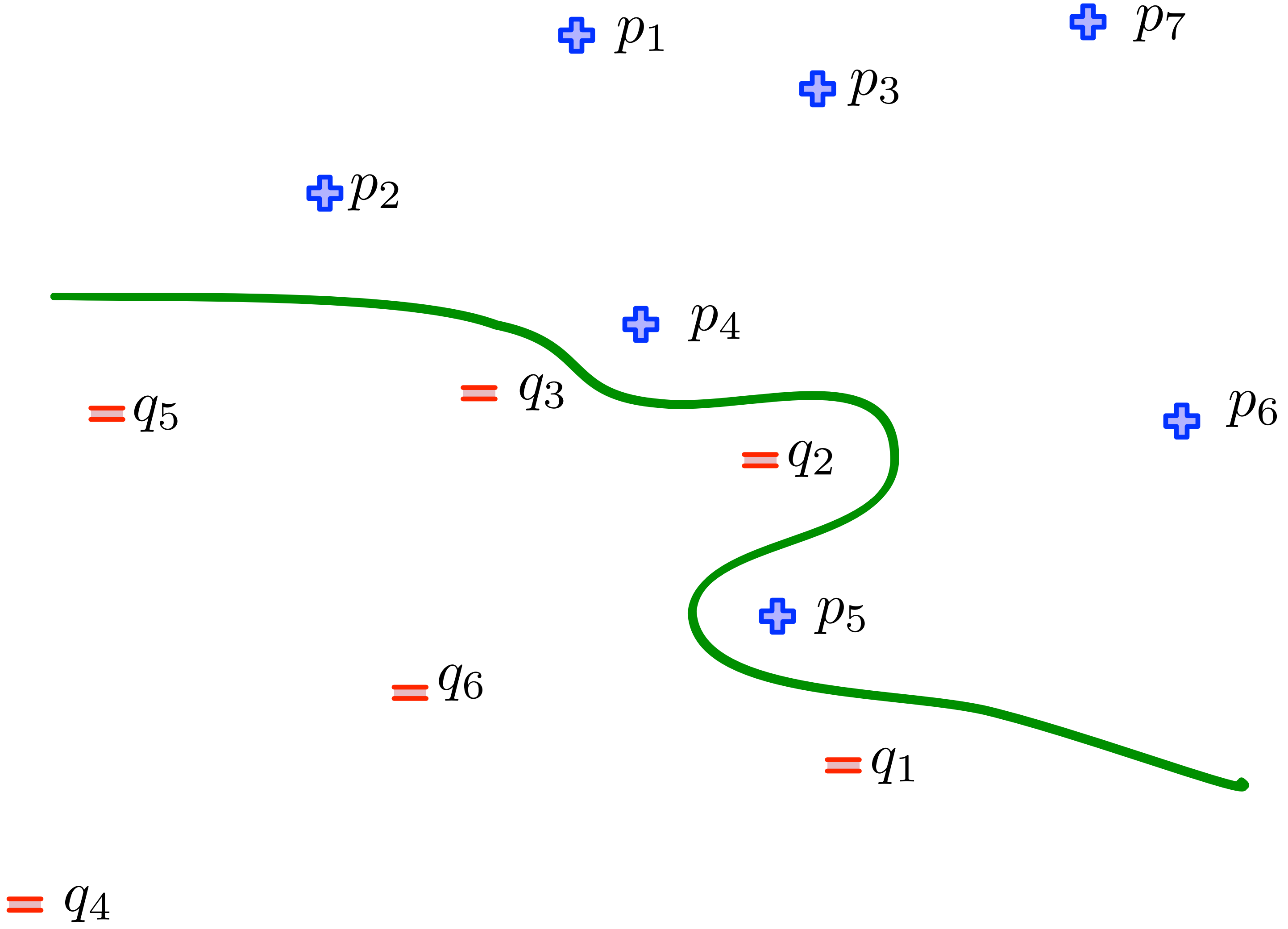
$$\text{KDE}_W(x) = \cancel{\frac{1}{|W|}} \sum_{w\in W} K(w,x)$$



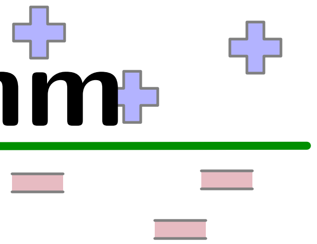


Sketched Kernel SVM Algorithm

```
Algorithm:
Define mapping $\phi : X \to \mathbb{R}^\rho$
    choose $w = \sigma(x)\phi(x)$
    for $i = 1$ to $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\langle\phi(x), w\rangle$
        $w = w + \sigma(x')\phi(x')$
```



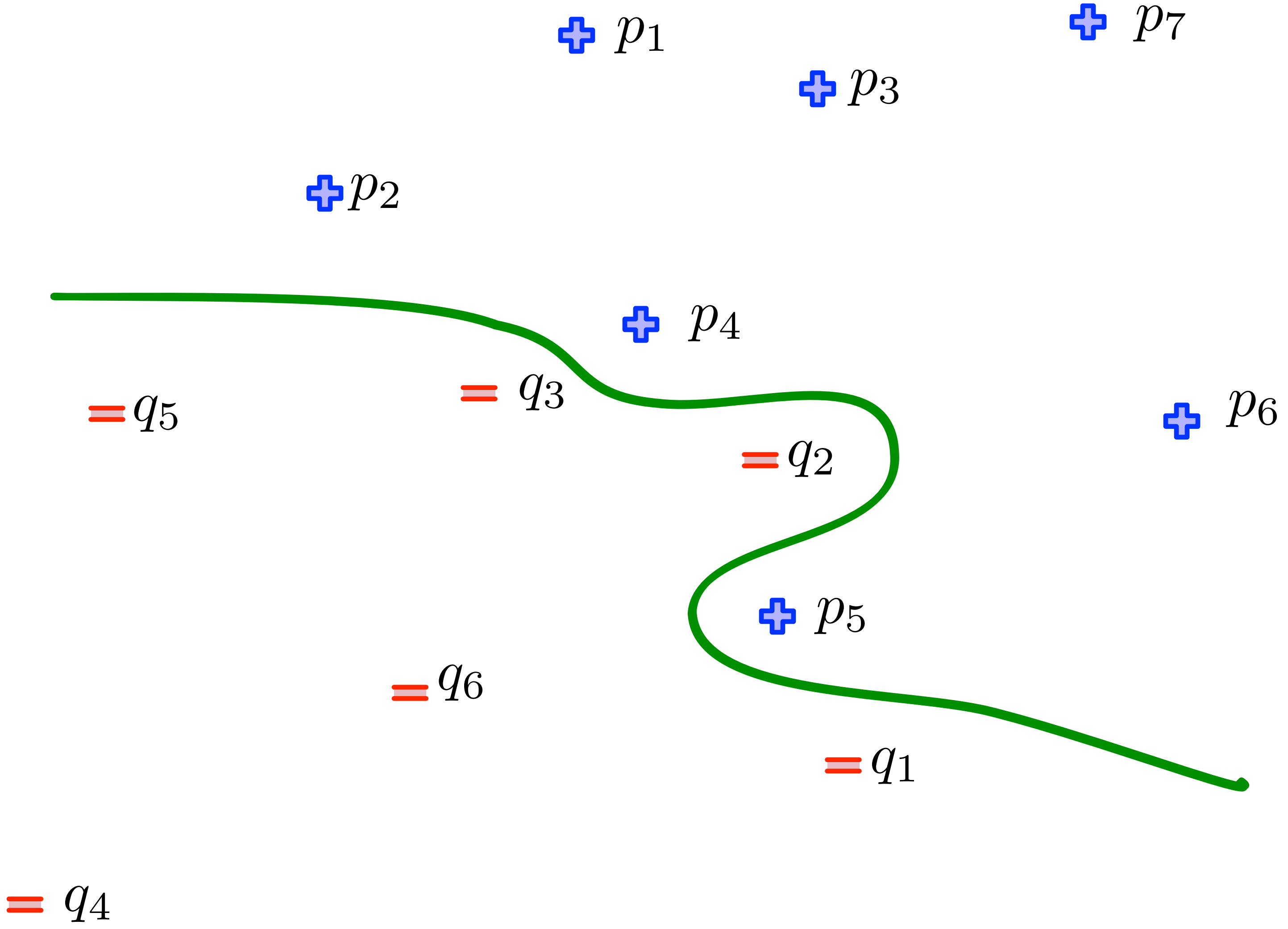
Sketched Kernel SVM Algorithm



Algorithm:

```
Define mapping $\phi : X \to \mathbb{R}^\rho$
    choose $w = \sigma(x)\phi(x)$
    for $i = 1$ to $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\langle \phi(x), w \rangle$
        $w = w + \sigma(x')\phi(x')$
```

$$K(x, p) = (1 + \langle x, p \rangle)^r \in \mathbb{R}^{O(rd)}$$



Sketched Kernel SVM Algorithm

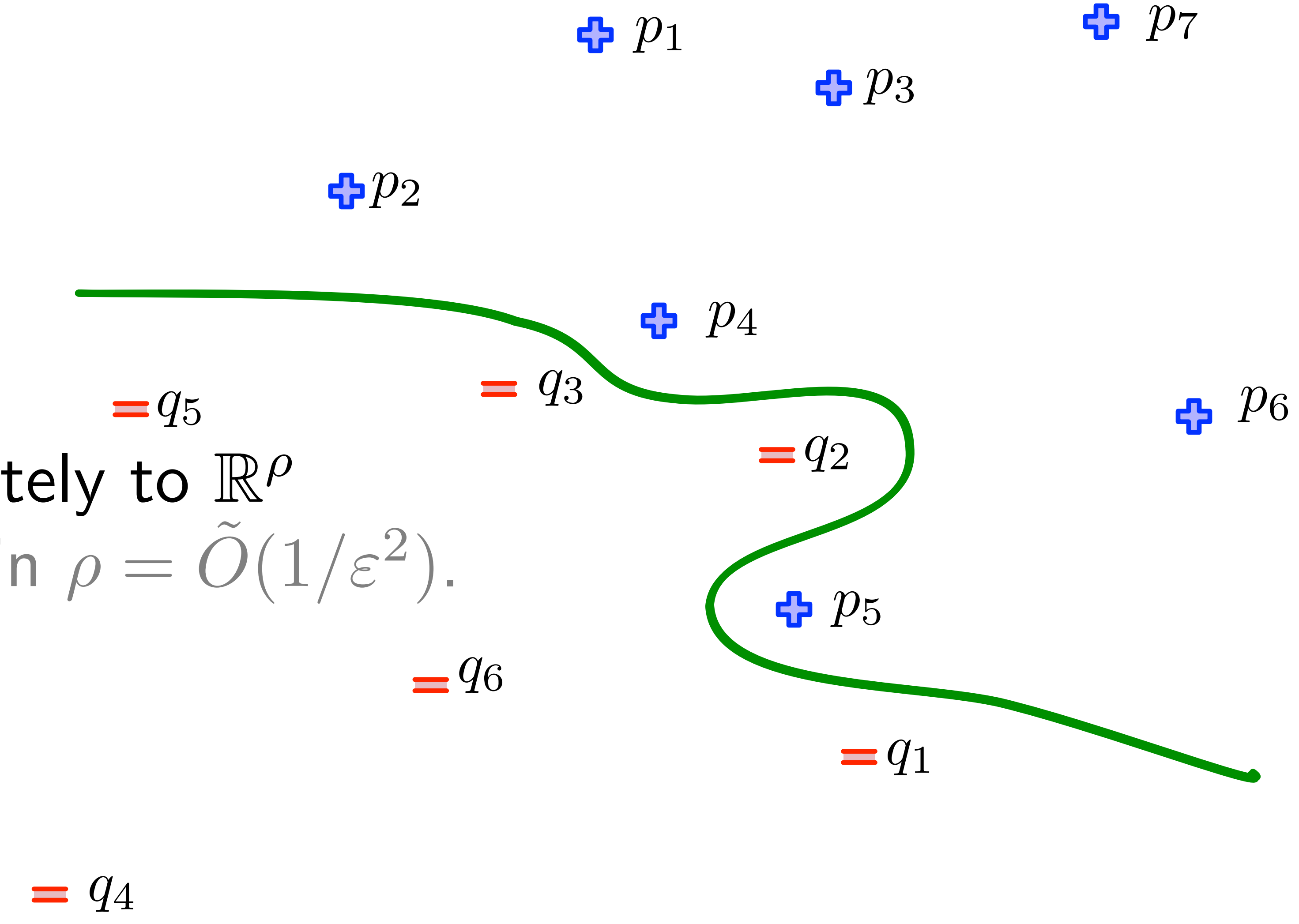
Algorithm:

```
Define mapping $\phi : X \to \mathbb{R}^\rho$
    choose $w = \sigma(x)\phi(x)$
    for $i = 1$ to $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\langle \phi(x), w\rangle$
        $w = w + \sigma(x')\phi(x')$
```

$K(x, p) = (1 + \langle x, p\rangle)^r \in \mathbb{R}^{O(rd)}$

$K(x, p) = \exp(-\|x - p\|^2)$ maps approximately to $\mathbb{R}^\rho$

e.g., Rahimi+Recht [NeurIPS07] additive ε in $\rho = \tilde{O}(1/\varepsilon^2)$.



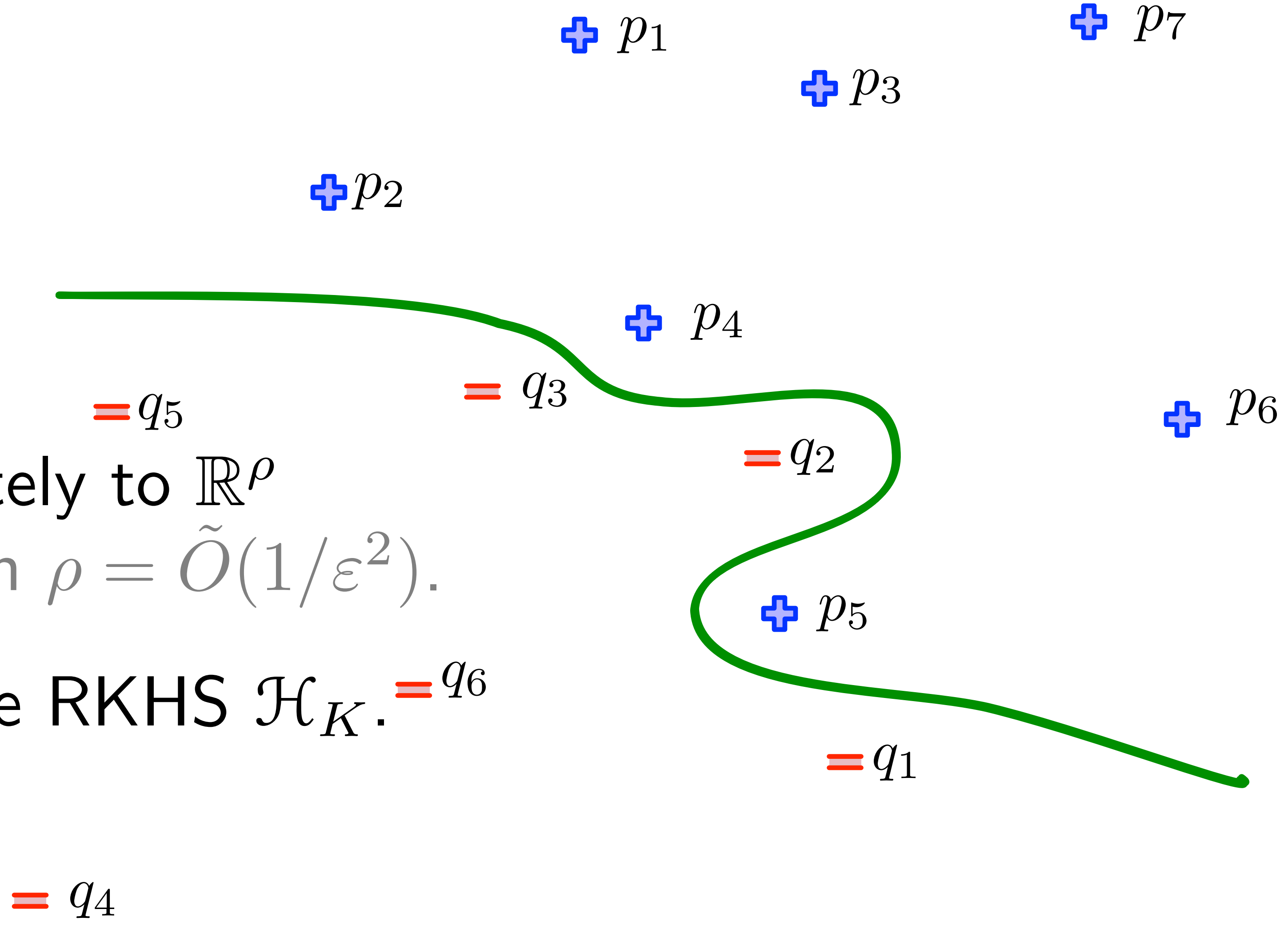
Sketched Kernel SVM Algorithm

```
Algorithm:
Define mapping $\phi : X \to \mathbb{R}^\rho$
    choose $w = \sigma(x)\phi(x)$
    for $i = 1$ to $1/\varepsilon\gamma^2$ steps
        $x' = \arg\min \sigma(x)\langle \phi(x), w\rangle$
        $w = w + \sigma(x')\phi(x')$
```

$K(x,p) = (1 + \langle x,p\rangle)^r \in \mathbb{R}^{O(rd)}$
$K(x,p) = \exp(-\|x-p\|^2)$ maps approximately to $\mathbb{R}^\rho$
e.g., Rahimi+Recht [NeurIPS07] additive ε in $\rho = \tilde{O}(1/\varepsilon^2)$.

$\phi(x) = K(x,\cdot)$ is a element of function space RKHS $\mathcal{H}_K$.
kernel density estimate:

$$\mathrm{KDE}_W = \frac{1}{|W|}\sum_{w\in W}\phi(w).$$

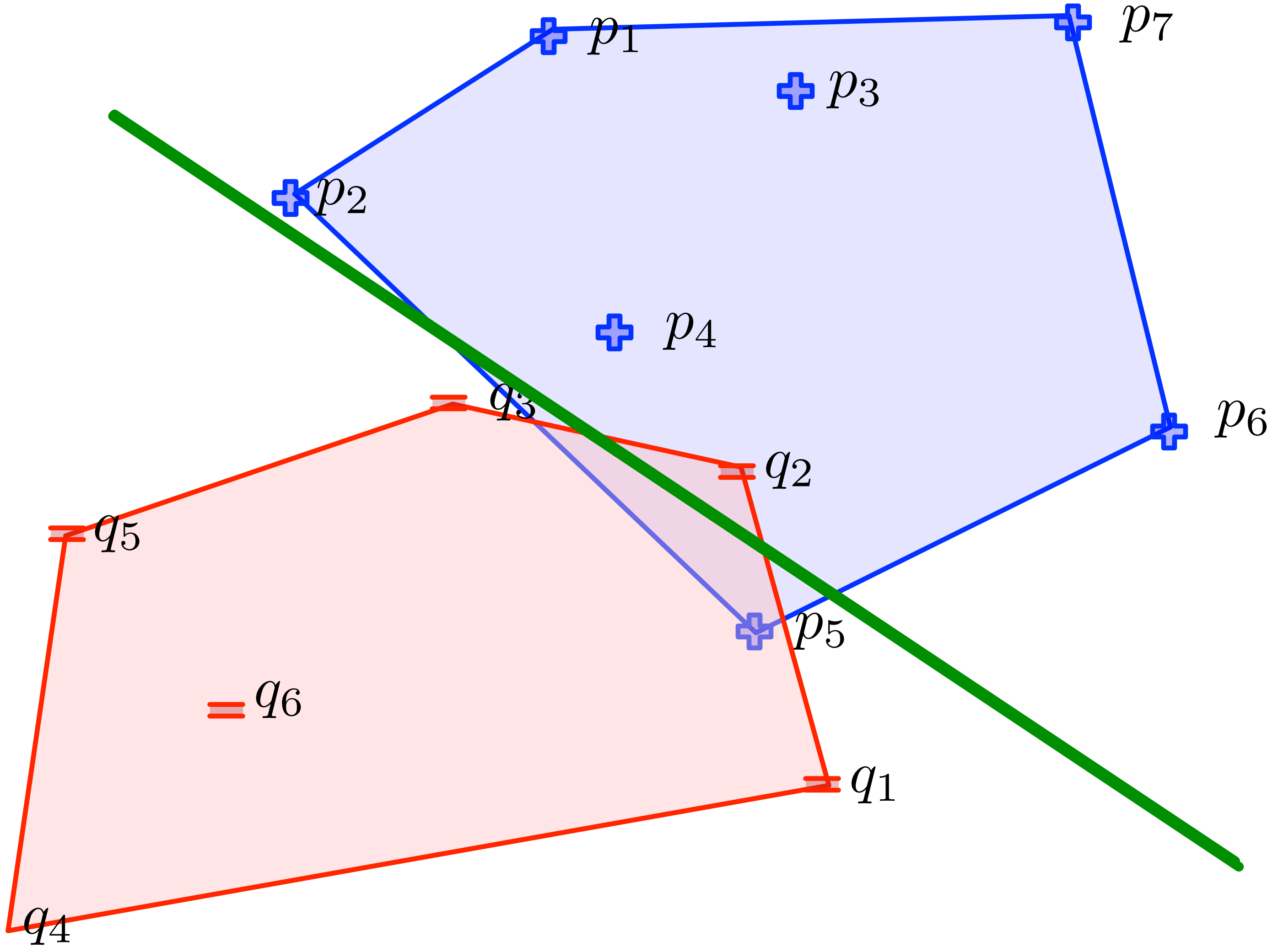


Loss Functions

Set up penalty for misclassified points

$$f_X(w) = \sum_{x \in X} \ell(w, x) + \mathsf{prior}(w)$$

loss $\ell_i = \ell(w, x_i) = \ell(z_i)$
with $z_i = \sigma(x_i)\langle w, x_i \rangle$

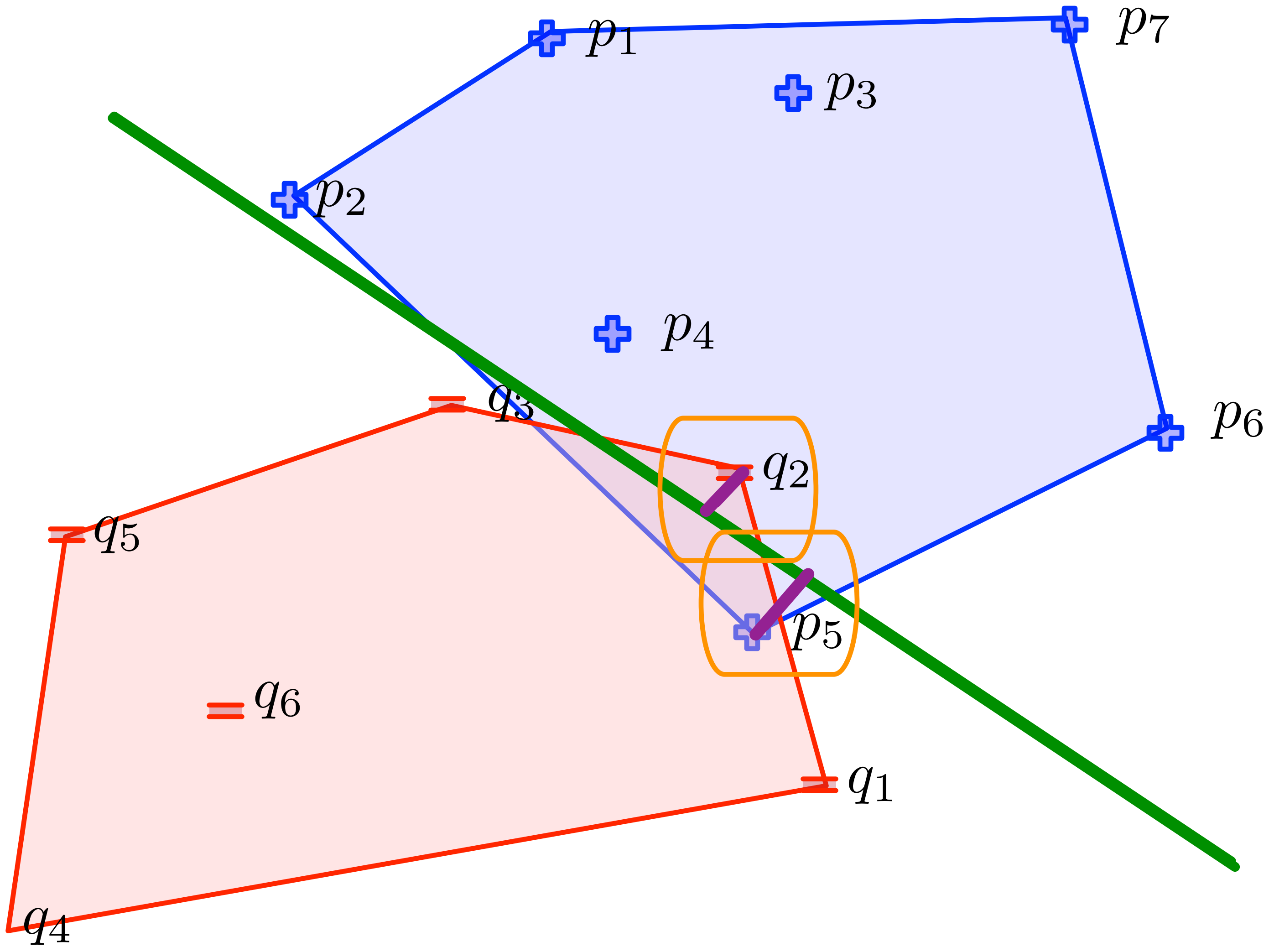


Loss Functions

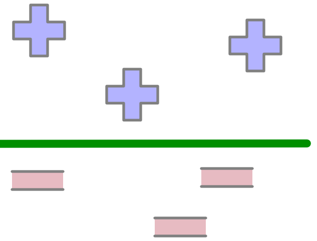
Set up penalty for misclassified points

$$f_X(w) = \sum_{x \in X} \ell(w, x) + \text{prior}(w)$$

loss $\ell_i = \ell(w, x_i) = \ell(z_i)$
with $z_i = \sigma(x_i)\langle w, x_i \rangle$



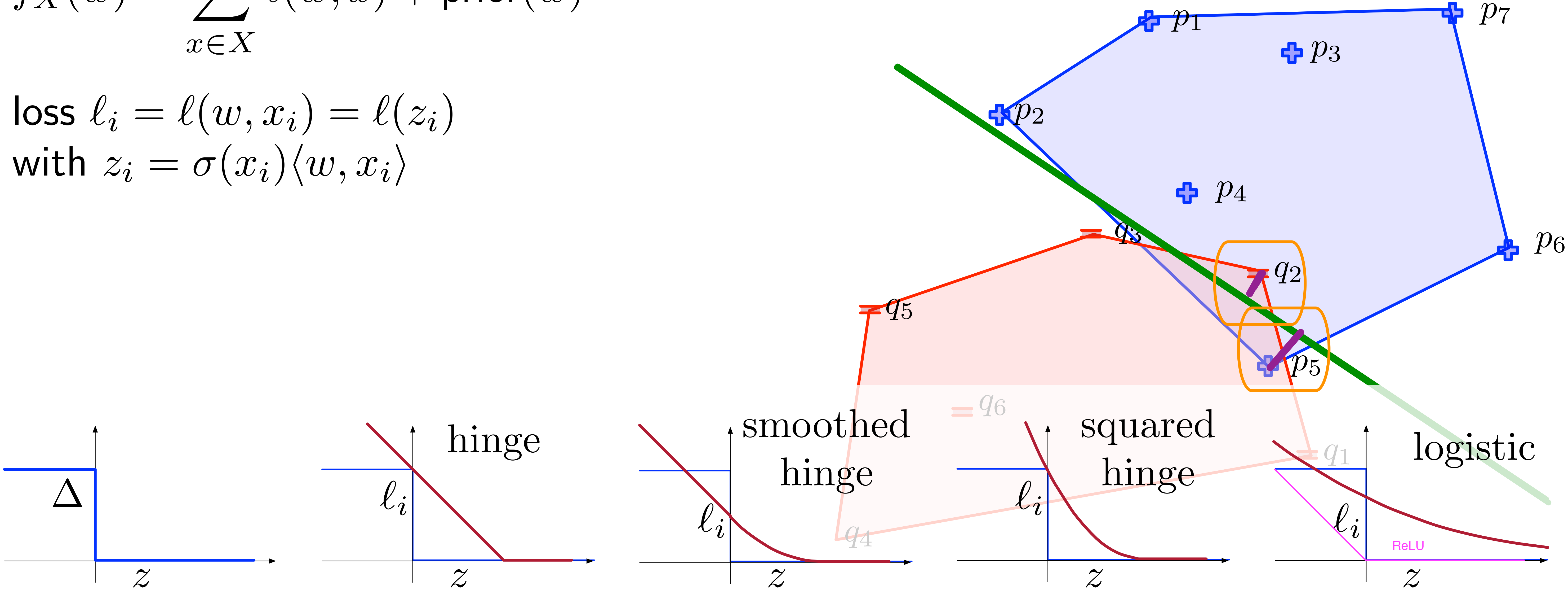
Loss Functions



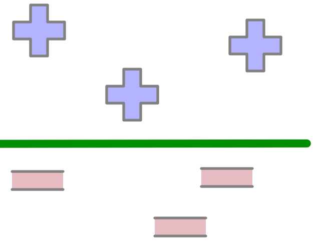
Set up penalty for misclassified points

$$f_X(w) = \sum_{x \in X} \ell(w, x) + \text{prior}(w)$$

loss $\ell_i = \ell(w, x_i) = \ell(z_i)$
with $z_i = \sigma(x_i)\langle w, x_i \rangle$



Loss Functions

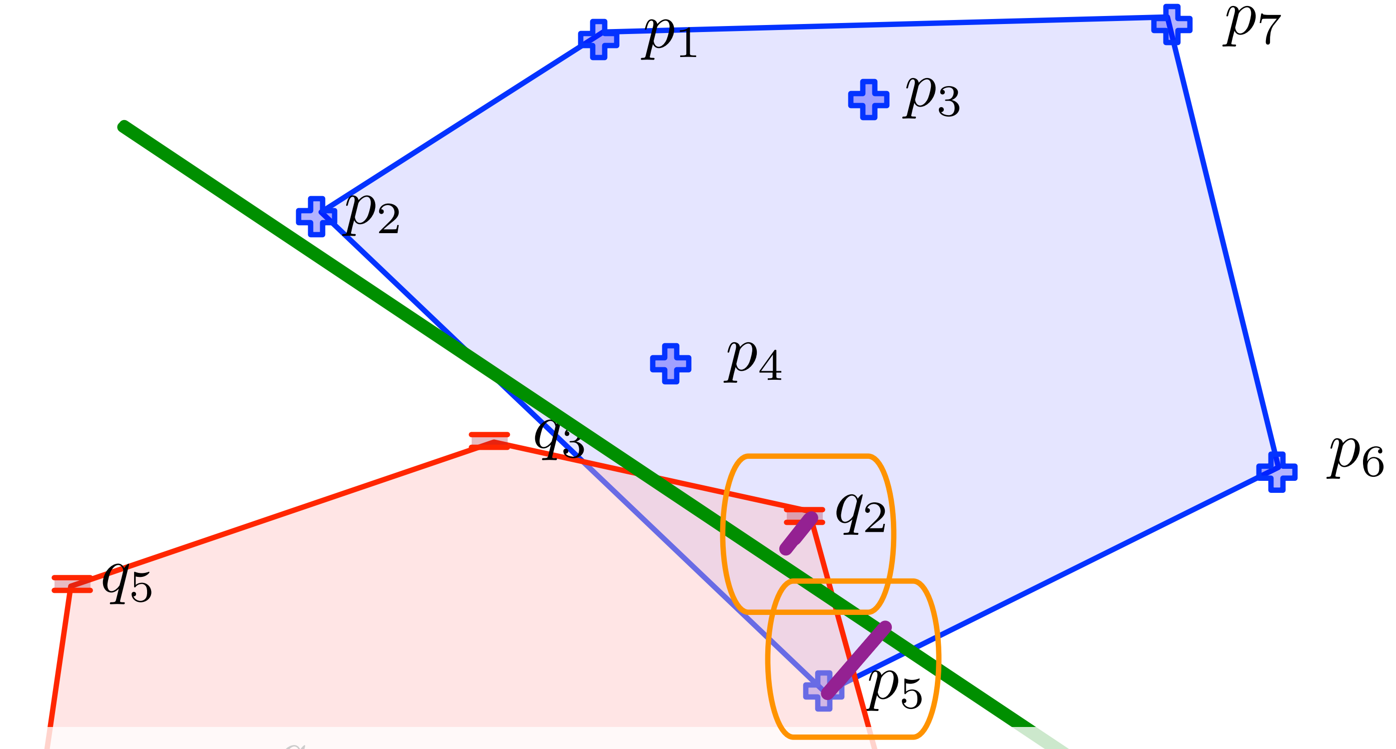


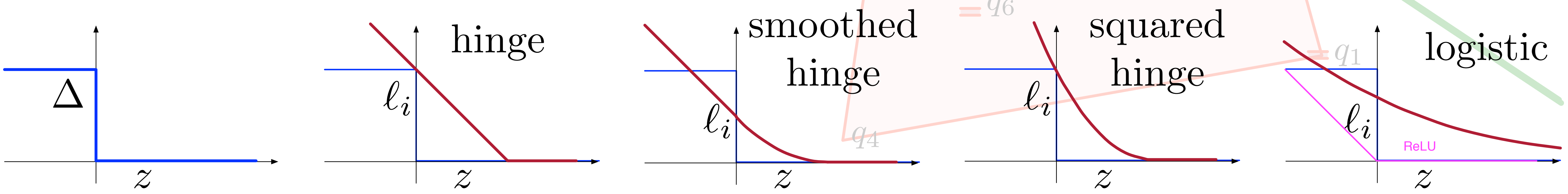
Set up penalty for misclassified points

$$f_X(w) = \sum_{x \in X} \ell(w, x) + \mathrm{prior}(w)$$

loss $\ell_i = \ell(w, x_i) = \ell(z_i)$
with $z_i = \sigma(x_i)\langle w, x_i \rangle$

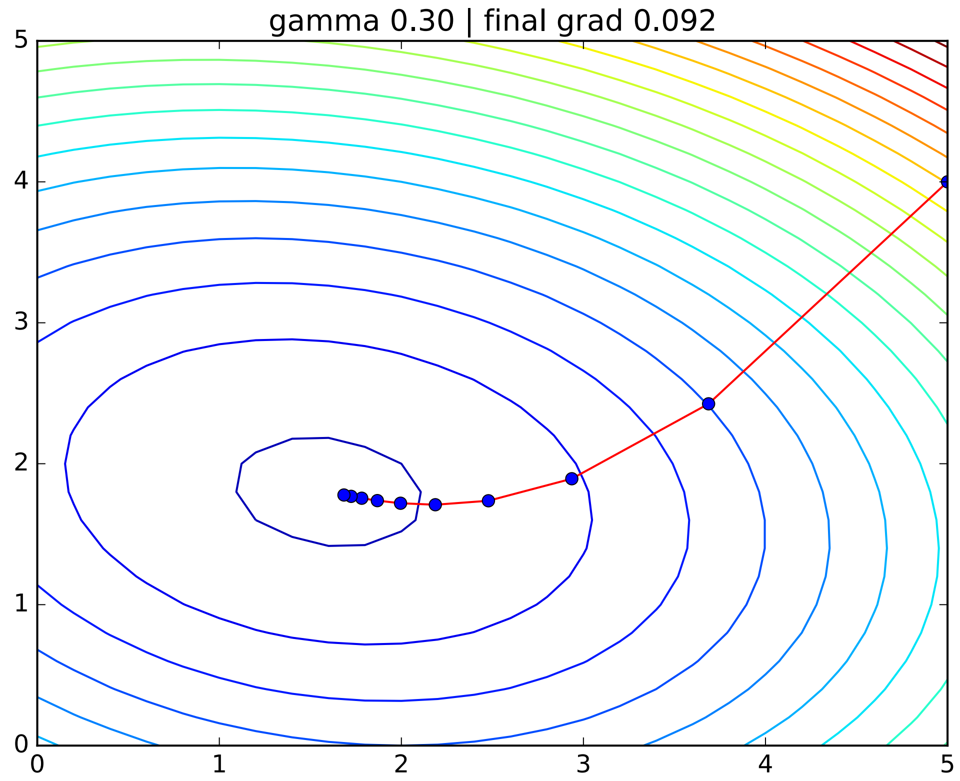
If linear or K is pd
then f_X is **convex**

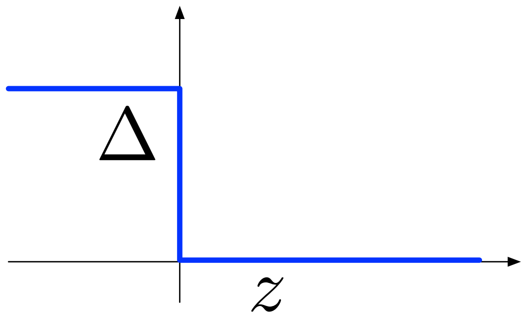


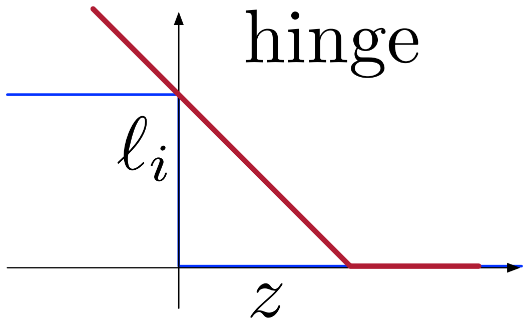


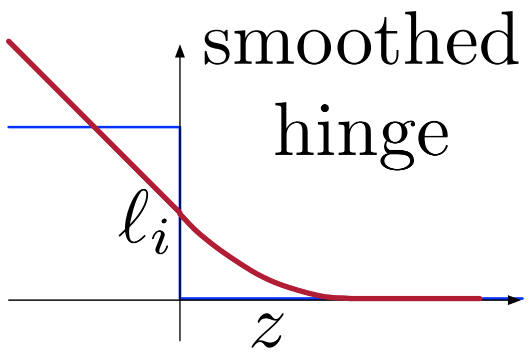
Coresets for Optimization

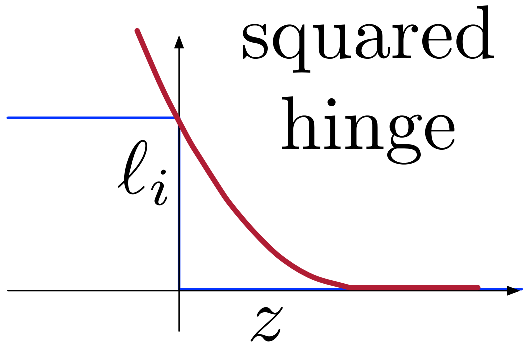
Solve for $w^* = \arg\min_w f_X(w)$ ... gradient descent

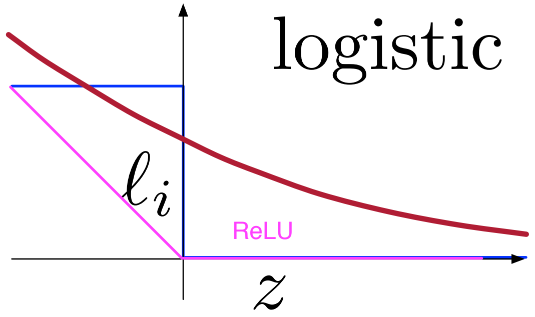










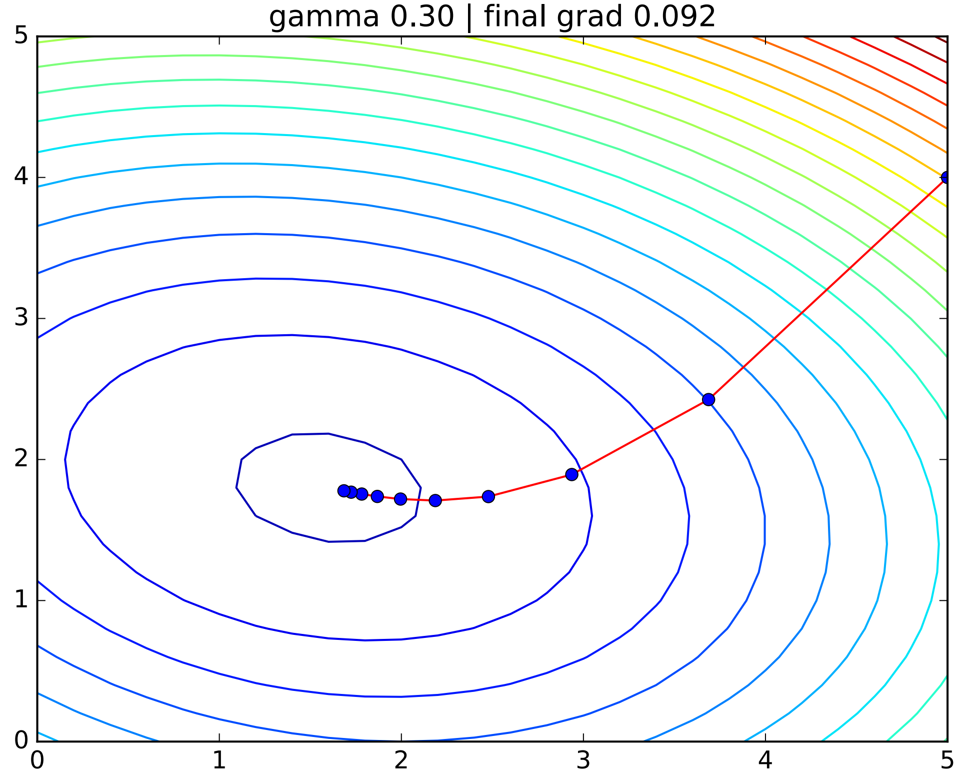


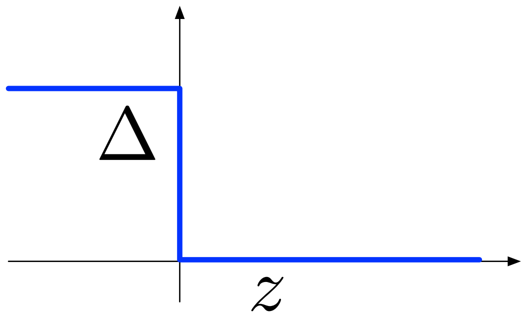
Coresets for Optimization

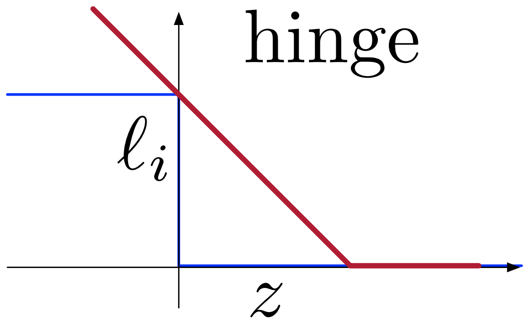
Solve for $w^* = \arg\min_w f_X(w)$ ... gradient descent

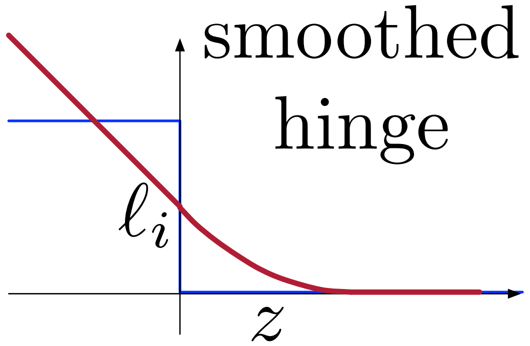
Subgradient Descent $\approx$ Frank-Wolfe
Move towards most helpful point
$O(C/\varepsilon^2)$ steps to $\pm\varepsilon$ mean for $\|x\| \leq 1$.

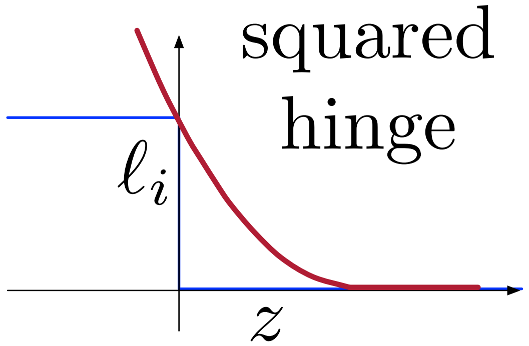
Stochastic gradient descent (SGD)
For large X, most common
Randomly chooses $x \in X$, and step towards $-\nabla f_x(w)$.

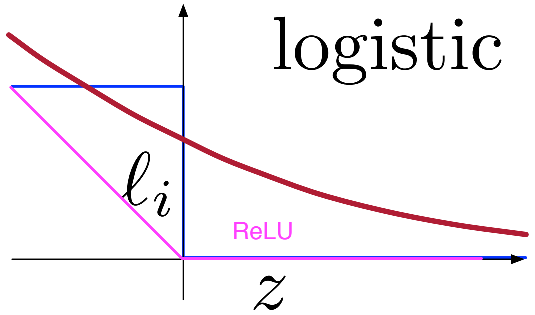










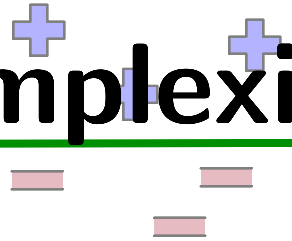


VC-dimension and Sample Complexity

Assume: *data* X *is drawn iid from* μ.

Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x\sim\mu}[\mathbf{1}(g(x) = \sigma(x))]$.

VC-dimension and Sample Complexity

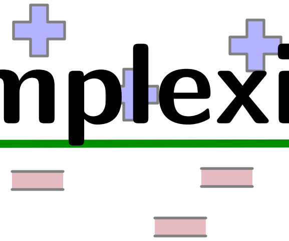


Assume: *data* X *is drawn iid from* μ.

Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x\sim\mu}[\mathbf{1}(g(x) = \sigma(x))]$.

how much accuracy is preserved under random sampling?

VC-dimension and Sample Complexity

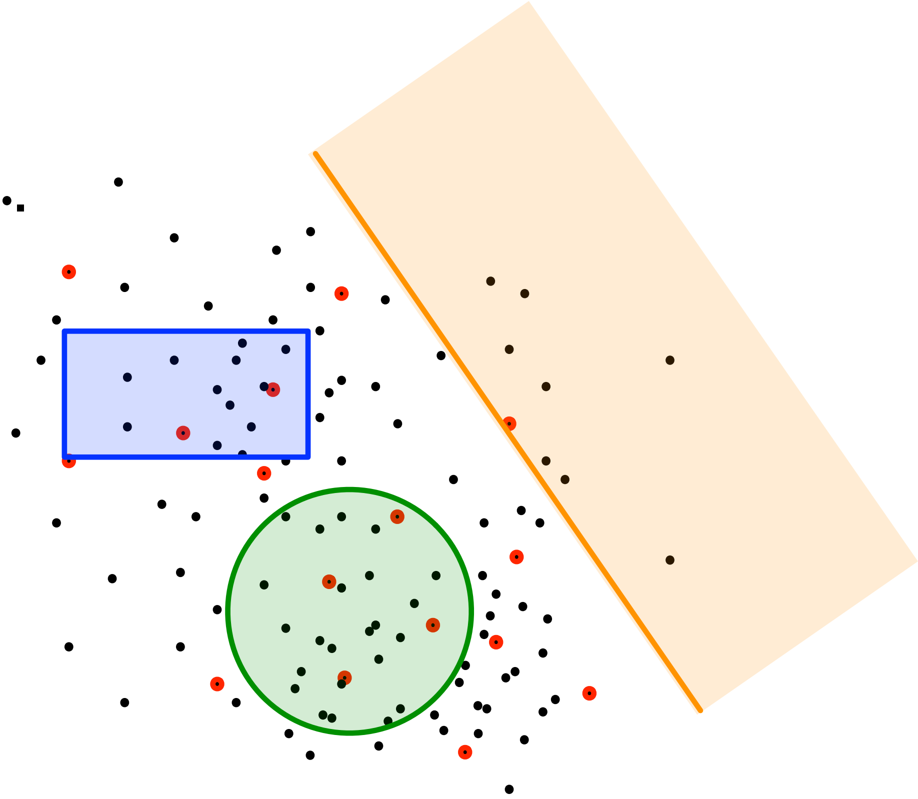


Assume: *data X is drawn iid from μ.*

Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x\sim\mu}[\mathbf{1}(g(x) = \sigma(x))]$.

how much accuracy is preserved under random sampling?

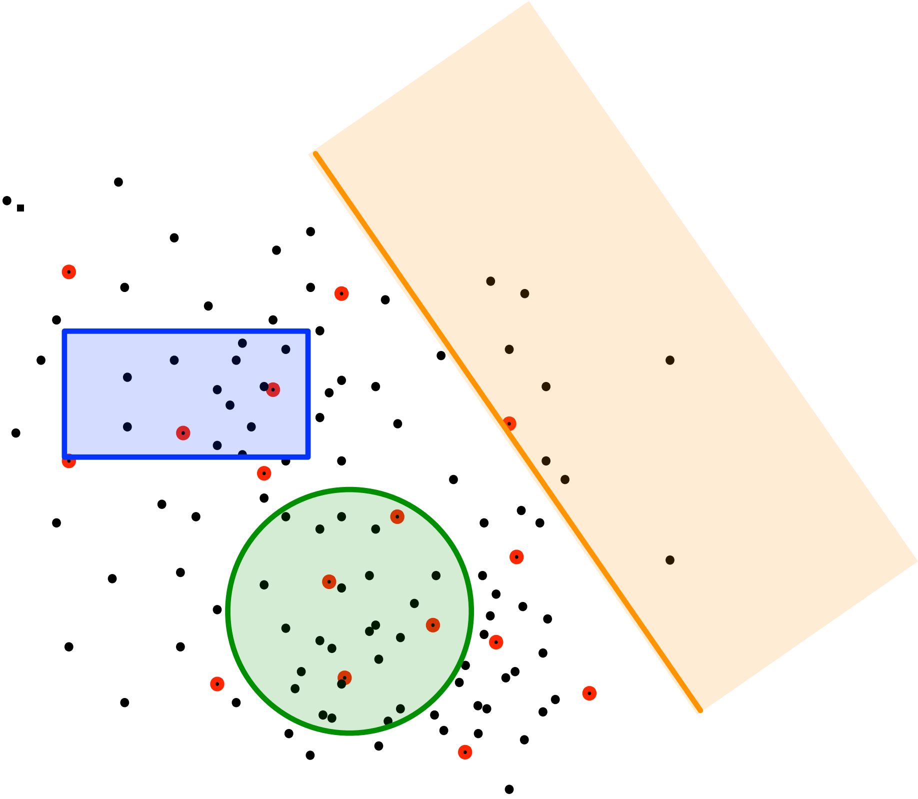
Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.



VC-dimension and Sample Complexity

Assume: *data X is drawn iid from μ.*
Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x \sim \mu}[\mathbf{1}(g(x) = \sigma(x))]$.

how much accuracy is preserved under random sampling?



Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.
VC-dimension ν: largest $Y \subset X$ so all $Z \subset Y$ can be defined so $Z = Y \cap R$ for some $R \in \mathcal{R}$.

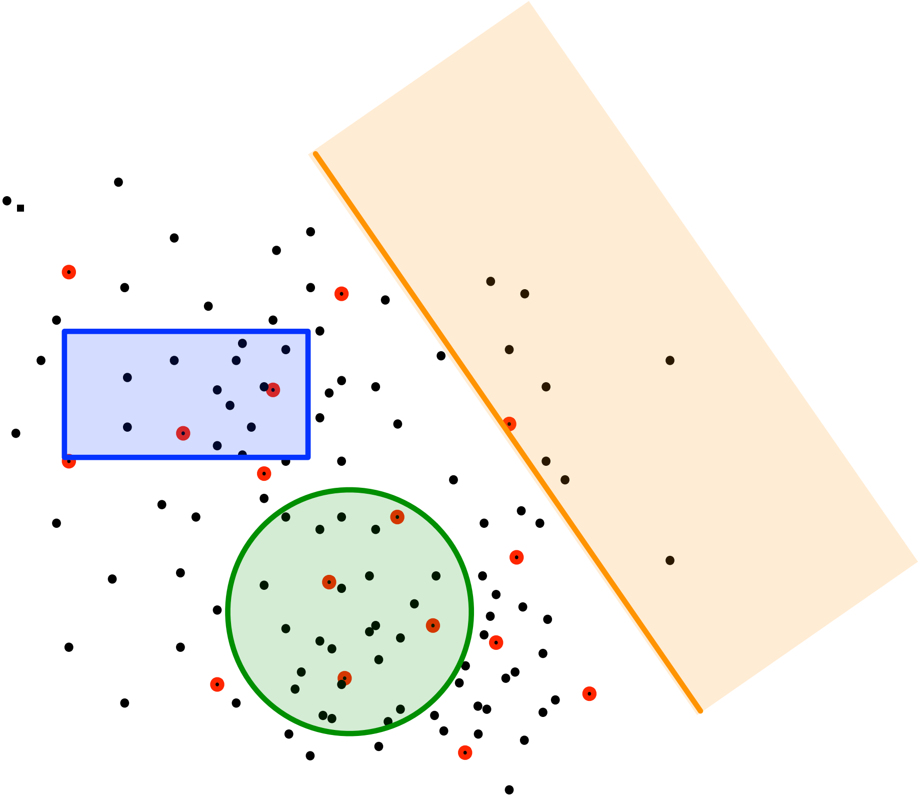
$\leftrightarrow$ sublinear range searching, approximate hitting sets

VC-dimension and Sample Complexity

Assume: *data X is drawn iid from μ.*

Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x \sim \mu}[\mathbf{1}(g(x) = \sigma(x))]$.

how much accuracy is preserved under random sampling?



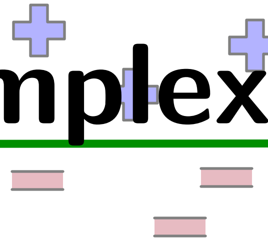
Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.
VC-dimension ν: largest $Y \subset X$ so all $Z \subset Y$ can be defined so $Z = Y \cap R$ for some $R \in \mathcal{R}$.

$\leftrightarrow$ sublinear range searching, approximate hitting sets

An ε-**sample** is a subset $S \subset X$ so $|S| = O(1/\varepsilon^{2-2/(\nu+1)})$

$$\max_{R \in \mathcal{R}} \left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon.$$

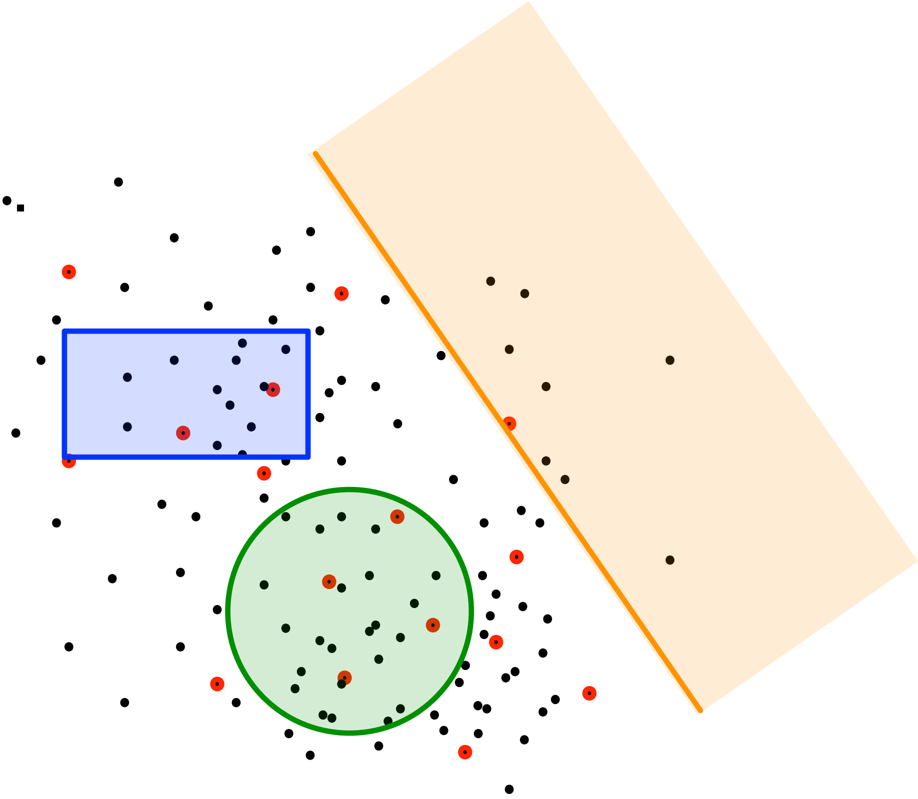
VC-dimension and Sample Complexity



Assume: *data X is drawn iid from μ.*

Build: classifier $g : \mathbb{R}^d \to \{-1, +1\}$ so $\mathbf{E}_{x \sim \mu}[\mathbf{1}(g(x) = \sigma(x))]$.

how much accuracy is preserved under random sampling?



Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.
VC-dimension ν: largest $Y \subset X$ so all $Z \subset Y$ can be defined so $Z = Y \cap R$ for some $R \in \mathcal{R}$.

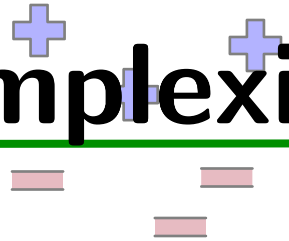
↔ sublinear range searching, approximate hitting sets

An ε-**sample** is a subset $S \subset X$ so

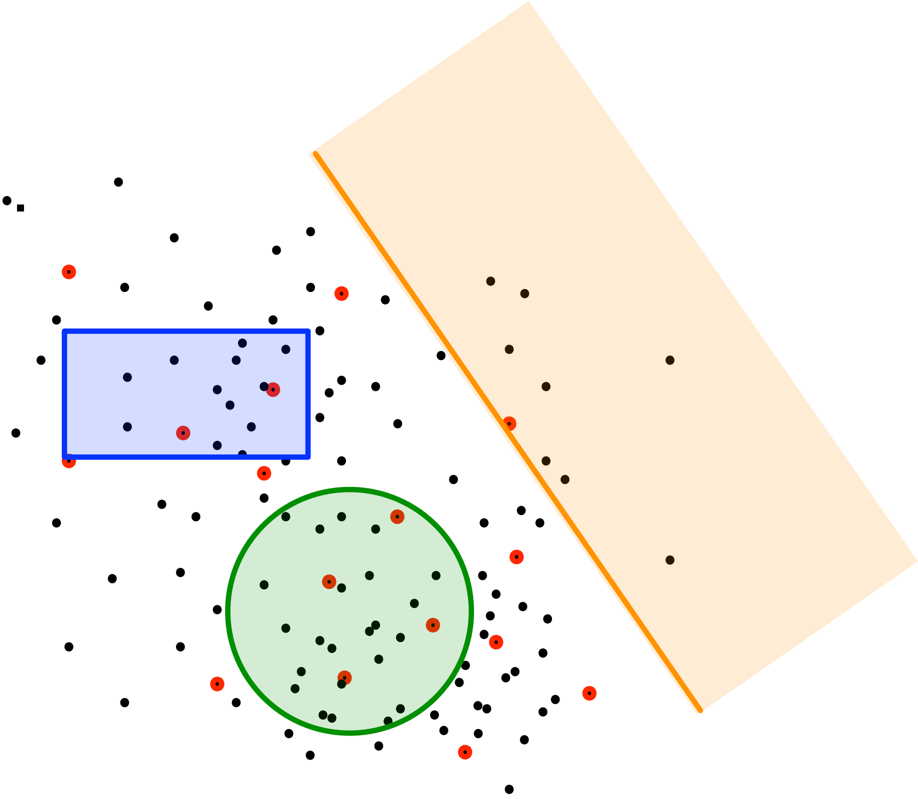
$$\max_{R \in \mathcal{R}} \left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon.$$

	Geom	**ML**
$\mathcal{R}$	ranges	classifiers
X	ground set	samples

VC-dimension and Sample Complexity



Assume: *data X is drawn iid from μ.*
Build: classifier $g : \mathbb{R}^d \to \{-1,+1\}$ so $\mathbf{E}_{x\sim\mu}[\mathbf{1}(g(x) = \sigma(x))]$.



how much accuracy is preserved under random sampling?

Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.
VC-dimension ν: largest $Y \subset X$ so all $Z \subset Y$ can be defined so $Z = Y \cap R$ for some $R \in \mathcal{R}$.

$\leftrightarrow$ sublinear range searching, approximate hitting sets

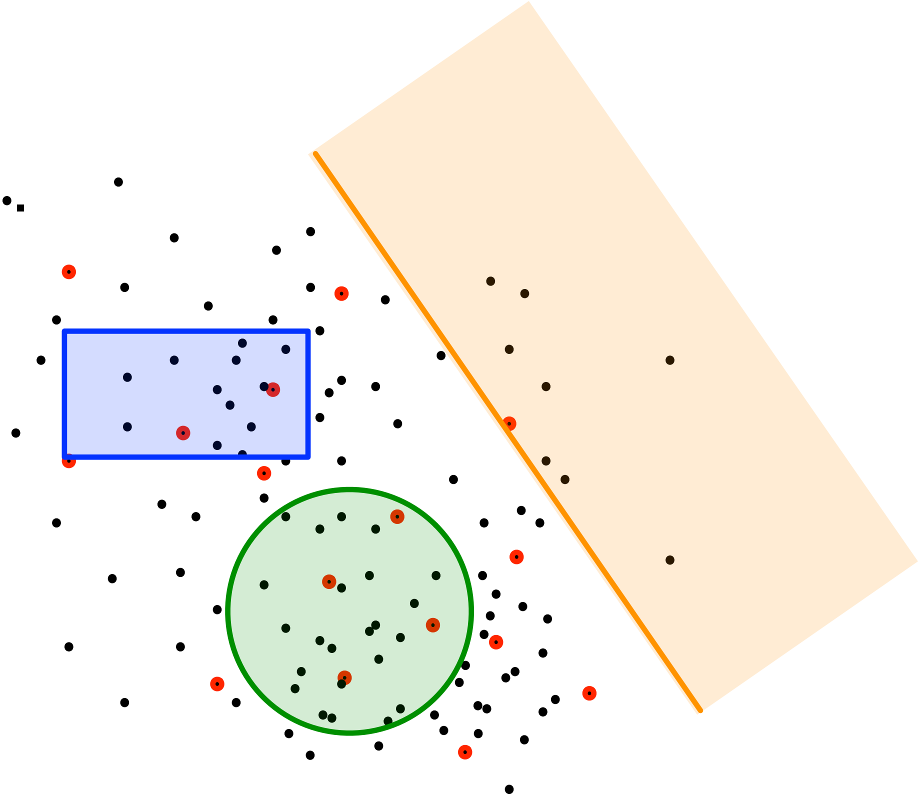
An ε-**sample** is a subset $S \subset X$ so

$$\max_{R\in\mathcal{R}} \left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon.$$

	Geom	**ML**
$\mathcal{R}$	ranges	classifiers
X	ground set	samples
$O(\frac{\nu}{\varepsilon^2})$	ε-sample [VC71]	agnostic learning

VC-dimension and Sample Complexity

Assume: *data X is drawn iid from μ.*
Build: classifier $g : \mathbb{R}^d \to \{-1,+1\}$ so $\mathbf{E}_{x\sim\mu}[\mathbf{1}(g(x) = \sigma(x))]$.



how much accuracy is preserved under random sampling?

Range space $(X, \mathcal{R})$, where $\mathcal{R}$ is family of subsets.
VC-dimension ν: largest $Y \subset X$ so all $Z \subset Y$ can be defined so $Z = Y \cap R$ for some $R \in \mathcal{R}$.

$\leftrightarrow$ sublinear range searching, approximate hitting sets

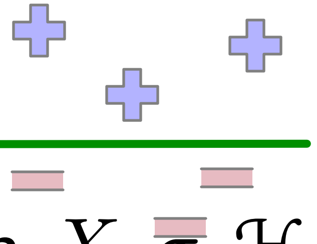
An ε-**sample** is a subset $S \subset X$ so

$$\max_{R\in\mathcal{R}} \left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon.$$

An ε-**net** is a subset $S \subset X$ such that each $R \in \mathcal{R}$ so $\frac{|R\cap X|}{|X|} \geq \varepsilon$ then $S \cap R \neq \emptyset$.

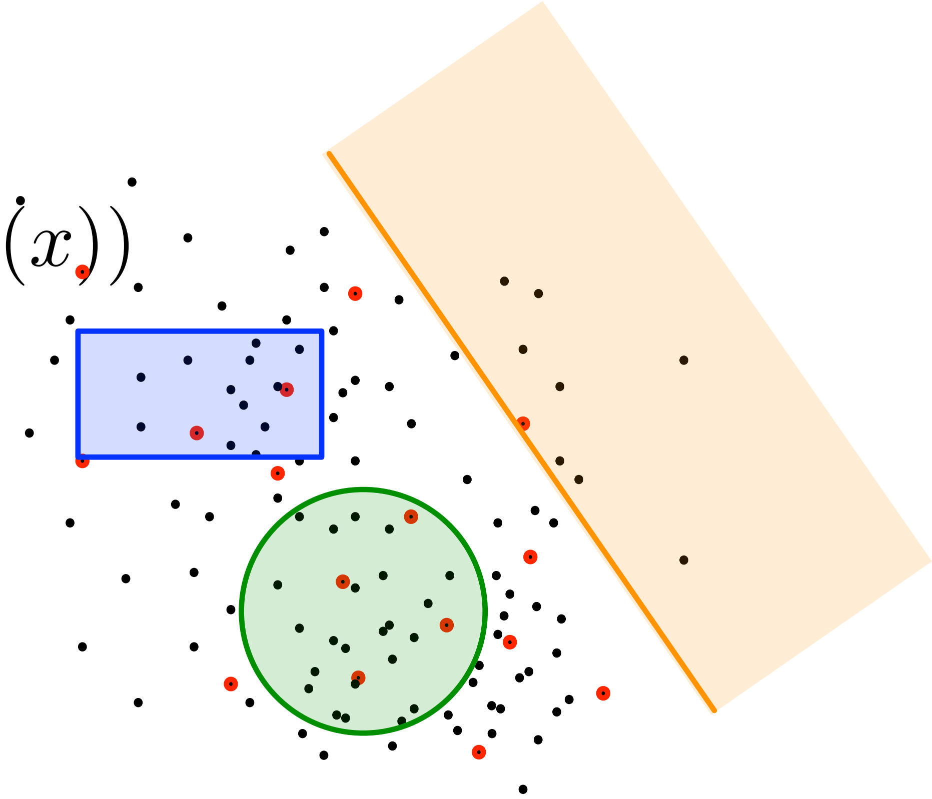
	Geom	**ML**
$\mathcal{R}$	ranges	classifiers
X	ground set	samples
$O(\frac{\nu}{\varepsilon^2})$	ε-sample [VC71]	agnostic learning
$O(\frac{\nu}{\varepsilon}\log\frac{\nu}{\varepsilon})$	ε-net [HW85]	perfect classifiers

Recent Results (... of mine)



- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

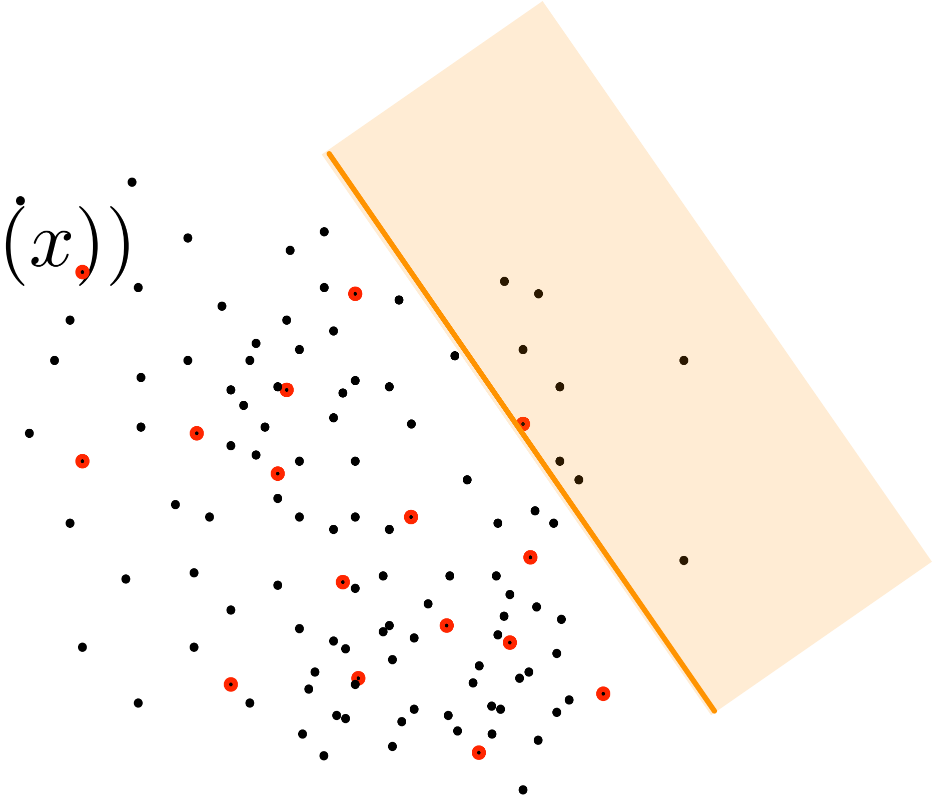


Recent Results (... of mine)

- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|}\sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$



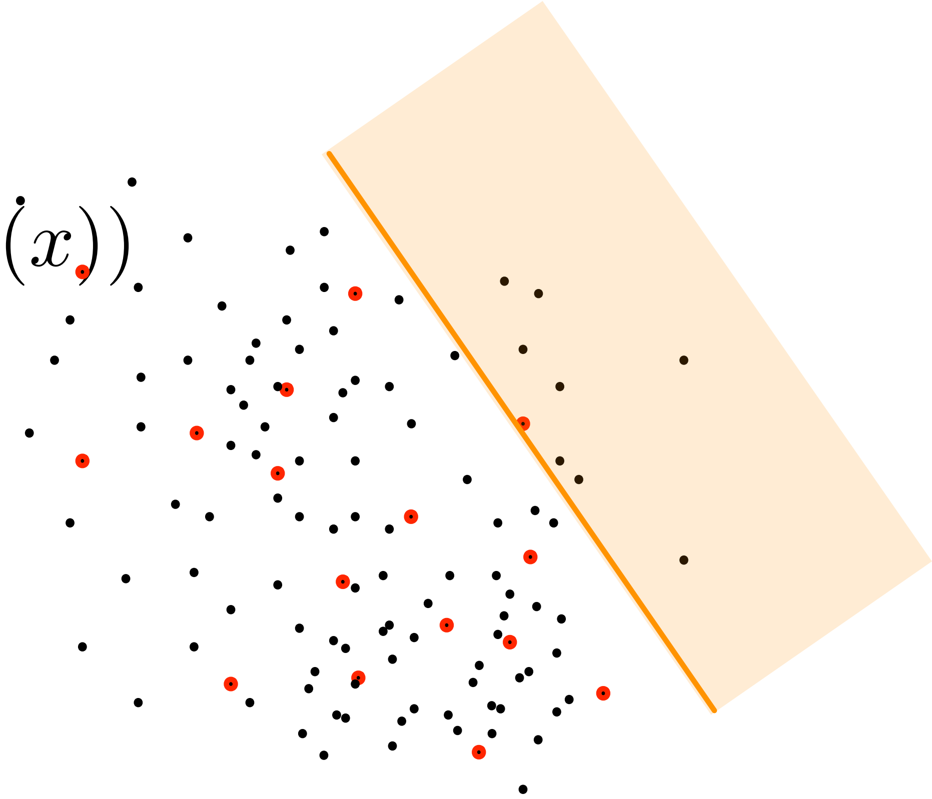
Recent Results (... of mine)

- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$

... in $O((1/\varepsilon^{d+1/3})\text{polylog}(1/\varepsilon))$ time. [Matheny+P '18]



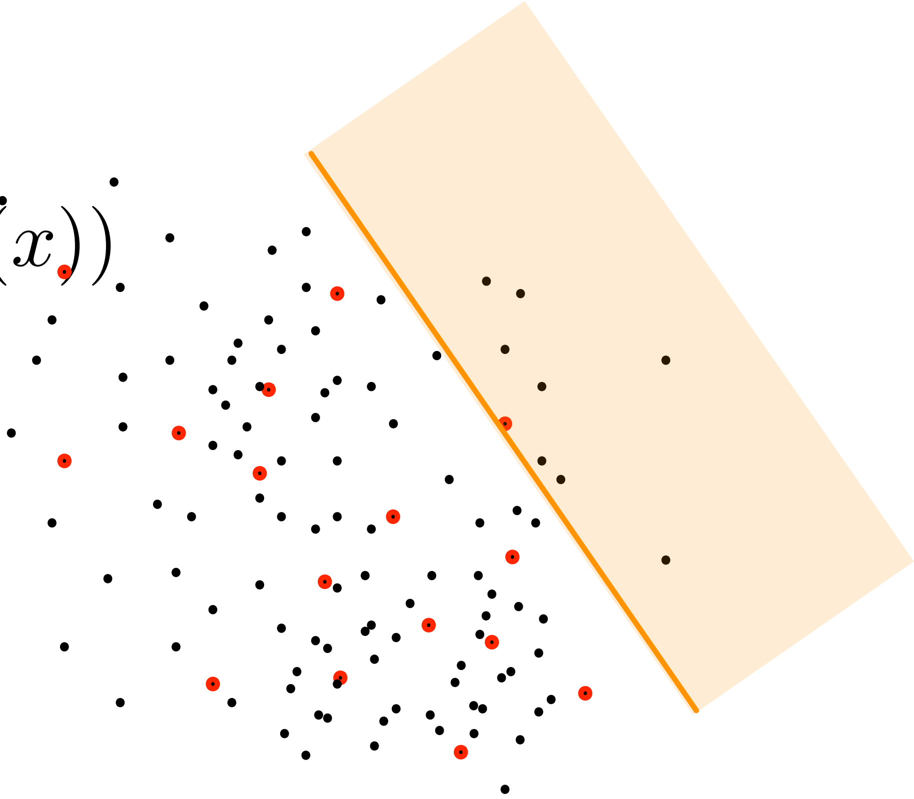
Recent Results (... of mine)

- Approximate Maximum Disagreement Problem $X, \overline{\sigma}, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$

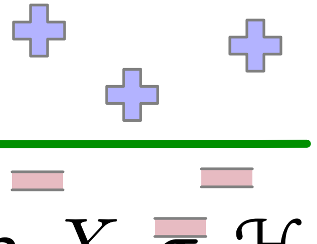
... in $O((1/\varepsilon^{d+1/3})$polylog$(1/\varepsilon))$ time. [Matheny+P '18]



- Sublevel-Set Range Space $(X, \mathcal{R}_{\mathtt{d}})$ defined for $R_{p,\tau} \in \mathcal{R}_{\mathtt{d}}$ as

$R_{p,\tau} = \{x \in X \mid \mathtt{d}(p,x) \leq \tau\}$.

Recent Results (… of mine)

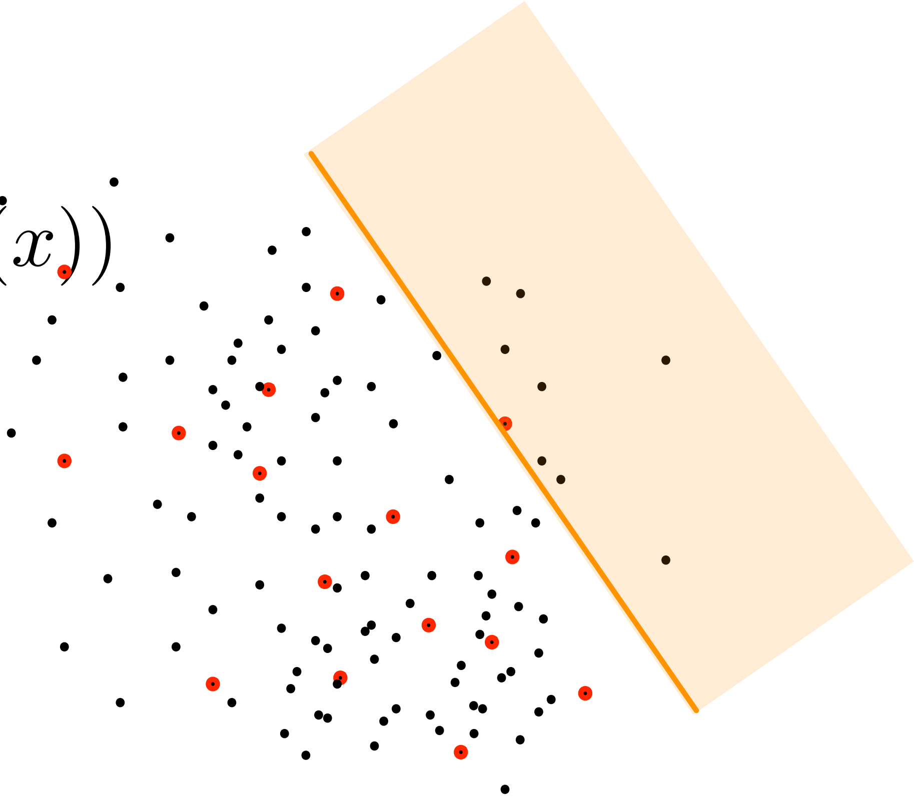


- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|}\sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$

… in $O((1/\varepsilon^{d+1/3})\text{polylog}(1/\varepsilon))$ time. [Matheny+P '18]

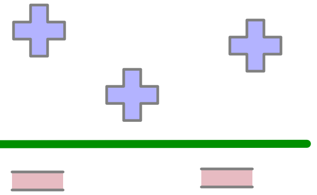


- Sublevel-Set Range Space $(X, \mathcal{R}_{\mathtt{d}})$ defined for $R_{p,\tau} \in \mathcal{R}_{\mathtt{d}}$ as

$R_{p,\tau} = \{x \in X \mid \mathtt{d}(p,x) \leq \tau\}$.

General KDE: $K_{\mathtt{d}}(x,p) = \exp(-\mathtt{d}(x,p)^2)$ and $\text{KDE}_X(p) = \frac{1}{|X|}\sum_{x \in X} K(x,p)$

Recent Results (... of mine)

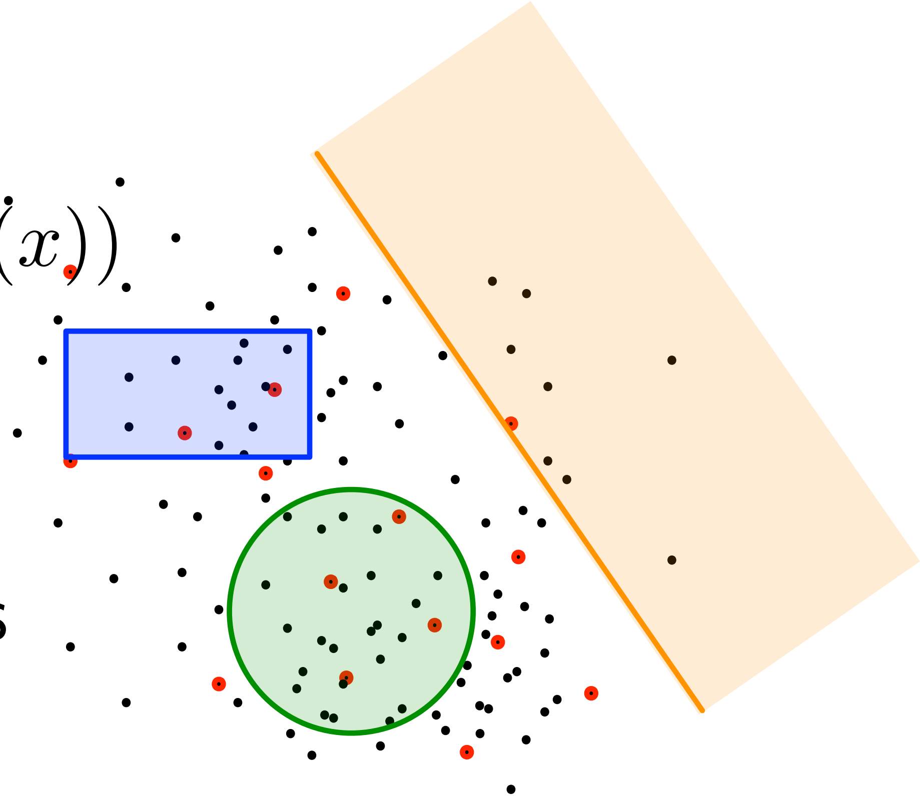


- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$

... in $O((1/\varepsilon^{d+1/3})\text{polylog}(1/\varepsilon))$ time. [Matheny+P '18]



- Sublevel-Set Range Space $(X, \mathcal{R}_{\mathtt{d}})$ defined for $R_{p,\tau} \in \mathcal{R}_{\mathtt{d}}$ as

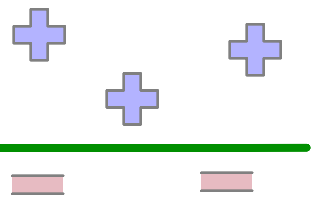
$R_{p,\tau} = \{x \in X \mid \mathtt{d}(p, x) \leq \tau\}$.

General KDE: $K_{\mathtt{d}}(x, p) = \exp(-\mathtt{d}(x, p)^2)$ and $\mathrm{KDE}_X(p) = \frac{1}{|X|} \sum_{x \in X} K(x, p)$

If VC-dimension of $(X, \mathcal{R}_{\mathtt{d}})$ is $\nu_{\mathtt{d}}$ then $S \sim X$ of size $O(\nu_{\mathtt{d}}/\varepsilon^2)$

has $\|\mathrm{KDE}_X - \mathrm{KDE}_S\|_\infty \leq \varepsilon$ [Komaraju etal '11]

Recent Results (... of mine)

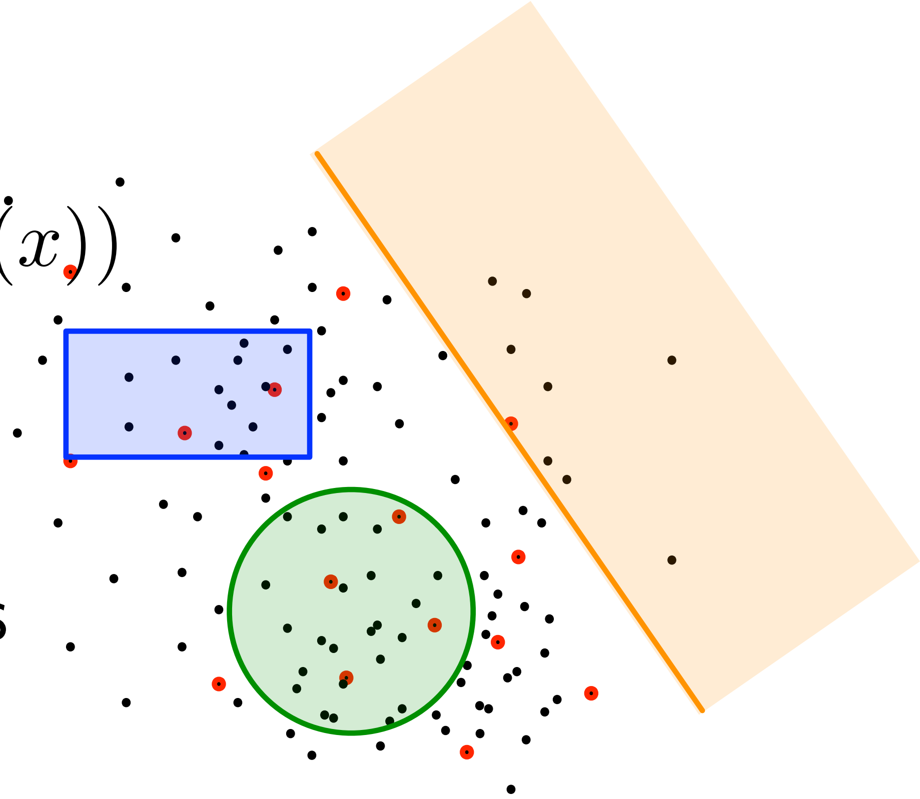


- Approximate Maximum Disagreement Problem $X, \sigma, \mathcal{H}$

Find halfspace h maximizes $\Delta_X(h) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}(h(x) = \sigma(x))$

Find $\hat{h} \in \mathcal{H}$ so $\Delta_X(h^*) - \Delta_X(\hat{h}) \leq \varepsilon$

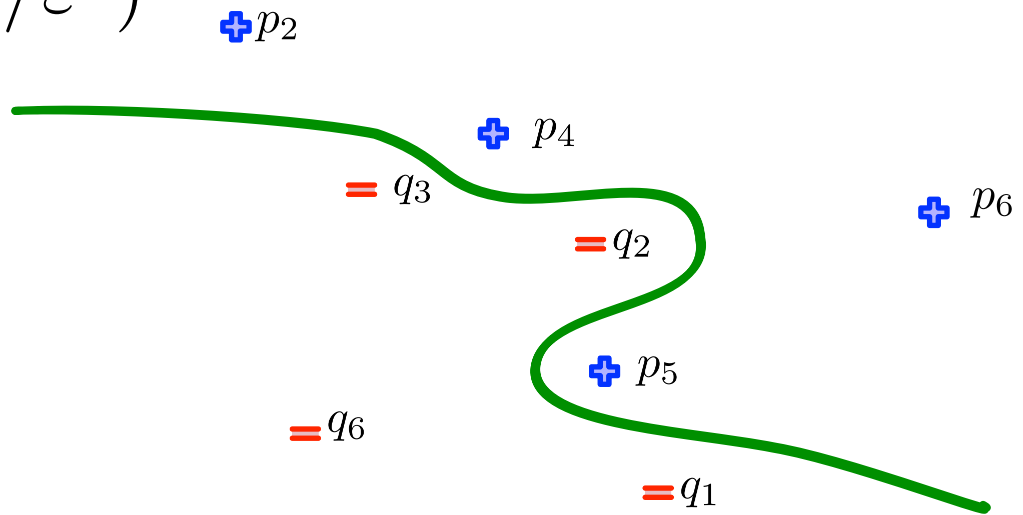
... in $O((1/\varepsilon^{d+1/3})\text{polylog}(1/\varepsilon))$ time. [Matheny+P '18]



- Sublevel-Set Range Space $(X, \mathcal{R}_\mathtt{d})$ defined for $R_{p,\tau} \in \mathcal{R}_\mathtt{d}$ as $R_{p,\tau} = \{x \in X \mid \mathtt{d}(p,x) \leq \tau\}$.

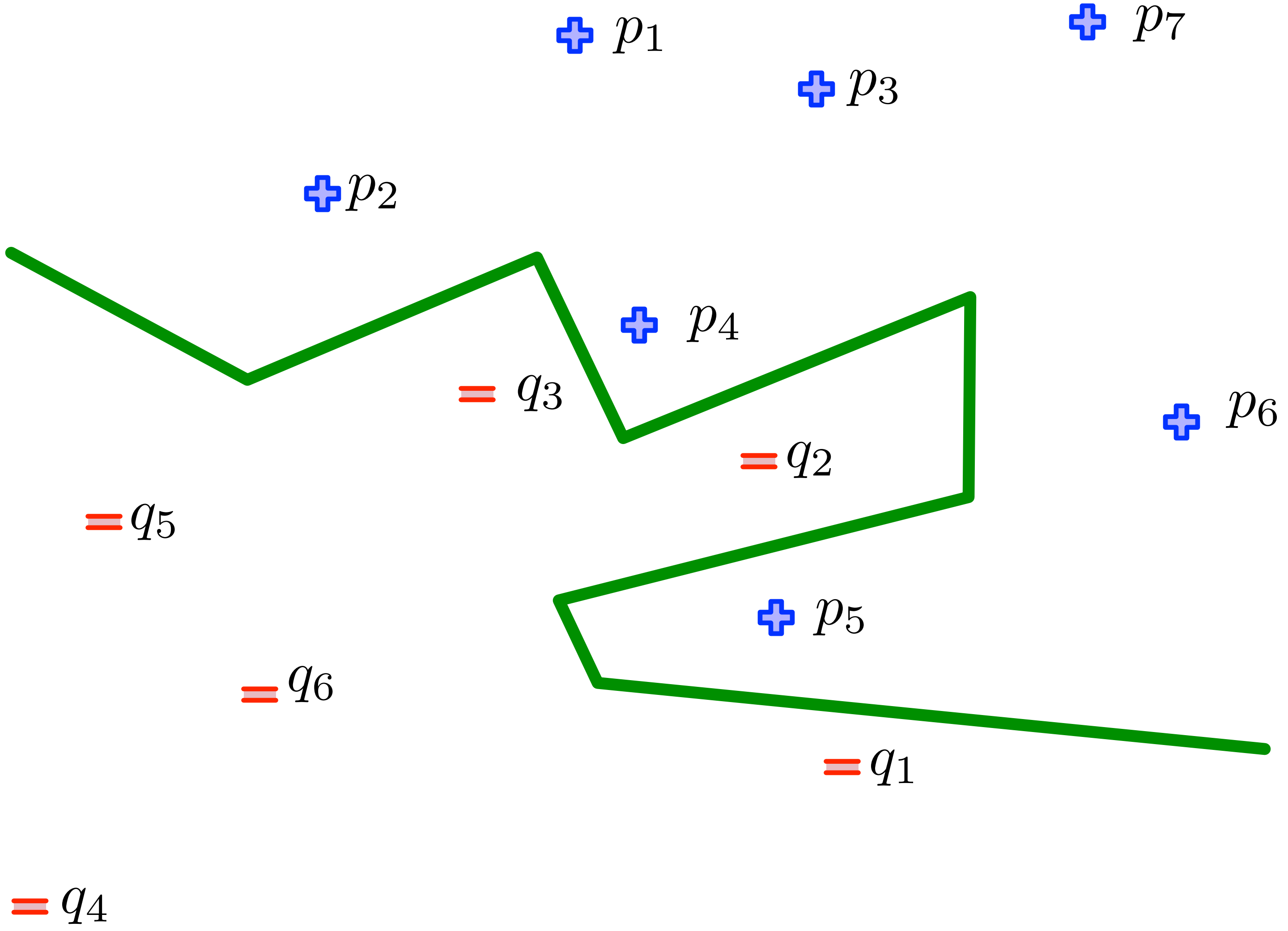
General KDE: $K_\mathtt{d}(x,p) = \exp(-\mathtt{d}(x,p)^2)$ and $\text{KDE}_X(p) = \frac{1}{|X|} \sum_{x \in X} K(x,p)$

If VC-dimension of $(X, \mathcal{R}_\mathtt{d})$ is $\nu_\mathtt{d}$ then $S \sim X$ of size $O(\nu_\mathtt{d}/\varepsilon^2)$ has $\|\text{KDE}_X - \text{KDE}_S\|_\infty \leq \varepsilon$ [Komaraju etal '11] approximates margin $\pm\varepsilon$.



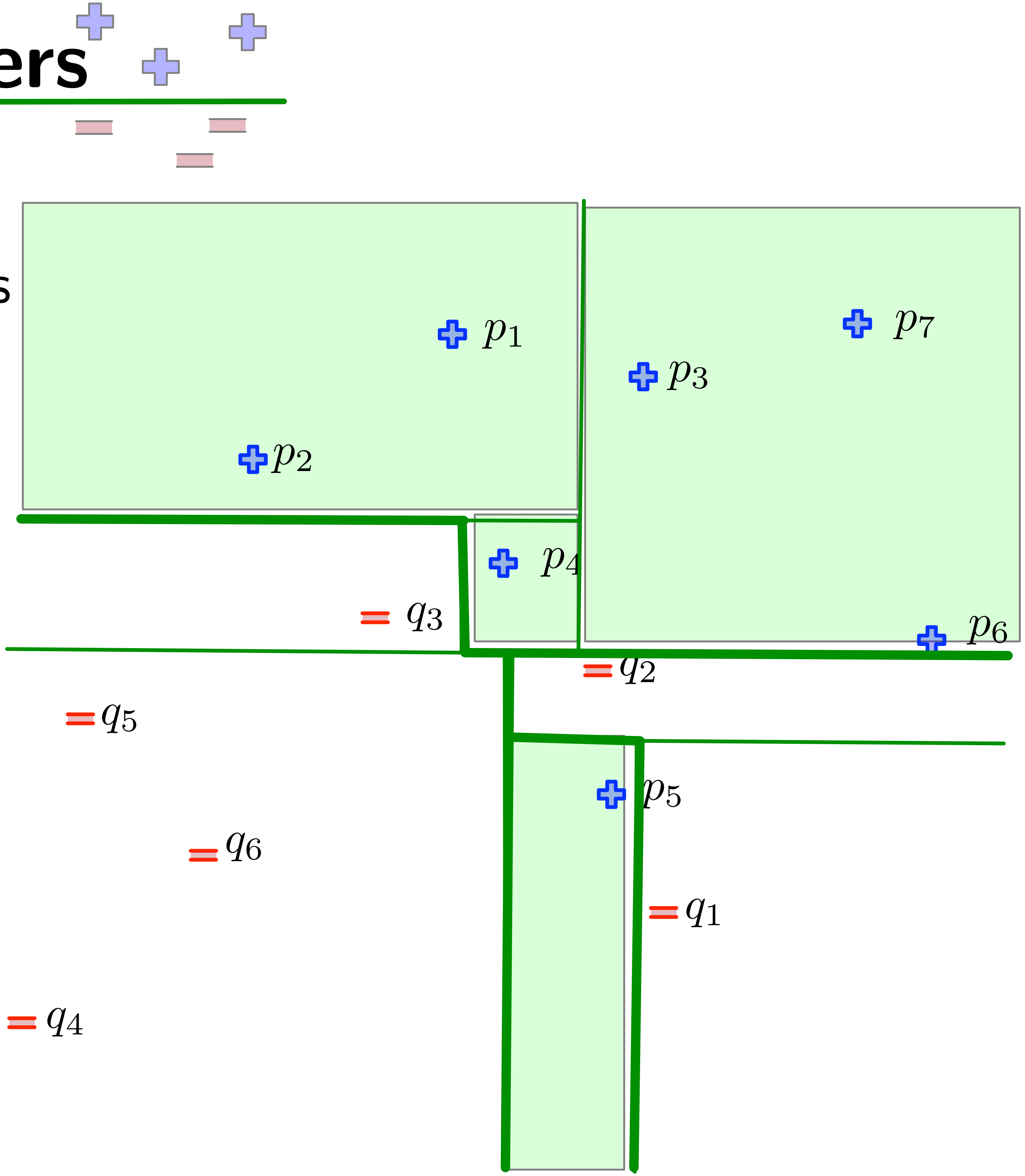
Many, many types of classifiers

- **KNN**: voting based on k nearest neighbors



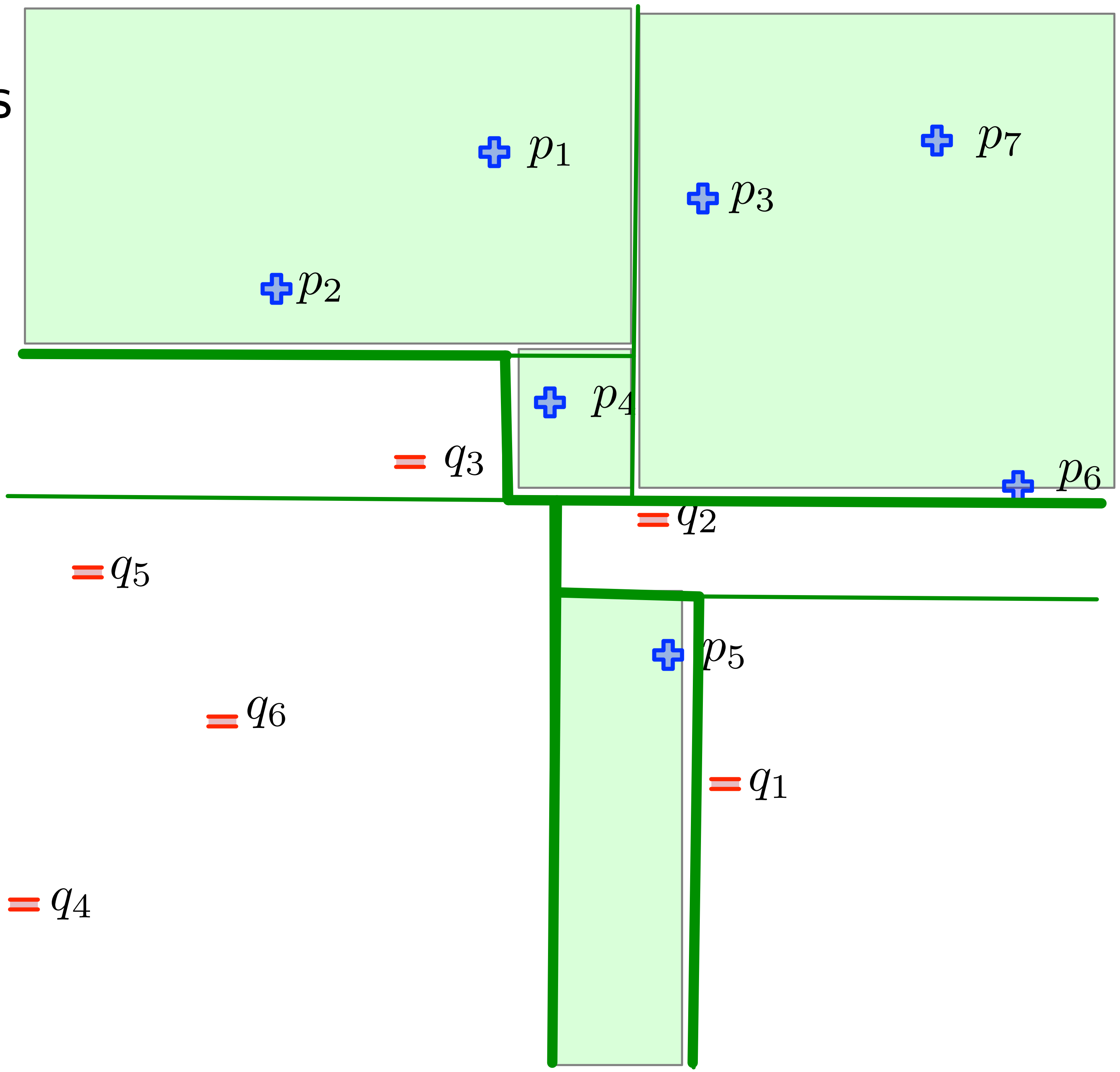
Many, many types of classifiers

- **KNN**: voting based on k nearest neighbors
- **decision trees**: kd-tree until uniform nodes



Many, many types of classifiers

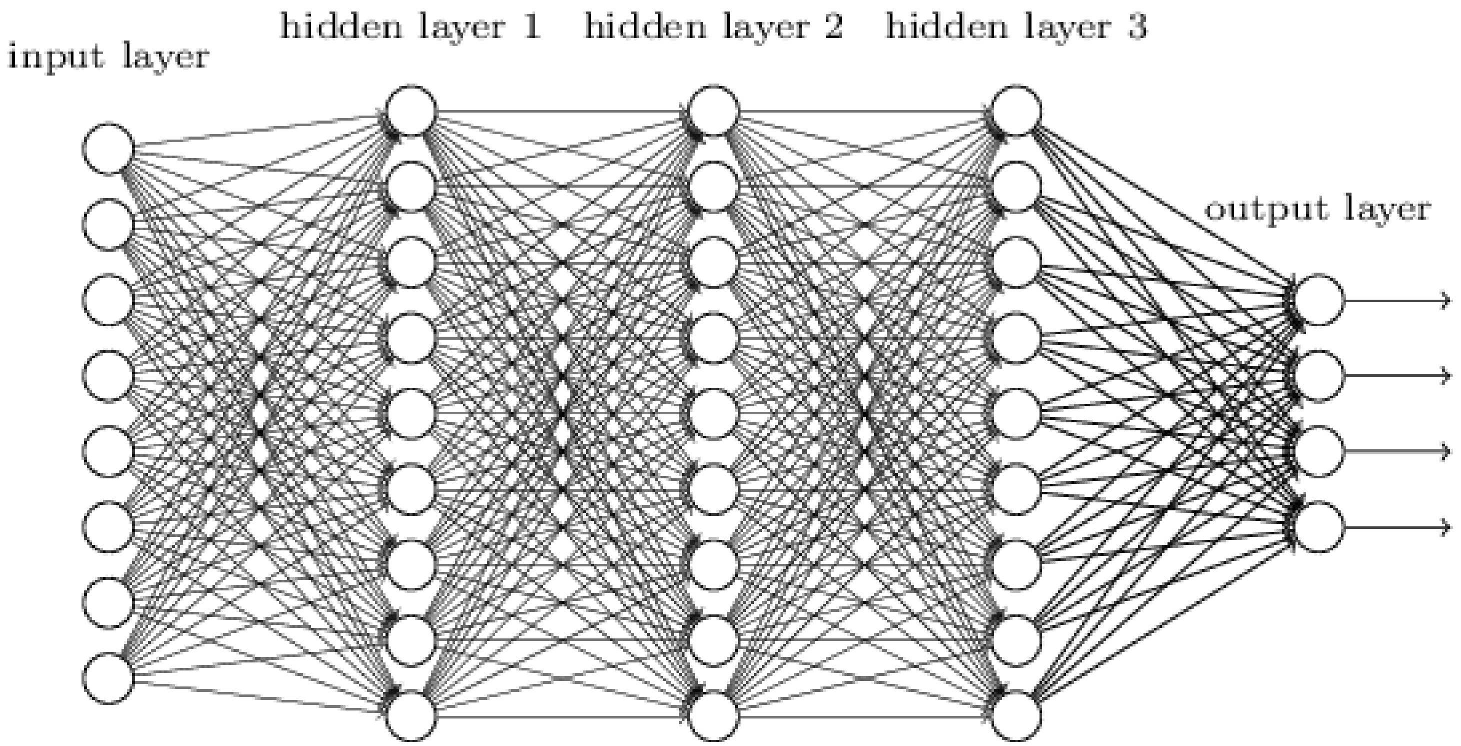
- **KNN**: voting based on k nearest neighbors
- **decision trees**: kd-tree until uniform nodes
- **random forests**: mixtures of decision trees

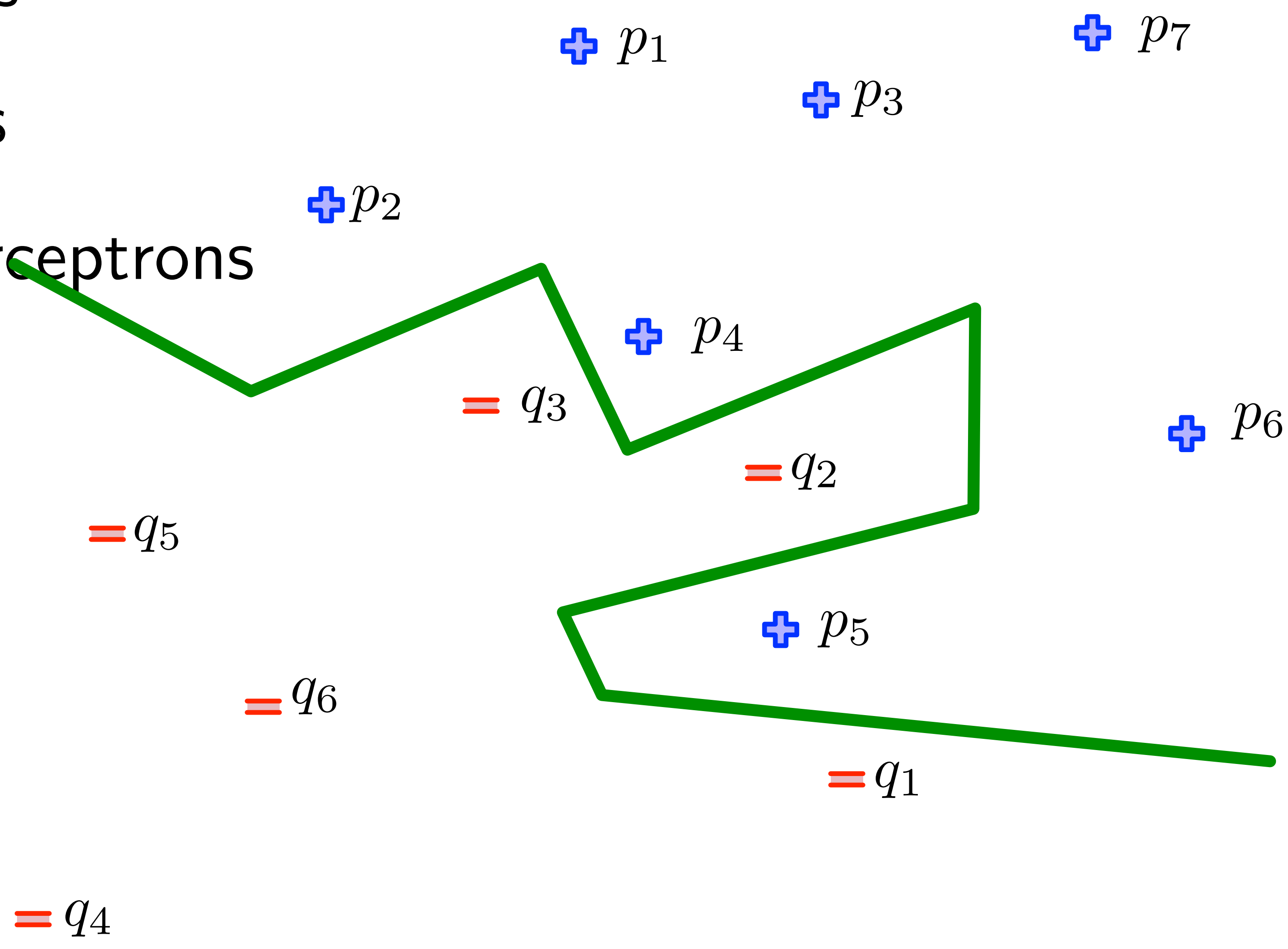


Many, many types of classifiers

- **KNN**: voting based on k nearest neighbors
- **decision trees**: kd-tree until uniform nodes
- **random forests**: mixtures of decision trees
- **deep neural networks**: many layers of perceptrons

piecewise-linear classifiers



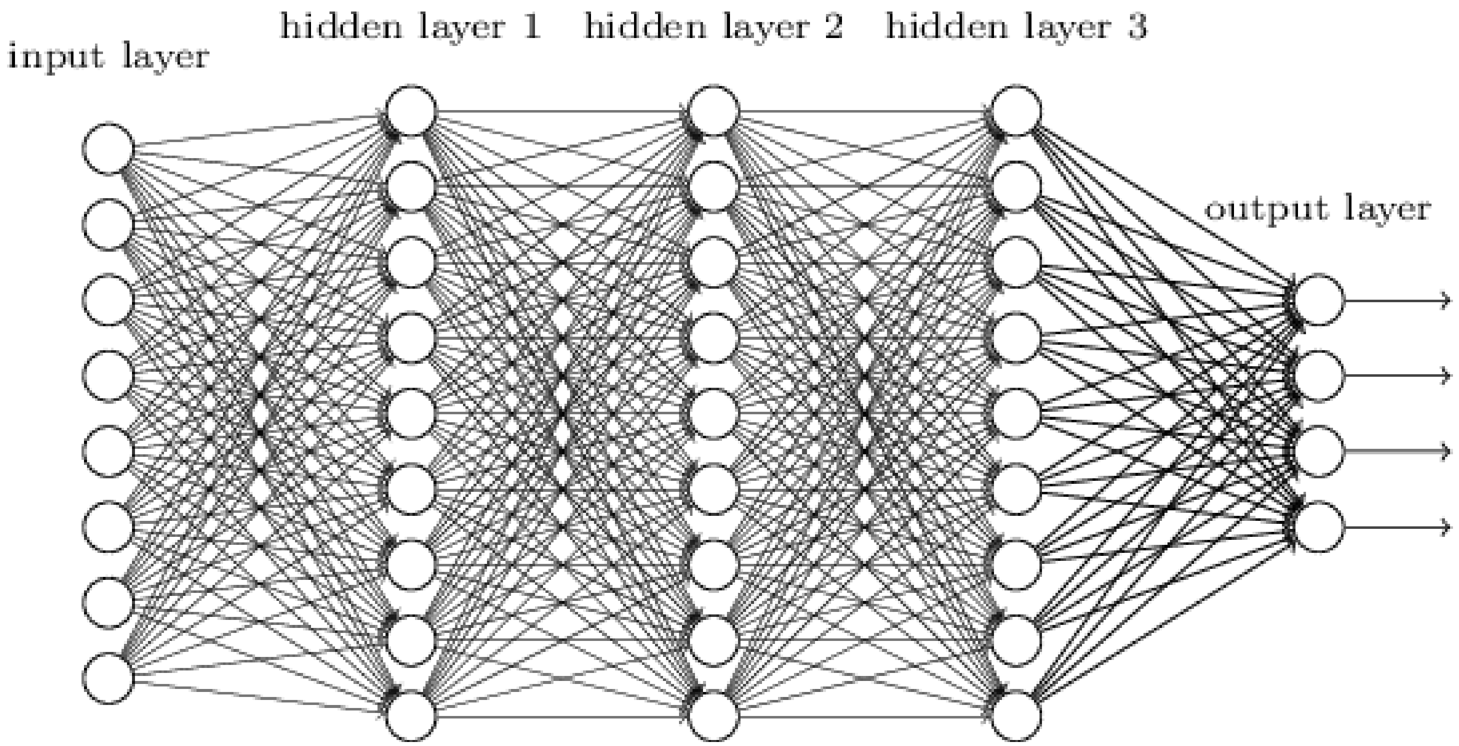


Many, many types of classifiers

- **KNN**: voting based on k nearest neighbors
- **decision trees**: kd-tree until uniform nodes
- **random forests**: mixtures of decision trees
- **deep neural networks**: many layers of perceptrons piecewise-linear classifiers

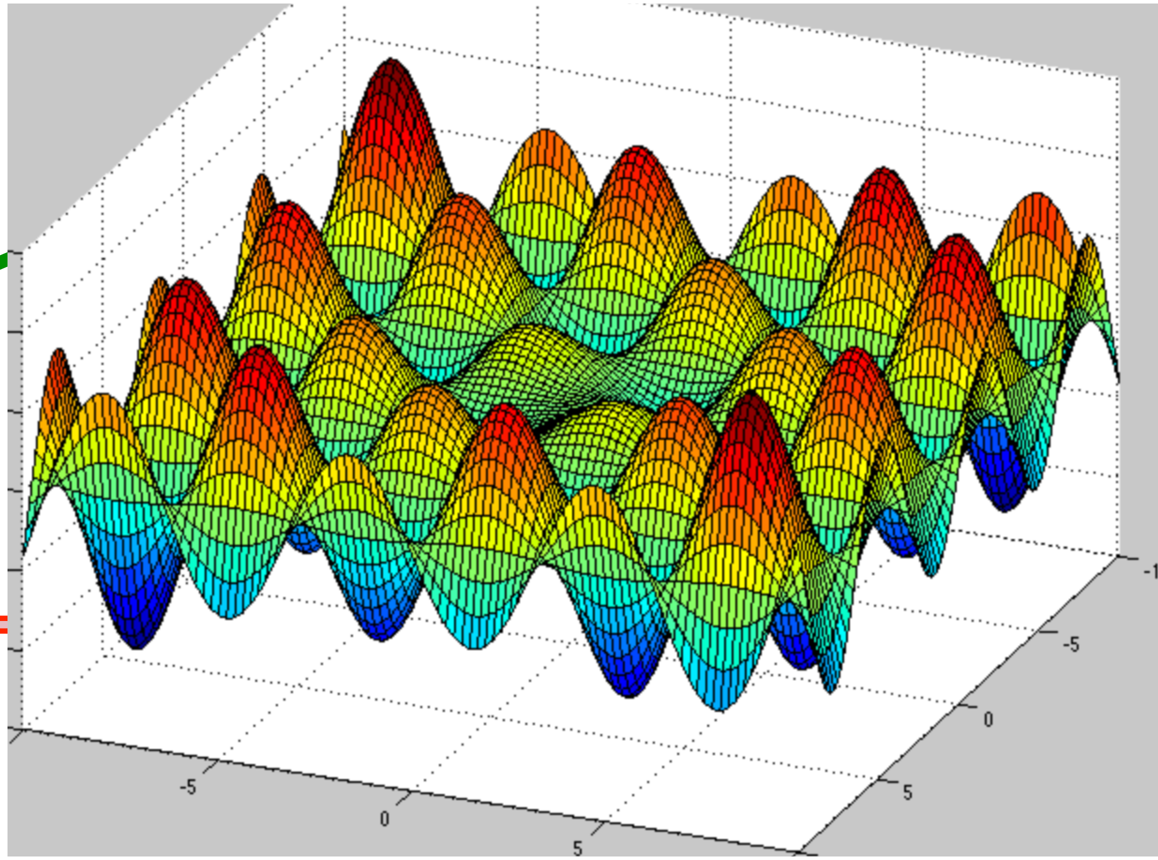
- non-convex function $f_X(w)$
- lots of data!
- lots of computers!

p_3



q_5

q_4



Many, many types of classifiers

- **KNN**: voting based on k nearest neighbors

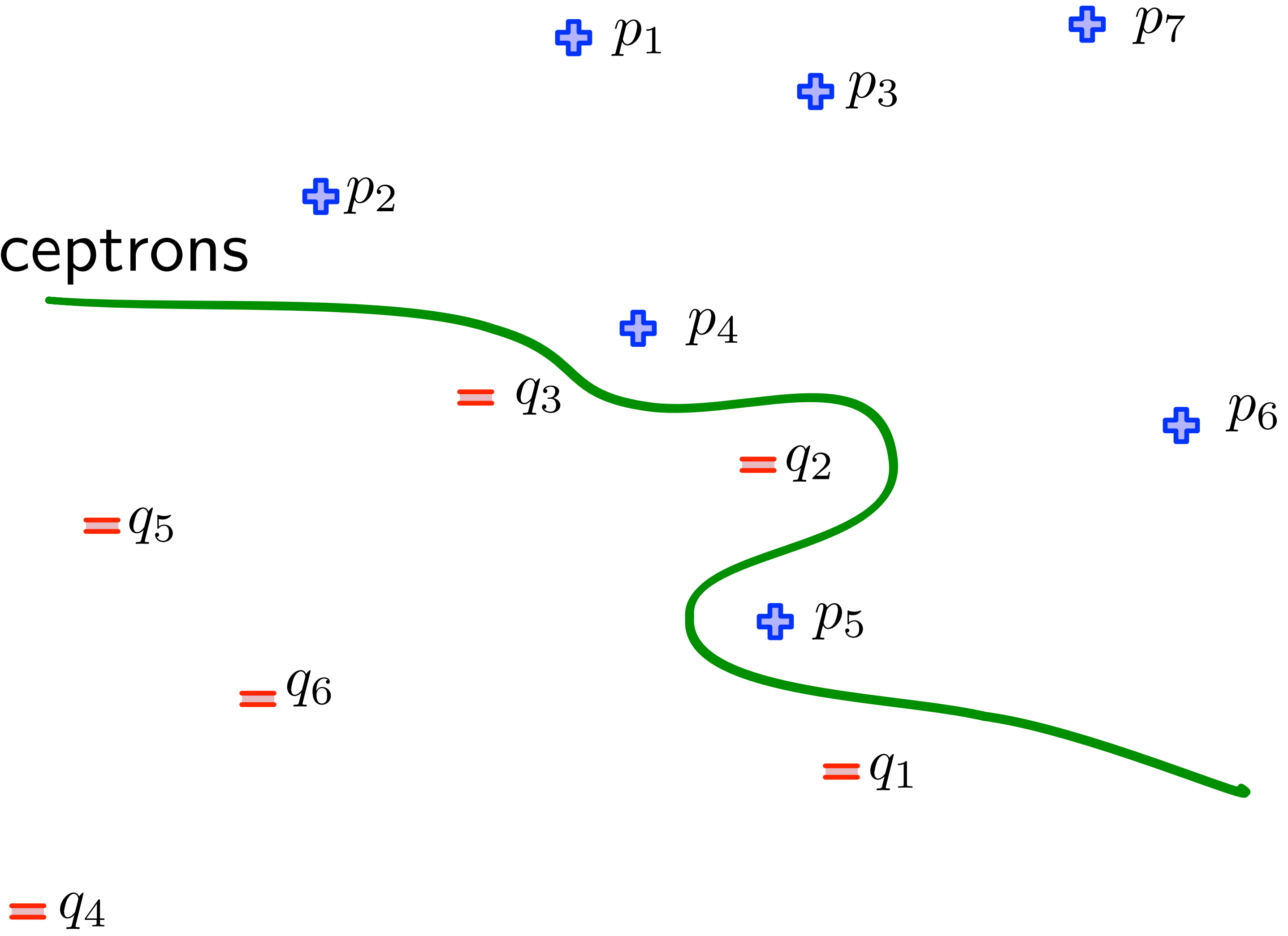
- **decision trees**: kd-tree until uniform nodes

- **random forests**: mixtures of decision trees

- **deep neural networks**: many layers of perceptrons

piecewise-linear classifiers

- ...



Where can geometry help today?

- non-convex optimization, function classes

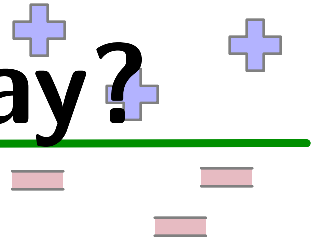
- generalizations of VC-dimension

- (implicit) regularization:
  what it means, when it occurs

- geometry of data ... e.g., adversarial data

- bias in data, algorithms

Where can geometry help today?



- non-convex optimization, function classes

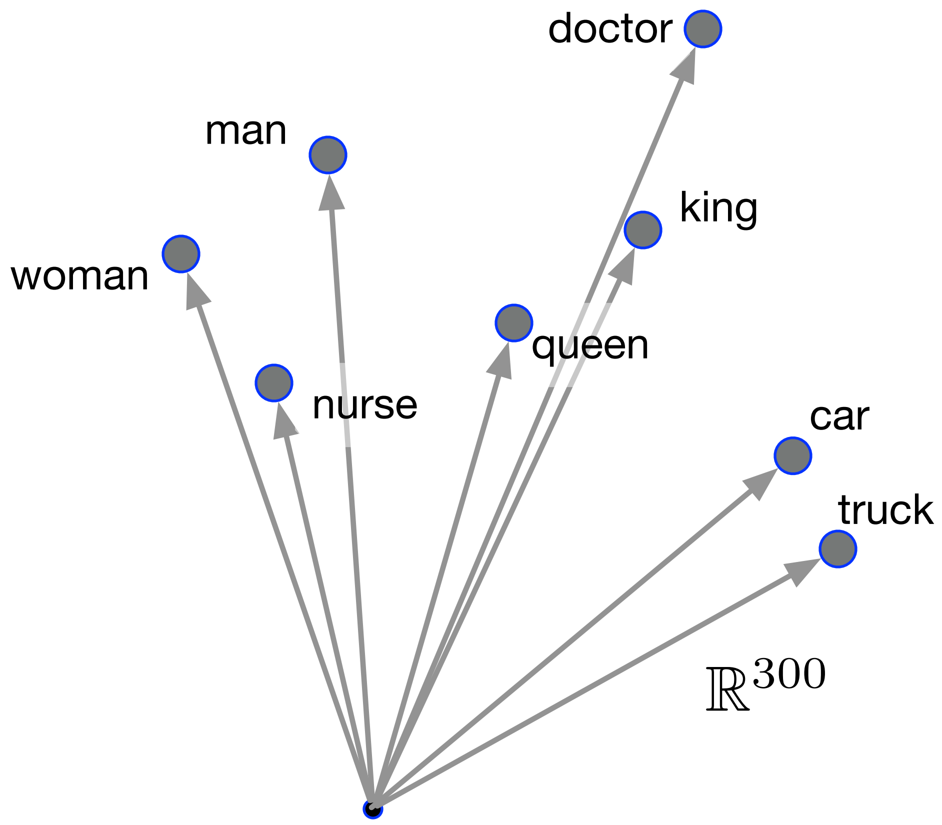
- generalizations of VC-dimension

- (implicit) regularization:
what it means, when it occurs

- geometry of data ... e.g., adversarial data

- bias in data, algorithms

word vector embeddings
word2vec, GloVe, ...



Where can geometry help today?

- non-convex optimization, function classes

- generalizations of VC-dimension

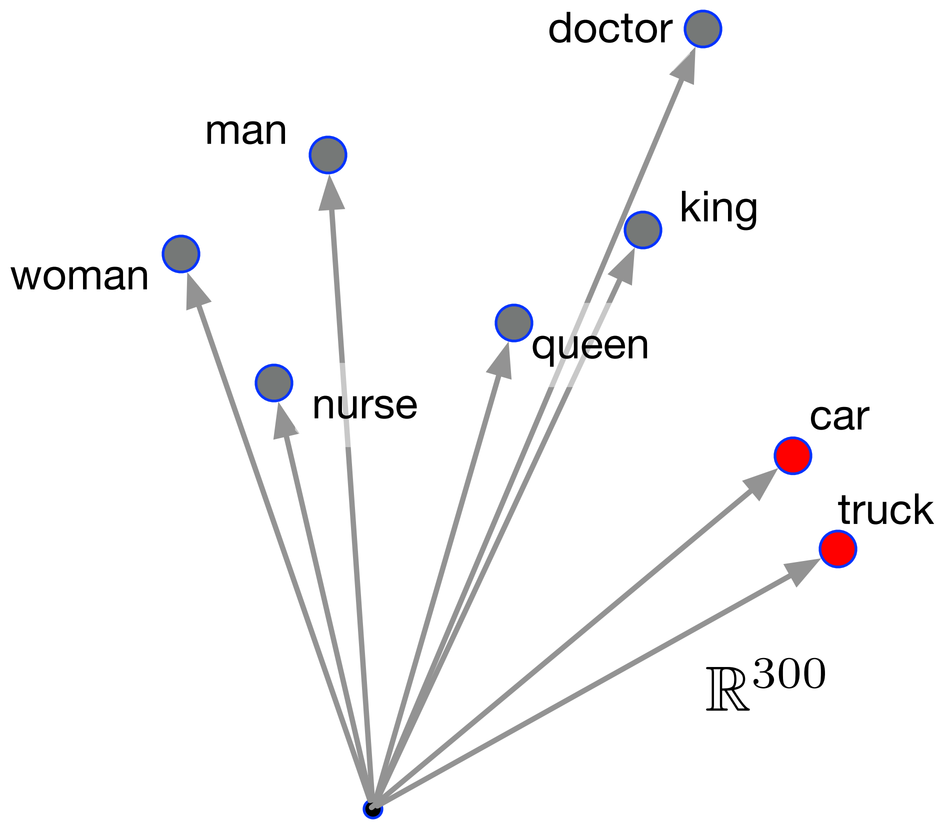
- (implicit) regularization:
what it means, when it occurs

- geometry of data ... e.g., adversarial data

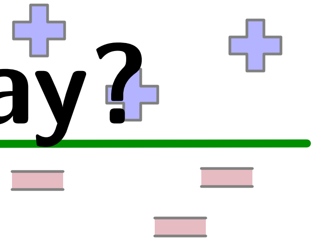
- bias in data, algorithms

word vector embeddings

word2vec, GloVe, ...

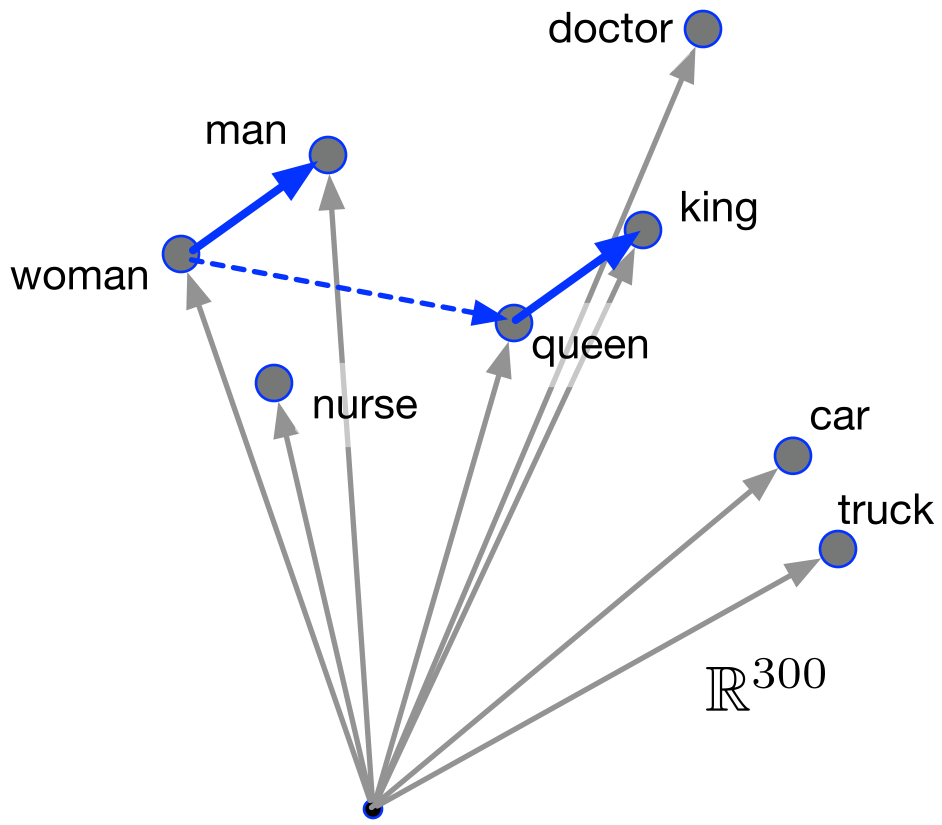


Where can geometry help today?



- non-convex optimization, function classes
- generalizations of VC-dimension
- (implicit) regularization:
what it means, when it occurs
- geometry of data … e.g., adversarial data
- bias in data, algorithms

word vector embeddings
word2vec, GloVe, …



Where can geometry help today?

- non-convex optimization, function classes

- generalizations of VC-dimension

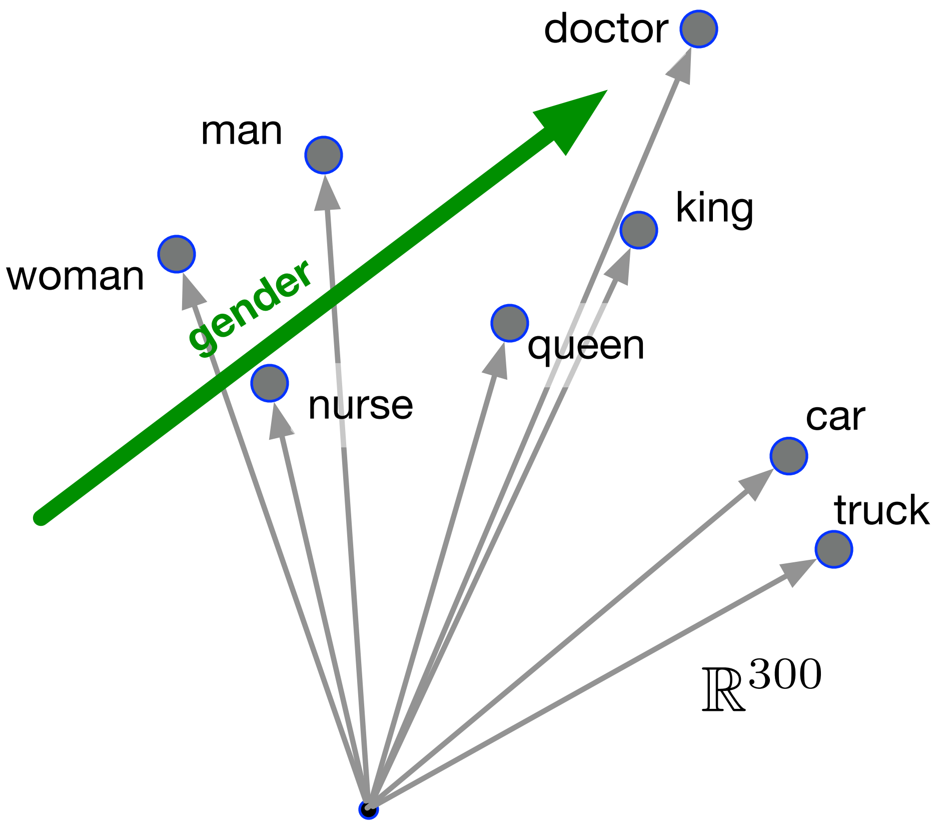
- (implicit) regularization:
  what it means, when it occurs

- geometry of data … e.g., adversarial data

- bias in data, algorithms

word vector embeddings

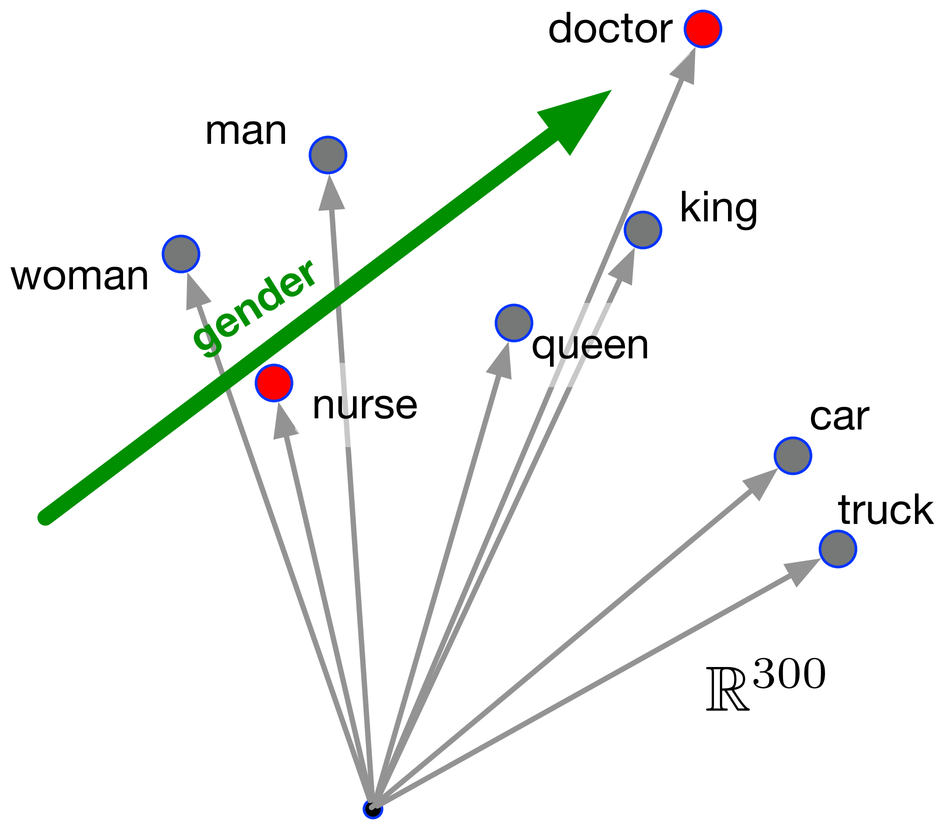
word2vec, GloVe, …



Where can geometry help today?

- non-convex optimization, function classes
- generalizations of VC-dimension
- (implicit) regularization:
what it means, when it occurs
- geometry of data ... e.g., adversarial data
- bias in data, algorithms

word vector embeddings
word2vec, GloVe, ...



Where can geometry help today?

- non-convex optimization, function classes

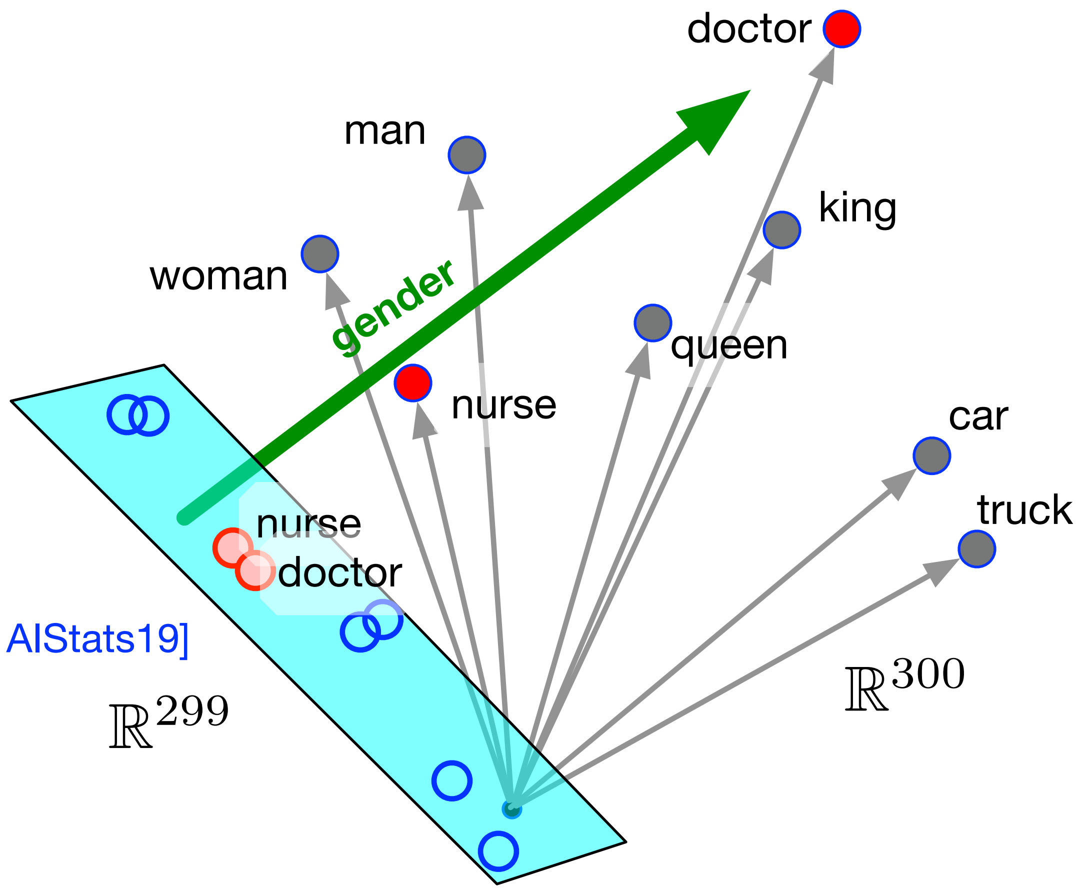
- generalizations of VC-dimension

- (implicit) regularization:
what it means, when it occurs

- geometry of data ... e.g., adversarial data

- bias in data, algorithms

word vector embeddings
word2vec, GloVe, ...

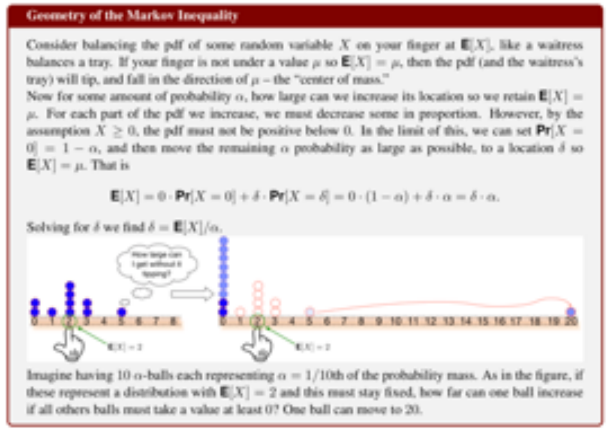


[Dev+P AIStats19]

MATHEMATICAL FOUNDATIONS FOR DATA ANALYSIS

JEFF M. PHILLIPS

http://www.cs.utah.edu/~jeffp/M4D/M4D.html



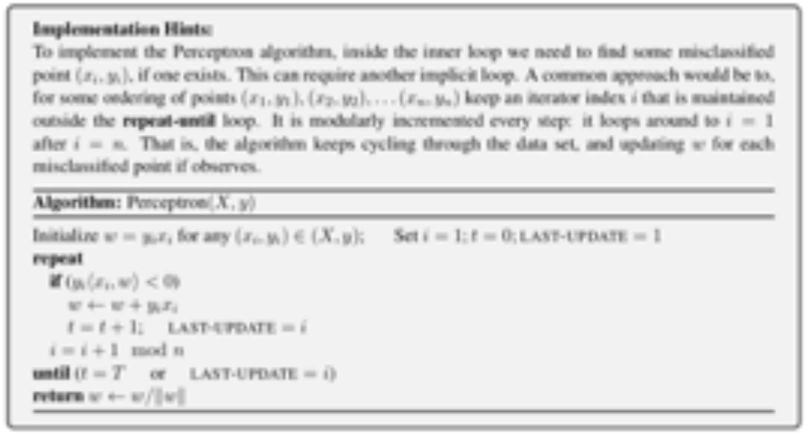
Geometry of the Markov Inequality

Consider balancing the pdf of some random variable X on your finger at $\mathbf{E}[X]$, like a waitress balances a tray. If your finger is not under a value μ so $\mathbf{E}[X] = \mu$, then the pdf (and the waitress's tray) will tip, and fall in the direction of μ – the "center of mass."
Now for some amount of probability α, how large can we increase its location so we retain $\mathbf{E}[X] = \mu$. For each part of the pdf we increase, we must decrease some in proportion. However, by the assumption $X \geq 0$, the pdf must not be positive below 0. In the limit of this, we can set $\mathbf{Pr}[X = 0] = 1 - \alpha$, and then move the remaining α probability as large as possible, to a location δ so $\mathbf{E}[X] = \mu$. That is

$$\mathbf{E}[X] = 0 \cdot \mathbf{Pr}[X = 0] + \delta \cdot \mathbf{Pr}[X = \delta] = 0 \cdot (1 - \alpha) + \delta \cdot \alpha = \delta \cdot \alpha.$$

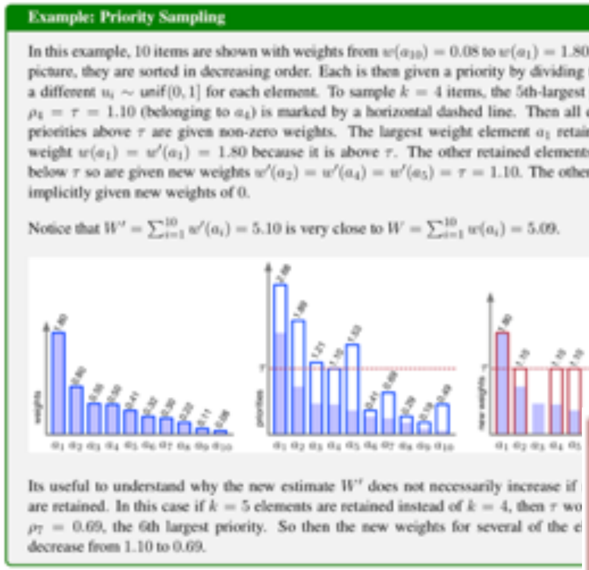
Solving for δ we find $\delta = \mathbf{E}[X]/\alpha$.

Imagine having 10 α-balls each representing $\alpha = 1/10$th of the probability mass. As in the figure, if these represent a distribution with $\mathbf{E}[X] = 2$ and this must stay fixed, how far can one ball increase if all others balls must take a value at least 0? One ball can move to 20.



Implementation Hints:
To implement the Perceptron algorithm, inside the inner loop we need to find some misclassified point (x_i, y_i), if one exists. This can require another implicit loop. A common approach would be to, for some ordering of points $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ keep an iterator index i that is maintained outside the **repeat-until** loop. It is modularly incremented every step: it loops around to $i = 1$ after $i = n$. That is, the algorithm keeps cycling through the data set, and updating w for each misclassified point if observes.

```
Algorithm: Perceptron$(X, y)$
Initialize $w = y_i x_i$ for any $(x_i, y_i) \in (X, y)$;     Set $i = 1$; $t = 0$; LAST-UPDATE $= 1$
repeat
  if $(y_i \langle x_i, w\rangle < 0)$
    $w \leftarrow w + y_i x_i$
    $t = t + 1$;    LAST-UPDATE $= i$
  $i = i + 1 \mod n$
until $(t = T$    or    LAST-UPDATE $= i)$
return $w \leftarrow w / \|w\|$
```



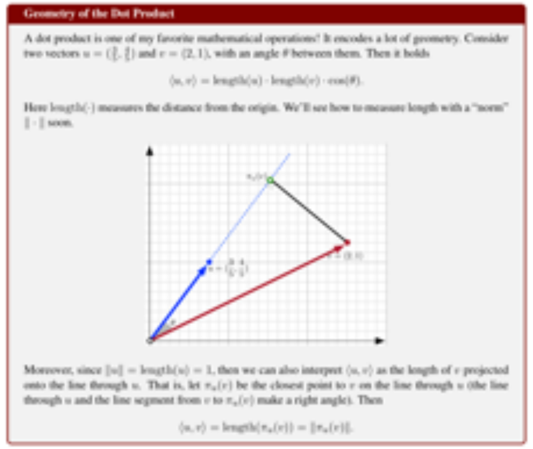
Example: Priority Sampling

In this example, 10 items are shown with weights from $w(a_{10}) = 0.08$ to $w(a_1) = 1.80$. For a clearer picture, they are sorted in decreasing order. Each is then given a priority by dividing the weight by a different $u_i \sim \mathsf{unif}(0, 1]$ for each element. To sample $k = 4$ items, the 5th-largest priority value $\rho_4 = \tau = 1.10$ (belonging to a_4) is marked by a horizontal dashed line. Then all elements with priorities above τ are given non-zero weights. The largest weight element a_1 retains its original weight $w(a_1) = w'(a_1) = 1.80$ because it is above τ. The other retained elements have weight below τ so are given new weights $w'(a_2) = w'(a_4) = w'(a_5) = \tau = 1.10$. The other elements are implicitly given new weights of 0.

Notice that $W' = \sum_{i=1}^{10} w'(a_i) = 5.10$ is very close to $W = \sum_{i=1}^{10} w(a_i) = 5.09$.

weights
1.80
0.80
0.55
0.50
0.41
0.32
0.30
0.22
0.11
0.08
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
priorities
τ
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
new weights
τ
1.80
1.10
1.10
1.10
a_1 a_2 a_3 a_4 a_5

Its useful to understand why the new estimate W' does not necessarily increase if are retained. In this case if $k = 5$ elements are retained instead of $k = 4$, then τ wo $\rho_7 = 0.69$, the 6th largest priority. So then the new weights for several of the e decrease from 1.10 to 0.69.



Geometry of the Dot Product

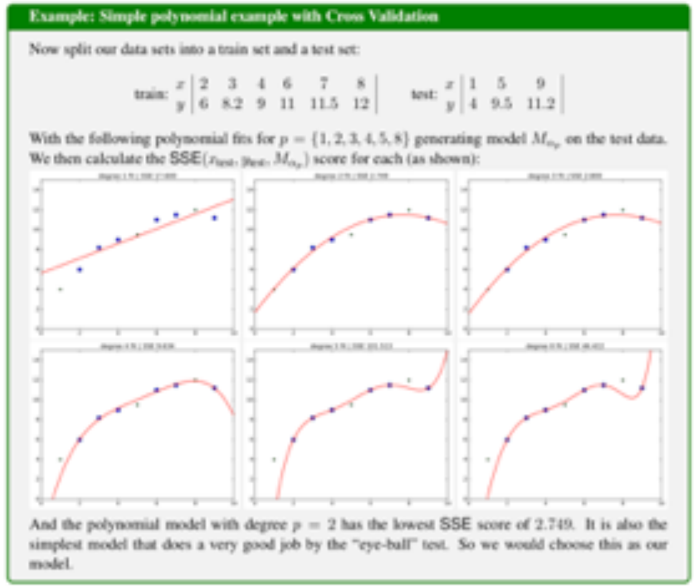
A dot product is one of my favorite mathematical operations! It encodes a lot of geometry. Consider two vectors $u = (\frac{3}{5}, \frac{4}{5})$ and $v = (2, 1)$, with an angle θ between them. Then it holds

$$\langle u, v\rangle = \mathsf{length}(u) \cdot \mathsf{length}(v) \cdot \cos(\theta).$$

Here $\mathsf{length}(\cdot)$ measures the distance from the origin. We'll see how to measure length with a "norm" $\|\cdot\|$ soon.

Moreover, since $\|u\| = \mathsf{length}(u) = 1$, then we can also interpret $\langle u, v\rangle$ as the length of v projected onto the line through u. That is, let $\pi_u(v)$ be the closest point to v on the line through u (the line through u and the line segment from v to $\pi_u(v)$ make a right angle). Then

$$\langle u, v\rangle = \mathsf{length}(\pi_u(v)) = \|\pi_u(v)\|.$$



Example: Simple polynomial example with Cross Validation

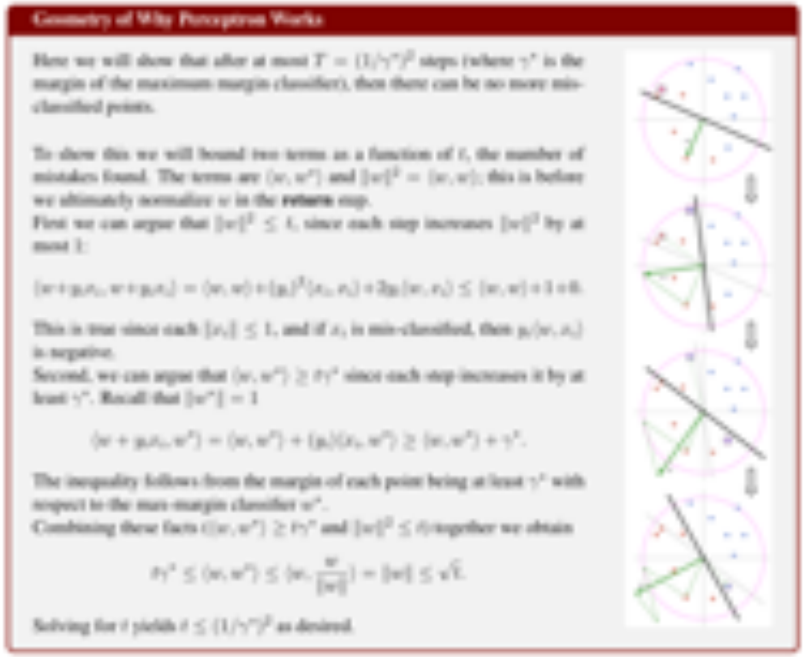
Now split our data sets into a train set and a test set:

train:						
x	2	3	4	6	7	8
y	6	8.2	9	11	11.5	12

test:			
x	1	5	9
y	4	9.5	11.2

With the following polynomial fits for $p = \{1, 2, 3, 4, 5, 8\}$ generating model M_{α_p} on the test data. We then calculate the $\mathsf{SSE}(x_{\text{test}}, y_{\text{test}}, M_{\alpha_p})$ score for each (as shown):

And the polynomial model with degree $p = 2$ has the lowest SSE score of 2.749. It is also the simplest model that does a very good job by the "eye-ball" test. So we would choose this as our model.



Geometry of Why Perceptron Works

Here we will show that after at most $T = (1/\gamma^*)^2$ steps (where γ^* is the margin of the maximum margin classifier), then there can be no more mis-classified points.

To show this we will bound two terms as a function of t, the number of mistakes found. The terms are $\langle w, w^*\rangle$ and $\|w\|^2 = \langle w, w\rangle$; this is before we ultimately normalize w in the **return** step.
First we can argue that $\|w\|^2 \leq t$, since each step increases $\|w\|^2$ by at most 1:

$$\langle w + y_i x_i, w + y_i x_i\rangle = \langle w, w\rangle + (y_i)^2\langle x_i, x_i\rangle + 2y_i\langle w, x_i\rangle \leq \langle w, w\rangle + 1 + 0.$$

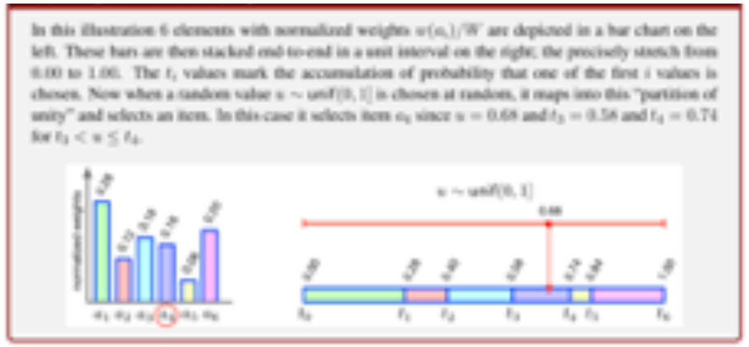
This is true since each $\|x_i\| \leq 1$, and if x_i is mis-classified, then $y_i\langle w, x_i\rangle$ is negative.
Second, we can argue that $\langle w, w^*\rangle \geq t\gamma^*$ since each step increases it by at least γ^*. Recall that $\|w^*\| = 1$

$$\langle w + y_i x_i, w^*\rangle = \langle w, w^*\rangle + (y_i)\langle x_i, w^*\rangle \geq \langle w, w^*\rangle + \gamma^*.$$

The inequality follows from the margin of each point being at least γ^* with respect to the max-margin classifier w^*.
Combining these facts ($\langle w, w^*\rangle \geq t\gamma^*$ and $\|w\|^2 \leq t$) together we obtain

$$t\gamma^* \leq \langle w, w^*\rangle \leq \langle w, \frac{w}{\|w\|}\rangle = \|w\| \leq \sqrt{t}.$$

Solving for t yields $t \leq (1/\gamma^*)^2$ as desired.



In this illustration 6 elements with normalized weights $w(a_i)/W$ are depicted in a bar chart on the left. These bars are then stacked end-to-end in a unit interval on the right; the precisely stretch from 0.00 to 1.00. The t_i values mark the accumulation of probability that one of the first i values is chosen. Now when a random value $u \sim \mathsf{unif}(0, 1]$ is chosen at random, it maps into this "partition of unity" and selects an item. In this case it selects item a_4 since $u = 0.68$ and $t_3 = 0.58$ and $t_4 = 0.74$ for $t_3 < u \leq t_4$.

normalized weights
0.28
0.12
0.18
0.16
0.06
0.20
a_1 a_2 a_3 a_4 a_5 a_6
$u \sim \mathsf{unif}(0, 1]$
0.68
0.00
0.28
0.40
0.58
0.74
0.80
1.00
t_0 t_1 t_2 t_3 t_4 t_5 t_6