Linear classifiers

Bayes classifier (for binary classification)

- Probability distribution P over $\mathcal{X} \times \{0,1\}$; let $(X,Y) \sim P$.
- Think of P as being comprised of two parts.
  1. Marginal distribution of X (a distribution over $\mathcal{X}$).
  2. Conditional distribution of Y given $X = x$, for each $x \in \mathcal{X}$:

$$\eta(x) \ := \ P(Y = 1 \mid X = x)\,.$$

- The optimal classifier with smallest error rate (i.e., *Bayes classifier*) is

$$f^*(x) \ = \ \begin{cases} 0 & \text{if } \eta(x) \le 1/2 \\ 1 & \text{if } \eta(x) > 1/2\,. \end{cases}$$

- Only depends on x through (the sign of) the **log-odds function** at x:

$$x \ \mapsto \ \log \frac{\eta(x)}{1 - \eta(x)} \ \in \ [-\infty, +\infty]\,.$$

(If positive, then predict 1; else, predict 0.)

Logistic regression

Suppose feature space is $\mathcal{X} = \mathbb{R}^d$.

Logistic regression: statistical model for $Y \mid \boldsymbol{X} = \boldsymbol{x}$ for each $\boldsymbol{x} \in \mathbb{R}^d$:

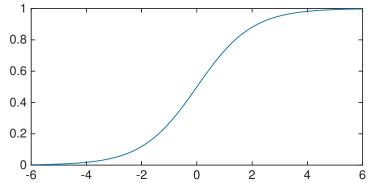
$$\mathcal{P} \ = \ \left\{ P_{(\beta_0, \boldsymbol{\beta})} : \beta_0 \in \mathbb{R},\, \boldsymbol{\beta} \in \mathbb{R}^d \right\},$$

where

$$\eta_{(\beta_0,\boldsymbol{\beta})}(\boldsymbol{x}) \ := \ P_{(\beta_0,\boldsymbol{\beta})}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) \ = \ \operatorname{logistic}(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle)$$

and

$$\operatorname{logistic}(z) \ := \ \frac{1}{1 + e^{-z}} \ = \ \frac{e^z}{1 + e^z}\,.$$



(Note: Logistic regression does not specify marginal distribution for $\boldsymbol{X}$.)

Parameter estimation

Given data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ (regarded as an iid sample), MLE for $(\beta_0, \boldsymbol{\beta})$ is

$$\begin{aligned}(\hat\beta_0, \hat{\boldsymbol{\beta}}) \ &= \ \mathop{\arg\max}_{\beta_0 \in \mathbb{R},\, \boldsymbol{\beta} \in \mathbb{R}^d} \ \log \prod_{i=1}^n \eta_{(\beta_0,\boldsymbol{\beta})}(\boldsymbol{x}_i)^{y_i} (1 - \eta_{(\beta_0,\boldsymbol{\beta})}(\boldsymbol{x}_i))^{1-y_i} \\ &\ \ \vdots \\ &= \ \mathop{\arg\max}_{\beta_0 \in \mathbb{R},\, \boldsymbol{\beta} \in \mathbb{R}^d} \ \sum_{i=1}^n y_i(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle) - \log(1 + \exp(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}_i \rangle))\,.\end{aligned}$$

- No closed-form solution for MLE.
- Nevertheless, there are efficient algorithms that obtain an approximate maximizer of the MLE objective function (which is a function of $(\beta_0, \boldsymbol{\beta})$).

Log-odds function and classifier

Log-odds function of $P_{(\beta_0,\boldsymbol{\beta})}$ is

$$\boldsymbol{x} \;\mapsto\; \log \frac{\eta_{(\beta_0,\boldsymbol{\beta})}(\boldsymbol{x})}{1-\eta_{(\beta_0,\boldsymbol{\beta})}(\boldsymbol{x})} \;=\; \log\left(\frac{\frac{\exp(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle)}{1+\exp(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle)}}{\frac{1}{1+\exp(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle)}} \right) \;=\; \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle \,,$$

which is an **affine function**.

Bayes classifier for $P_{(\beta_0,\boldsymbol{\beta})}$ is

$$\boldsymbol{x} \;\mapsto\; \begin{cases} 0 & \text{if } \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle \leq 0\,, \\ 1 & \text{if } \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle > 0\,. \end{cases}$$

Such classifiers are called **linear classifiers**.

Linear classifiers

A **linear classifier** is specified by a *weight vector* $\boldsymbol{w} \in \mathbb{R}^d$ and *threshold* $t \in \mathbb{R}$:

$$f_{\boldsymbol{w},t}(\boldsymbol{x}) \;:=\; \begin{cases} 0 & \text{if } \langle \boldsymbol{w}, \boldsymbol{x}\rangle \leq t\,, \\ 1 & \text{if } \langle \boldsymbol{w}, \boldsymbol{x}\rangle > t\,. \end{cases}$$

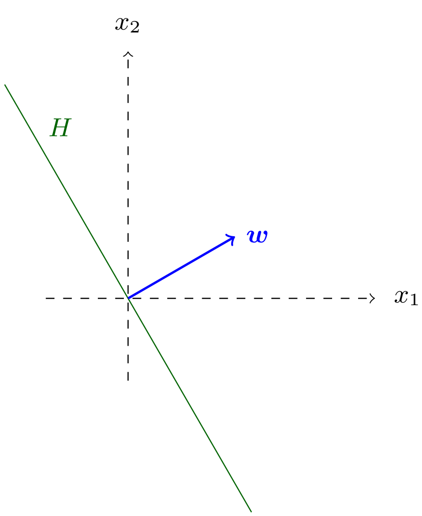
Interpretation: does a linear combination of input features exceed a threshold?

$$\langle \boldsymbol{w}, \boldsymbol{x}\rangle \;=\; \sum_{i=1}^{d} w_i x_i \;\overset{?}{>}\; t\,.$$

Translation from logistic regression parameters $(\beta_0, \boldsymbol{\beta})$:

$$\boldsymbol{w} \;=\; \boldsymbol{\beta}\,, \qquad t \;=\; -\beta_0\,.$$

Geometry of linear classifiers



A **hyperplane** in $\mathbb{R}^d$ is a linear subspace of dimension $d-1$.

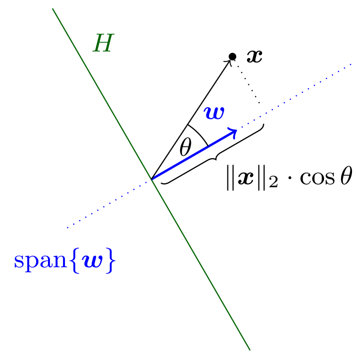
- A $\mathbb{R}^2$-hyperplane is a line.
- A $\mathbb{R}^3$-hyperplane is a plane.
- As a linear subspace, a hyperplane always contains the origin.

A hyperplane H can be specified by a (non-zero) **normal vector** $\boldsymbol{w} \in \mathbb{R}^d$.

The hyperplane with normal vector $\boldsymbol{w}$ is the set of points orthogonal to $\boldsymbol{w}$:

$$H \;=\; \left\{ \boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{w}, \boldsymbol{x}\rangle = 0 \right\}.$$

The hyperplane and the point



Projection of $\boldsymbol{x}$ onto $\mathrm{span}\{\boldsymbol{w}\}$ (a line) has coordinate

$$\|\boldsymbol{x}\|_2 \cdot \cos(\theta)$$

where

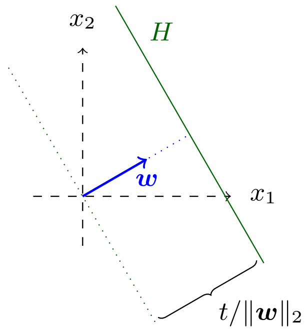
$$\cos\theta \;=\; \frac{\langle \boldsymbol{w}, \boldsymbol{x}\rangle}{\|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2}\,.$$

(Distance to hyperplane is $\|\boldsymbol{x}\|_2 \cdot |\cos(\theta)|$.)

Which side of the hyperplane (oriented by $\boldsymbol{w}$)?

$$\|\boldsymbol{x}\|_2 \cdot \cos(\theta) > 0 \iff \langle \boldsymbol{w}, \boldsymbol{x}\rangle > 0 \iff \boldsymbol{x} \text{ on same side of } H \text{ as } \boldsymbol{w}$$

Affine hyperplanes



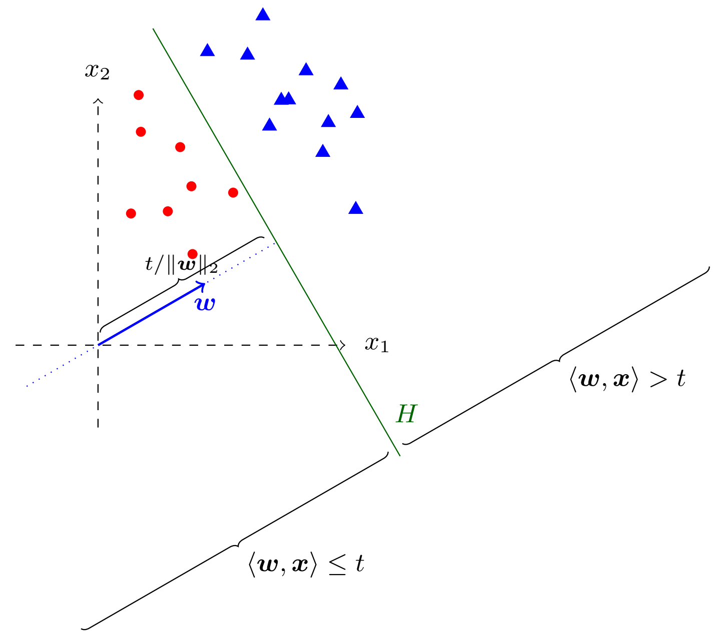
An **affine hyperplane** H is a hyperplane that may be shifted away from the origin.

Can be specified with a **normal vector** $\boldsymbol{w}$ and a **threshold** $t \in \mathbb{R}$:

$$H = \left\{ \boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{w}, \boldsymbol{x} \rangle = t \right\}.$$

side of affine hyperplane that x is on $\equiv$ classification of x by $f_{\boldsymbol{w},t}$

Linear classifiers



Learning linear classifiers

Even if the Bayes classifier is not a linear classifier, we can hope that it has a good linear approximation.

Goal: learn a linear classifier $f_{\hat{\boldsymbol{w}},\hat{t}}$ using iid sample S such that

$$\underbrace{\operatorname{err}(f_{\hat{\boldsymbol{w}},\hat{t}})}_{\text{error rate of your classifier}} \quad - \quad \underbrace{\min_{\boldsymbol{w},t} \operatorname{err}(f_{\boldsymbol{w},t})}_{\text{error rate of \textit{best} linear classifier}}$$

is as small as possible.

A natural approach is **"empirical risk minimization" (ERM)**: find a linear classifier $f_{\boldsymbol{w},t}$ with minimum *training error rate* (a.k.a. **empirical risk**):

$$\arg\min_{\boldsymbol{w},t} \operatorname{err}(f_{\boldsymbol{w},t}, S) = \arg\min_{\boldsymbol{w},t} \frac{1}{|S|} \sum_{(\boldsymbol{x},y)\in S} \mathbb{1}\{f_{\boldsymbol{w},t}(\boldsymbol{x}) \neq y\}.$$

Note: there is worry that ERM classifier will "overfit" the training data, but this is not a problem when (for example) $|S| \gg d$.

Empirical risk minimization

Unforunately, this is not possible in general.

- The following problem is NP-hard:

  input labeled examples S from $\mathbb{R}^d \times \{0, 1\}$ with promise that there is a linear classifier with training error rate 0.01.

  output a linear classifier with training error rate ≤ 0.49.

Potential saving grace:

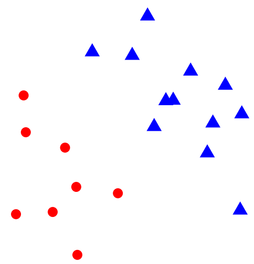
- Real-world problems we need to solve do not look like the encodings of difficult 3-SAT instances.

Linearly separable data

Suppose there is a linear classifier that perfectly classifies S:
i.e., for some $\boldsymbol{w}_\star \in \mathbb{R}^d$ and $t_\star \in \mathbb{R}$,

$$f_{\boldsymbol{w}_\star, t_\star}(\boldsymbol{x}) \;=\; y \quad \text{for all } (\boldsymbol{x}, y) \in S.$$

In this case, we say the training data is **linearly separable**.



Homogeneous linear classifiers

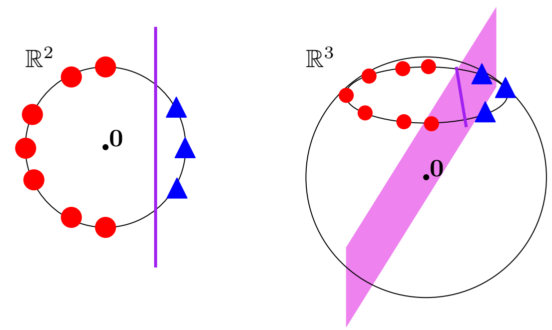
Homogeneous linear classifier: a linear classifier with threshold $t = 0$.

A crude but easy reduction via "lifting" to $\mathbb{R}^{d+1}$: map $\boldsymbol{x} \in \mathbb{R}^d$ to $\mathbb{R}^{d+1}$ with

$$\boldsymbol{\phi}(\boldsymbol{x}) \;:=\; (\boldsymbol{x}, 1) \;\in\; \mathbb{R}^{d+1},$$

and define $\tilde{\boldsymbol{w}} := (\boldsymbol{w}, -t) \in \mathbb{R}^{d+1}$.

Then linear classifier $f_{\boldsymbol{w},t}$ is the same as $f_{\tilde{\boldsymbol{w}}} \circ \boldsymbol{\phi}$,
where $f_{\tilde{\boldsymbol{w}}} = f_{\tilde{\boldsymbol{w}},0}$ is a homogeneous linear classifier in $\mathbb{R}^{d+1}$.



Finding a homogeneous linear separator

Problem: given training data S from $\mathbb{R}^d \times \{0,1\}$, determine whether or not there exists $\boldsymbol{w} \in \mathbb{R}^d$ such that

$$f_{\boldsymbol{w}}(\boldsymbol{x}) \;=\; y \quad \text{for all } (\boldsymbol{x}, y) \in S;$$

(and find such a vector if one exists).

- d variables: $\boldsymbol{w} \in \mathbb{R}^d$
- $|S|$ inequalities: for $(\boldsymbol{x}, y) \in S$,

$$\begin{aligned} \text{if } y = 0: &\quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle \;\le\; 0\,, \\ \text{if } y = 1: &\quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle \;>\; 0\,. \end{aligned}$$

Can be solved in polynomial time using algorithms for **linear programming** (e.g., ellipsoid algorithm, interior point).

If one exists, and the inequalities can be satisfied with some non-negligible "wiggle room", then there is a very **simple** algorithm that finds a solution: **Perceptron**.

Perceptron (Rosenblatt, 1958)

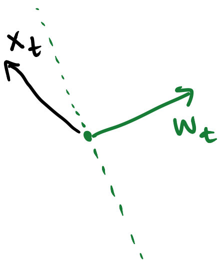
(Notationally simpler to use $\mathcal{Y} := \{-1,+1\}$ instead of $\{0,1\}$.)

Perceptron

input Labeled examples $S \subset \mathbb{R}^d \times \{-1,+1\}$.
initialize $\hat{\boldsymbol{w}}_1 := \boldsymbol{0}$.
for $t = 1, 2, \ldots,$ **do**
  if there is an example in S misclassified by $f_{\hat{\boldsymbol{w}}_t}$ **then**
    Let $(\boldsymbol{x}_t, y_t)$ be any such misclassified example.
    Update: $\hat{\boldsymbol{w}}_{t+1} := \hat{\boldsymbol{w}}_t + y_t \boldsymbol{x}_t$.
  else
    return $\hat{\boldsymbol{w}}_t$.
  end if
end for

Note 1: An example $(\boldsymbol{x}, y)$ is misclassified by $f_{\boldsymbol{w}}$ if $y\langle \boldsymbol{w}, \boldsymbol{x} \rangle \le 0$.
Note 2: If Perceptron terminates, then $f_{\hat{\boldsymbol{w}}_t}$ perfectly classifies the data!

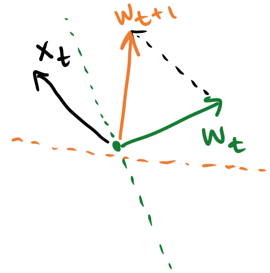
Perceptron



predict $a_t = -1$

correct label $y_t = +1$

Perceptron

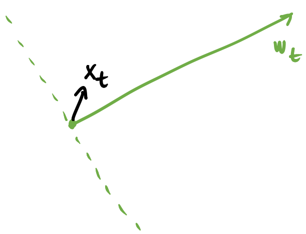


predict $a_t = -1$

correct label $y_t = +1$

$w_{t+1} := w_t + y_t x_t$

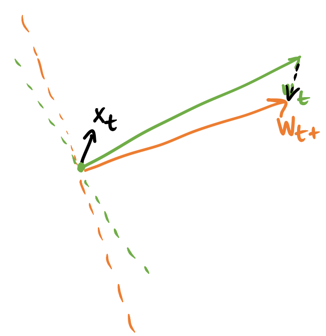
Perceptron



predict $a_t = +1$

correct label $y_t = -1$

Perceptron



predict $a_t = +1$

correct label $y_t = -1$

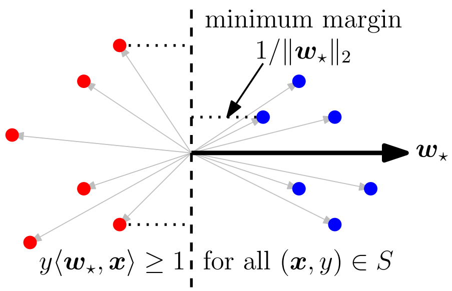
$w_{t+1} := w_t + y_t x_t$

When does Perceptron work?

Let $\boldsymbol{w}_\star \in \mathbb{R}^d$ be the shortest vector such that

$$y\langle \boldsymbol{w}_\star, \boldsymbol{x}\rangle \;\geq\; 1 \quad \text{for all } (\boldsymbol{x}, y) \in S$$

(assuming one exists).



Perceptron terminates quickly when $\|\boldsymbol{w}_\star\|_2$ is small,
i.e., when the **minimum margin** $1/\|\boldsymbol{w}_\star\|_2$ is large.

Perceptron convergence

Theorem: If $R := \max_{(\boldsymbol{x},y)\in S} \|\boldsymbol{x}\|_2$, and $\boldsymbol{w}_\star \in \mathbb{R}^d$ satisfies

$$y\langle \boldsymbol{w}_\star, \boldsymbol{x}\rangle \;\geq\; 1 \quad \text{for all } (\boldsymbol{x}, y) \in S\,,$$

then Perceptron halts after at most $\|\boldsymbol{w}_\star\|_2^2 \cdot R^2$ iterations.

Proof: Suppose Perceptron does not exit the for-loop in interation t.

Then there is a labeled example in S — which we call $(\boldsymbol{x}_t, y_t)$ — such that:

- $y_t\langle \boldsymbol{w}_\star, \boldsymbol{x}_t\rangle \geq 1$ (by definition of $\boldsymbol{w}_\star$)
- $y_t\langle \hat{\boldsymbol{w}}_t, \boldsymbol{x}_t\rangle \leq 0$ (since $f_{\hat{\boldsymbol{w}}_t}$ misclassifies the example)
- $\hat{\boldsymbol{w}}_{t+1} = \hat{\boldsymbol{w}}_t + y_t\boldsymbol{x}_t$ (since an update is made)

Use this information to (inductively) lower-bound

$$\cos(\text{angle between } \boldsymbol{w}_\star \text{ and } \hat{\boldsymbol{w}}_{t+1}) \;=\; \frac{\langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_{t+1}\rangle}{\|\boldsymbol{w}_\star\|_2\|\hat{\boldsymbol{w}}_{t+1}\|_2}\,.$$

Perceptron convergence

Goal: to lower-bound $\frac{\langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_{t+1}\rangle}{\|\boldsymbol{w}_\star\|_2\|\hat{\boldsymbol{w}}_{t+1}\|_2}$.

$$\langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_{t+1}\rangle \;=\; \langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_t + y_t\boldsymbol{x}_t\rangle \;=\; \langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_t\rangle + y_t\langle \boldsymbol{w}_\star, \boldsymbol{x}_t\rangle \;\geq\; \langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_t\rangle + 1\,.$$

Therefore, by induction (with $\hat{\boldsymbol{w}}_1 = \boldsymbol{0}$), $\langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_{t+1}\rangle \geq t$.

$$\|\hat{\boldsymbol{w}}_{t+1}\|_2^2 \;=\; \|\hat{\boldsymbol{w}}_t + y_t\boldsymbol{x}_t\|_2^2 \;=\; \|\hat{\boldsymbol{w}}_t\|_2^2 + 2y_t\langle \hat{\boldsymbol{w}}_t, \boldsymbol{x}_t\rangle + \|\boldsymbol{x}_t\|_2^2 \;\leq\; \|\hat{\boldsymbol{w}}_t\|_2^2 + R^2\,.$$

Therefore, by induction (with $\hat{\boldsymbol{w}}_1 = \boldsymbol{0}$), $\|\hat{\boldsymbol{w}}_{t+1}\|_2^2 \leq R^2 \cdot t$.

$$\cos(\text{angle between } \boldsymbol{w}_\star \text{ and } \hat{\boldsymbol{w}}_{t+1}) \;=\; \frac{\langle \boldsymbol{w}_\star, \hat{\boldsymbol{w}}_{t+1}\rangle}{\|\boldsymbol{w}_\star\|_2\|\hat{\boldsymbol{w}}_{t+1}\|_2} \;\geq\; \frac{t}{\|\boldsymbol{w}_\star\|_2 \cdot R \cdot \sqrt{t}}\,.$$

Since cosine is at most one, we conclude

$$t \;\leq\; \|\boldsymbol{w}_\star\|_2^2 \cdot R^2\,.$$

□

Online Perceptron

If data is not linearly separable, Perceptron runs forever!

Alternative: consider each example once, then halt.

Online Perceptron

input Labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

```
initialize  ŵ_1 := 0.
for t = 1, 2, ..., n do
   if y_t⟨ŵ_t, x_t⟩ ≤ 0 then
      ŵ_{t+1} := ŵ_t + y_t x_t.
   else
      ŵ_{t+1} := ŵ_t
   end if
end for
return  ŵ_{n+1}.
```

Final classifier $f_{\hat{\boldsymbol{w}}_{n+1}}$ is not necessarily a linear separator (even if one exists!).

Online learning

Online learning algorithms:

- Go through examples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots$ one-by-one.
  - Before seeing $(\boldsymbol{x}_t, y_t)$, learner has a "current" classifier $\hat{f}_t$ in hand.
  - Upon seeing $\boldsymbol{x}_t$, learner makes a prediction: $\hat{a}_t := \hat{f}_t(\boldsymbol{x}_t)$.
  - If $\hat{a}_t \neq y_t$, then the prediction was a "mistake".
  - Can now update $\hat{f}_t$ (to get $\hat{f}_{t+1}$) on the basis of all past examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t)$ (but usually just based on $(\boldsymbol{x}_t, y_t)$).

Theorem: If $R := \max_{t \in \{1,\ldots,n\}} \|\boldsymbol{x}_t\|_2$, and $\boldsymbol{w}_\star \in \mathbb{R}^d$ satisfies

$$y\langle \boldsymbol{w}_\star, \boldsymbol{x}_t \rangle \geq 1 \quad \text{for all } (\boldsymbol{x}_t, y_t),$$

then Online Perceptron makes at most $\|\boldsymbol{w}_\star\|_2^2 \cdot R^2$ mistakes (and updates).

What good is a mistake bound?

- Suppose you have an **online learning algorithm** that makes only a few mistakes on a sequence of iid random examples.
- Then the **sequence of classifiers** produced by the online learner are, on average, good at **predicting labels of random examples**.
- Thus, can combine the sequence of classifiers into **a single classifier** with **low true error rate**.

This is achieved via an **"Online-to-Batch" conversion**.

Online-to-Batch conversion

- Run online learning algorithm on sequence of examples

  $$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$$

  (in random order) to produce sequence of binary classifiers

  $$\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{n+1},$$

  ($\hat{f}_i \colon \mathcal{X} \to \{\pm 1\}$).
- Final classifier: majority vote over the $\hat{f}_i$'s

$$\hat{f}(\boldsymbol{x}) := \begin{cases} -1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(\boldsymbol{x}) \leq 0, \\ +1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(\boldsymbol{x}) > 0. \end{cases}$$

Note #1: many of the $\hat{f}_i$'s could be the same.
Note #2: many variants that improve this in some cases (e.g., only use last $n/2$ classifiers).

Voted-Perceptron (Freund & Schapire, 1999)

Voted-Perceptron = Online Perceptron with Online-to-Batch conversion.

Classifying OCR digits (digit '9' $\to$ class $+1$; all other digits $\to$ class -1).
$n = 60000$ (about 6000 are of class $+1$), presented in a random order.

# passes	0.1	1	2	3	4	10
err(OP, test)	0.079	0.064	0.057	0.063	0.058	0.059
err(VP, test)	0.045	0.039	0.038	0.038	0.038	0.037

(**In practice**: Making multiple passes through the data can sometimes help!)

Key takeaways

1. Logistic regression model, and structure/geometry of linear classifiers, including "lifting" trick for homogeneous linear classifiers.
2. High-level idea of empirical risk minimization for linear classifiers and intractability.
3. Concept of linear separability, two approaches to find a linear separator (linear programming and Perceptron).
4. Online Perceptron; online-to-batch conversion via voting.