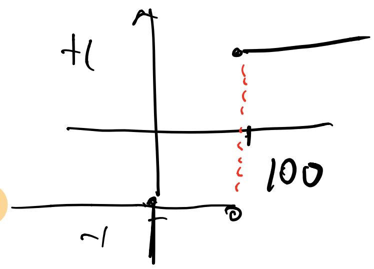
Lecture 14: Logistic Regression

Recall the credit question: given existing customer data, can a decision be made for a new customer's loan request?

(a) PLA algorithm: approximate a target function that outputs $+1, -1$ to make a binary decision on credit (approve/deny):

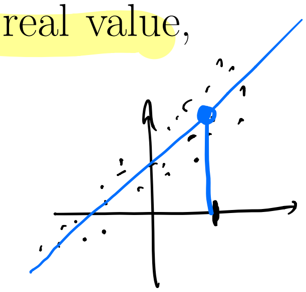
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$$



The sign function is a step function $\rightarrow$ *hard threshold.*

(b) Linear regression: find a target function that outputs a real value, to estimate a credit score:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}.$$



The output can be negative, f is unbounded $\rightarrow$ *no threshold.*

(c) Logistic regression: approximate the *probability* that a customer will default on their loan, given their customer data.

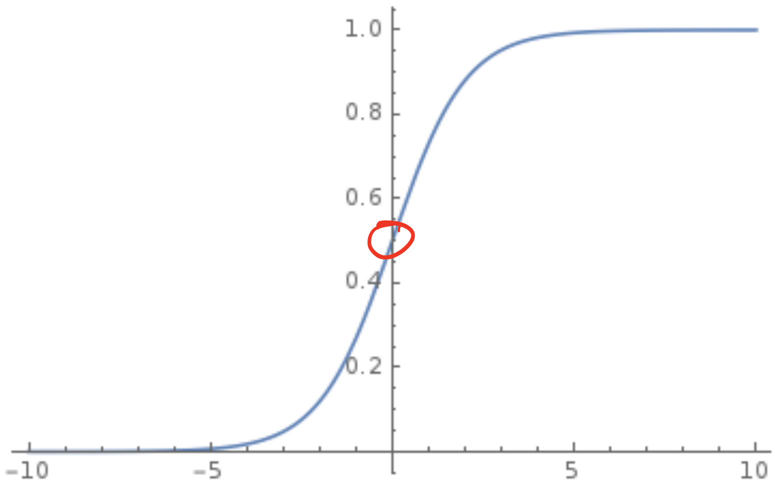
$$f(\mathbf{x}) = P(y = +1 \,|\, \mathbf{x}),$$

where $y = +1$ encodes the occurrence of a specific event, such as the customer defaults on loan, and $y = -1$ means such event has not occurred.

Target function is bounded, outputs real values $\rightarrow$ *soft threshold.*

Sigmoid function

Let $\theta : \mathbb{R} \to (0,1)$ be the logistic function $\boxed{\theta(s) = \dfrac{e^s}{1+e^s}}$.



Properties:

$$\lim_{s\to-\infty} \theta(s) = \lim_{s\to-\infty} \frac{\overset{0}{\cancel{e^s}}}{1+\underset{0}{\cancel{e^s}}} = 0$$

$$\lim_{s\to\infty} \theta(s) = \lim_{s\to\infty} \frac{e^s}{1+e^s} \overset{LH}{=} \lim_{s\to\infty} \frac{e^s}{e^s} = 1$$

$$\theta(-s) = \frac{e^{-s}}{1+e^{-s}} = \frac{1}{1+e^s} = 1 - \frac{e^s}{1+e^s} = 1 - \theta(s)$$

$$\theta(s) = \frac{1}{2} \Leftrightarrow \frac{e^s}{1+e^s} = \frac{1}{2} \Leftrightarrow 2e^s = 1+e^s \Leftrightarrow e^s = 1 \Leftrightarrow \underline{s=0}$$

Remark: Another popular sigmoid function is the hyperbolic tangent

$$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}},$$

but this function is not as easy to work with.

We want to approximate the target function:

$$f(\mathbf{x}) = P(y = +1 \,|\, \mathbf{x}),$$

The probability of outcome y given the input data $\mathbf{x}$ is

$$P(y \,|\, \mathbf{x}) = \begin{cases} f(\mathbf{x}) & , \text{ if } y = +1 \\ 1 - f(\mathbf{x}) & , \text{ if } y = -1 \end{cases}$$

Possible candidates from $\mathcal{H}$ to approximate $f(\mathbf{x})$, with range $(0, 1)$:

(i) $$\boxed{h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}}.}$$

$\vec{w}$ weights
$\vec{x}$ input

(ii) $$1 - h(\mathbf{x}) = 1 - \theta(\mathbf{w}^T \mathbf{x}) = \theta(-\mathbf{w}^T \mathbf{x}).$$ from (*)

Thus, combining with the probability above, we approximate

$$\boxed{P(y \,|\, \mathbf{x}) \approx \theta(y\, \mathbf{w}^T \mathbf{x})}$$

(i) if $y = +1$ $\quad P(y = +1 \,|\, \vec{x}) \approx \theta(\vec{\omega}^T \vec{x})$

(ii) if $y = -1$ $\quad P(y = -1 \,|\, \vec{x}) \approx \theta(-\vec{\omega}^T \vec{x})$

Question: How do we know which function h approximates the target function f best? Note that the function h depends on the coefficient vector $\mathbf{w}$, so we are really trying to find the best $\mathbf{w}$ to approximate f.

Error

- we will use the method of **maximum likelihood** to measure how close h is to f.
- this method maximizes how likely it is to get output y from the input $\mathbf{x}$ by using the function h.
- use the training data to maximize this probability.

Suppose the training data is $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where the data points are assumed to be independent of each other.

Maximize

$$P(y_1, y_2, \ldots, y_N \,|\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \prod_{k=1}^{N} P(y_k \,|\, \mathbf{x}_k).$$

Minimize

$$-\frac{1}{N} \log\, P(y_1, y_2, \ldots, y_N \,|\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = -\frac{1}{N} \log \left(\prod_{k=1}^{N} P(y_k \,|\, \mathbf{x}_k) \right)$$

Why? To maximize $f(x)$ is
To maximize $\log f(x)$ b/c log is increasing
To minimize $-\frac{\log f(x)}{N}$ [max becomes min
min becomes max]

Let the error function be given by

$$\boxed{E_{in}(\mathbf{w}) = -\frac{1}{N} \log \left(\prod_{k=1}^{N} P(y_k \,|\, \mathbf{x}_k) \right)}$$

$$
\begin{aligned}
E_{in}(\mathbf{w}) &= -\frac{1}{N} \log \left(\prod_{k=1}^{N} P(y_k \,|\, \mathbf{x}_k) \right) \\
&= -\frac{1}{N} \sum_{k=1}^{N} \log(P(y_k \,|\, \mathbf{x}_k)) \qquad \text{(log. prop.)} \\
&= -\frac{1}{N} \sum_{k=1}^{N} \log(\theta(y_k \, \mathbf{w}^T \mathbf{x}_k)) \qquad \text{(sigmoid)} \\
&= -\frac{1}{N} \sum_{k=1}^{N} \log \left(\frac{e^{y_k \, \mathbf{w}^T \mathbf{x}_k}}{1 + e^{y_k \, \mathbf{w}^T \mathbf{x}_k}} \right) \qquad \text{(sigmoid)} \\
&= \frac{1}{N} \sum_{k=1}^{N} \log \left(\frac{1 + e^{y_k \, \mathbf{w}^T \mathbf{x}_k}}{e^{y_k \, \mathbf{w}^T \mathbf{x}_k}} \right) \qquad \left(-\log \tfrac{1}{a} = \log a\right) \\
&= \frac{1}{N} \sum_{k=1}^{N} \log \left(1 + e^{-y_k \, \mathbf{w}^T \mathbf{x}_k} \right)
\end{aligned}
$$

Remark: if $y_k \, \mathbf{w}^T \mathbf{x}_k$ is large and positive, the error is small, hence y_k is probably correctly labeled.

Find the minimum of this function: set the gradient to zero.

$$
\begin{aligned}
\nabla_{\mathbf{w}} E_{in}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} \log\left(1 + e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}\right) \right] \\
\text{(pull out } \tfrac{1}{N}\text{)} \qquad &= \frac{1}{N} \nabla_{\mathbf{w}} \left[\sum_{k=1}^{N} \log\left(1 + e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}\right) \right] \\
\text{(}\nabla \text{ inside sum)} \qquad &= \frac{1}{N} \sum_{k=1}^{N} \nabla_{\mathbf{w}} \log\left(1 + e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}\right) \\
\begin{array}{c}\text{(differentiate)} \\ \text{rember Chain Rule}\end{array} \qquad &= \frac{1}{N} \sum_{k=1}^{N} \frac{1}{1 + e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}} \underbrace{\left(e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}\right) \left(-y_k \mathbf{x}_k\right)}_{\text{numerator}} \\
&= \frac{1}{N} \sum_{k=1}^{N} \frac{-y_k \mathbf{x}_k\, e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}}{1 + e^{-y_k\, \mathbf{w}^T \mathbf{x}_k}} \\
\begin{array}{c}\text{mult top/bottom} \\ \text{by } e^{y_k \vec{w}\cdot\vec{x}}\end{array} \qquad &= \frac{1}{N} \sum_{k=1}^{N} \frac{-y_k \mathbf{x}_k}{1 + e^{y_k\, \mathbf{w}^T \mathbf{x}_k}} \\
\text{(pull out sign)} \qquad &= \boxed{-\frac{1}{N} \sum_{k=1}^{N} \frac{y_k \mathbf{x}_k}{1 + e^{y_k\, \mathbf{w}^T \mathbf{x}_k}}} \\
&= -\frac{1}{N} \sum_{k=1}^{N} y_k \mathbf{x}_k\, \theta(-y_k\, \mathbf{w}^T \mathbf{x}_k).
\end{aligned}
$$

Remark: Computing the gradient is easy, but solving $\nabla_{\mathbf{w}} E_{in}(\mathbf{w}) = \mathbf{0}$ is not trivial $\rightarrow$ use the gradient descent algorithm or the stochastic gradient descent algorithm to find a $\mathbf{w}$ that minimizes $E_{in}(\mathbf{w})$ instead.

Logistic Regression Algorithm (with Gradient Descent)

1. Set the initial weights $\mathbf{w}_0$ and step size η

2. For $t \geq 0$,

   - find the gradient $\mathbf{g}_t = -\frac{1}{N}\sum_{k=1}^{N} \frac{y_k \mathbf{x}_k}{1+e^{y_k \mathbf{w}_t^T \mathbf{x}_k}}$
   - update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\, \mathbf{g}_t$.

3. Stop when "done"

4. Return final $\mathbf{w}_t$.

Remarks:

(a) To initialize $\mathbf{w}_0$, one can set it to $\mathbf{0}$. Another option is to initialize each coordinate in $\mathbf{w}(0)$ by independently sampling from a normal distribution with mean zero and small variance.

(b) To end the algorithm, one can run it for a fixed (thousands) number of steps, or run it until $\|\mathbf{g}_t\|$ drops below a certain small threshold (since minimum error is achieved at $\mathbf{g}_t = \mathbf{0}$), or a combination of both.

(c) Instead of using a constant step η, one can use variable η_t, typically with η_t decreasing. [when $\vec{g}_t$ changes direction is a good time to decrease η_t].

Instead of using the error from all N data points, one can use error from one data point uniformly picked at random from the training set. Let

$$e_k(\mathbf{w}) = \log\left(1 + e^{-y_k\,\mathbf{w}^T\mathbf{x}_k}\right)$$

be the error from data point $(\mathbf{x}_k, y_k)$. Then the update step in the gradient descent algorithm will be based only on the error from this point, as described below:

Logistic Regression Algorithm (with Stochastic Gradient Descent)

1. Set the initial weights $\mathbf{w}_0$ and step size η
2. For $t \geq 0$,
   - pick one data point from $\mathcal{D}$ uniformly at random. Suppose it is $(\mathbf{x}_k, y_k)$.
   - find the gradient $\mathbf{g}_t = \nabla e_k(\mathbf{w}_t) = -\dfrac{y_k\mathbf{x}_k}{1 + e^{y_k\,\mathbf{w}_t^T\mathbf{x}_k}}$
   - update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\,\mathbf{g}_t$.
3. Stop when "done"
4. Return final $\mathbf{w}_t$

Remarks:

(a) $E_{in}(\mathbf{w}) = \dfrac{1}{N}\displaystyle\sum_{k=1}^{N} e_k(\mathbf{w})$

(b) $\nabla E_{in}(\mathbf{w}) = \dfrac{1}{N}\displaystyle\sum_{k=1}^{N} \nabla e_k(\mathbf{w})$

(c) the computational cost of using the stochastic version is cheaper by a factor of N

(d) the stochastic version is more wiggly, but in the long run it averages out.

(e) Stochastic Gradient Descent Algorithm is as efficient as the Gradient Descent Algorithm: on average, the change at each iteration is

$$\begin{aligned}
\underbrace{\mathbb{E}\left[-\eta\nabla e(\mathbf{w})\right]}_{\text{change for stochastic gradient descent}} &= -\eta\sum_{k=1}^{N} \overbrace{P(\text{pick data point } k)}^{1/N}\nabla e_k(\mathbf{w}) \\
&= -\frac{\eta}{N}\sum_{k=1}^{N}\nabla e_k(\mathbf{w}) \\
&= \underbrace{-\eta\nabla E_{in}(\mathbf{w})}_{\text{change for gradient descent}}.
\end{aligned}$$

Midterm exam

PLA: $[w_0, w_1, \dots, w_d]$

augment $\vec{x}$ by $x_0 = 1$

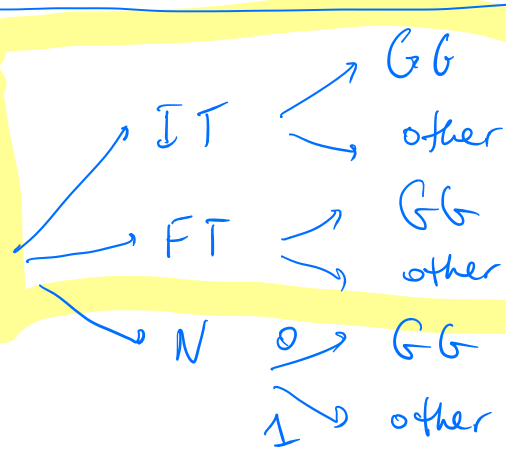
$$h(\vec{x}) = \text{sign}(w_0 + w_1 x_1 + \dots + w_d x_d)$$

Missing w_0 term $\Rightarrow$ hyperplane goes thru $\vec{0}$

threshold term

$$w_1(\text{salary}) + w_2(\text{age}) + w_3(\text{debt}) > \underbrace{b}_{= w_0}$$

#4 ?



Given : Twin girls

$P(FT \mid GG)$