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

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

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

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

(Logistic regression does not specify marginal distribution for $X$.)

Logistic regression

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

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

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

where

$$\eta(\beta_0, \beta)(x) := P(\beta_0, \beta)(Y = 1 \mid X = x) = \text{logistic}(\beta_0 + \langle \beta, x \rangle)$$

and

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

(Note: Logistic regression does not specify marginal distribution for $X$.)

Parameter estimation

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

$$\left( \hat{\beta}_0, \hat{\beta} \right) = \arg \max_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^d} \log \prod_{i=1}^n \eta(\beta_0, \beta)(x_i)^{y_i} (1 - \eta(\beta_0, \beta)(x_i))^{1-y_i}$$

$$= \arg \max_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^d} \sum_{i=1}^n y_i (\beta_0 + \langle \beta, x_i \rangle) - \log(1 + \exp(\beta_0 + \langle \beta, x_i \rangle)).$$

- 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 $(\beta_0, \beta)$).
Log-odds function and classifier

Log-odds function of $P(\beta_0, \beta)$ is
\[
x \mapsto \log \frac{\eta(\beta_0, \beta)(x)}{1 - \eta(\beta_0, \beta)(x)} = \beta_0 + \langle \beta, x \rangle,
\]
which is an affine function.

Bayes classifier for $P(\beta_0, \beta)$ is
\[
x \mapsto \begin{cases} 
0 & \text{if } \beta_0 + \langle \beta, x \rangle \leq 0, \\
1 & \text{if } \beta_0 + \langle \beta, x \rangle > 0.
\end{cases}
\]
Such classifiers are called linear classifiers.

Linear classifiers

A linear classifier is specified by a weight vector $w \in \mathbb{R}^d$ and threshold $t \in \mathbb{R}$:
\[
f_{w,t}(x) := \begin{cases} 
0 & \text{if } \langle w, x \rangle \leq t, \\
1 & \text{if } \langle w, x \rangle > t.
\end{cases}
\]

Interpretation: does a linear combination of input features exceed a threshold?
\[
\langle w, x \rangle = \sum_{i=1}^{d} w_i x_i > t.
\]

Translation from logistic regression parameters $(\beta_0, \beta)$:
\[
w = \beta, \quad t = -\beta_0.
\]

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 $w \in \mathbb{R}^d$.

The hyperplane with normal vector $w$ is the set of points orthogonal to $w$:
\[
H = \{ x \in \mathbb{R}^d : \langle w, x \rangle = 0 \}.
\]

The hyperplane and the point

Projection of $x$ onto span$\{w\}$ (a line) has coordinate
\[
||x||_2 \cdot \cos(\theta)
\]
where
\[
\cos(\theta) = \frac{\langle w, x \rangle}{||w||_2 \cdot ||x||_2}.
\]

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

Which side of the hyperplane (oriented by $w$)?

$||x||_2 \cdot \cos(\theta) > 0$ $\iff$ $\langle w, x \rangle > 0$ $\iff$ $x$ on same side of $H$ as $w$.
Affine hyperplanes

An affine hyperplane $H$ is a hyperplane that may be shifted away from the origin. Can be specified with a normal vector $w$ and a threshold $t \in \mathbb{R}$:

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

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

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 $f_{w,t}$ using iid sample $S$ such that

$$\text{err}(f_{w,t}) = \min_{w,t} \text{err}(f_{w,t})$$

is as small as possible.

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

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

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

Unfortunately, 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 $\leq 0.49$.

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 $w_\star \in \mathbb{R}^d$ and $t_\star \in \mathbb{R}$,

$$f_{w_\star, t_\star}(x) = y$$

for all $(x, y) \in S$.

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 $x \in \mathbb{R}^d$ to $\mathbb{R}^{d+1}$ with

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

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

Then linear classifier $f_{w,t}$ is the same as $f_{\tilde{w}} \circ \phi$,

where $f_{\tilde{w}} = f_{\tilde{w},0}$ 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 $w \in \mathbb{R}^d$ such that

$$f_w(x) = y$$

for all $(x, y) \in S$;

(and find such a vector if one exists).

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

  if $y = 0$: $\langle w, x \rangle \leq 0$,
  if $y = 1$: $\langle w, x \rangle > 0$.

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 $Y := \{-1, +1\}$ instead of $\{0, 1\}$.)*

**Perceptron**

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

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

**Perceptron**

**Predict** $a_t = -1$

**Correct label** $y_t = +1$

**Perceptron**

**Perceptron**

**Predict** $a_t = -1$

**Correct label** $y_t = +1$

$W_{t+1} := W_t + y_t x_t$

**Perceptron**

**Perceptron**

**Predict** $a_t = +1$

**Correct label** $y_t = -1$

**Perceptron**

**Perceptron**

**Predict** $a_t = +1$

**Correct label** $y_t = -1$

$W_{t+1} := W_t + y_t x_t$
When does Perceptron work?

Let $w_\ast \in \mathbb{R}^d$ be the shortest vector such that
\[
y(w_\ast, x) \geq 1 \text{ for all } (x, y) \in S
\]
(assuming one exists).

Perceptron terminates quickly when $\|w_\ast\|_2$ is small, i.e., when the minimum margin $1/\|w_\ast\|_2$ is large.

Online Perceptron

If data is not linearly separable, Perceptron runs forever!

Alternative: consider each example once, then halt.

Online Perceptron

\begin{algorithm}
\textbf{input} Labeled examples $\{(x_i, y_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.
\begin{enumerate}
\item initialize $\hat{w}_1 := 0$.
\item for $t = 1, 2, \ldots, n$ do
  \begin{enumerate}
  \item if $y_t \langle \hat{w}_t, x_t \rangle \leq 0$ then
    \begin{enumerate}
    \item $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
    \item else
    \end{enumerate}
  \item end if
  \end{enumerate}
\item end for
\item return $\hat{w}_{n+1}$.
\end{enumerate}
\end{algorithm}

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

Perceptron convergence

\textbf{Theorem}: If $R := \max_{(x, y) \in S} \|x\|_2$, and $w_\ast \in \mathbb{R}^d$ satisfies
\[
y(w_\ast, x) \geq 1 \text{ for all } (x, y) \in S,
\]
then Perceptron halts after at most $\|w_\ast\|_2^2 \cdot R^2$ iterations.

\textbf{Proof}: Suppose Perceptron does not exit the for-loop in iteration $t$.

Then there is a labeled example in $S$ — which we call $(x_t, y_t)$ — such that:
\begin{itemize}
\item $y_t(w_\ast, x_t) \geq 1$ (by definition of $w_\ast$)
\item $y_t(\hat{w}_t, x_t) \leq 0$ (since $f_{\hat{w}_t}$ misclassifies the example)
\item $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$ (since an update is made)
\end{itemize}

Use this information to (inductively) lower-bound
\[
\cos(\text{angle between } w_\ast \text{ and } \hat{w}_{t+1}) = \frac{\langle w_\ast, \hat{w}_{t+1} \rangle}{\|w_\ast\|_2 \|\hat{w}_{t+1}\|_2}.
\]

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

\[
\|\hat{w}_{t+1}\|_2^2 = \|\hat{w}_t + y_t x_t\|_2^2 = \|\hat{w}_t\|_2^2 + 2 y_t \langle \hat{w}_t, x_t \rangle + \|x_t\|_2^2 \leq \|\hat{w}_t\|_2^2 + R^2.
\]

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

\[
\cos(\text{angle between } w_\ast \text{ and } \hat{w}_{t+1}) = \frac{\langle w_\ast, \hat{w}_{t+1} \rangle}{\|w_\ast\|_2 \|\hat{w}_{t+1}\|_2} \geq \frac{t}{\|w_\ast\|_2 \cdot R \cdot \sqrt{t}}.
\]

Since cosine is at most one, we conclude
\[
t \leq \|w_\ast\|_2^2 \cdot R^2.
\]
Online learning

Online learning algorithms:
- Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.
- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.
- Upon seeing \(x_t\), learner makes a prediction: \(\hat{a}_t := \hat{f}_t(x_t)\).
- 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 \((x_1, y_1), \ldots, (x_t, y_t)\) (but usually just based on \((x_t, y_t)\)).

**Theorem:** If \(R := \max_{t \in \{1, \ldots, n\}} \|x_t\|_2\) and \(w_* \in \mathbb{R}^d\) satisfies
\[
y(w_*, x_t) \geq 1 \text{ for all } (x_t, y_t),
\]
then Online Perceptron makes at most \(\|w_*\|_2^2 \cdot R^2\) mistakes (and updates).

Online-to-Batch conversion

- Run online learning algorithm on sequence of examples
\((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
(in random order) to produce sequence of binary classifiers
\(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{n+1}\),
\(\hat{f}_i : \mathcal{X} \to \{\pm 1\}\).
- Final classifier: majority vote over the \(\hat{f}_i\)'s
\[
\hat{f}(x) := \begin{cases} -1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(x) \leq 0, \\ +1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(x) > 0. \end{cases}
\]

**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).

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.

Voted-Perceptron (Freund & Schapire, 1999)

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

Classifying OCR digits (digit ‘9’ → class +1; all other digits → class −1). 
\(n = 60000\) (about 6000 are of class +1), presented in a random order.

<table>
<thead>
<tr>
<th># passes</th>
<th>0.1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>err(OP, test)</td>
<td>0.079</td>
<td>0.064</td>
<td>0.057</td>
<td>0.063</td>
<td>0.058</td>
<td>0.059</td>
</tr>
<tr>
<td>err(VP, test)</td>
<td>0.045</td>
<td>0.039</td>
<td>0.038</td>
<td>0.038</td>
<td>0.038</td>
<td>0.037</td>
</tr>
</tbody>
</table>

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