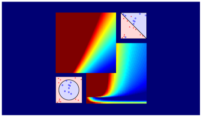
Machine Learning Foundations
(機器學習基石)



Lecture 10: Logistic Regression

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. When Can Machines Learn?
2. Why Can Machines Learn?
3. **How** Can Machines Learn?

Lecture 9: Linear Regression

analytic solution $\mathbf{w}_{\text{LIN}} = \mathrm{X}^{\dagger}\mathbf{y}$ with
linear regression hypotheses and **squared error**

Lecture 10: Logistic Regression

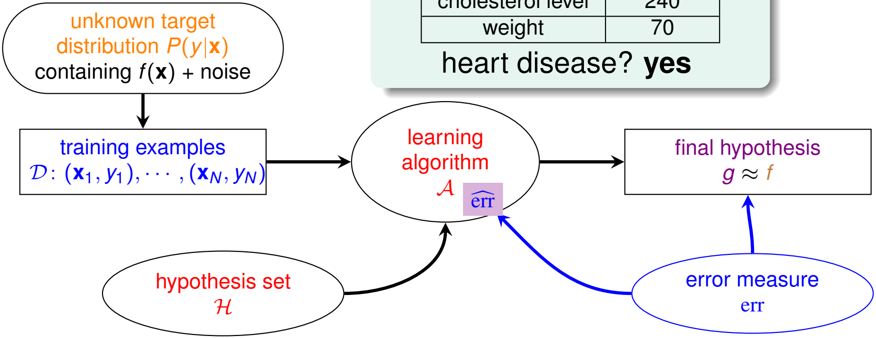
- Logistic Regression Problem
- Logistic Regression Error
- Gradient of Logistic Regression Error
- Gradient Descent

4. How Can Machines Learn Better?

Heart Attack Prediction Problem (1/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart disease? **yes**



binary classification:

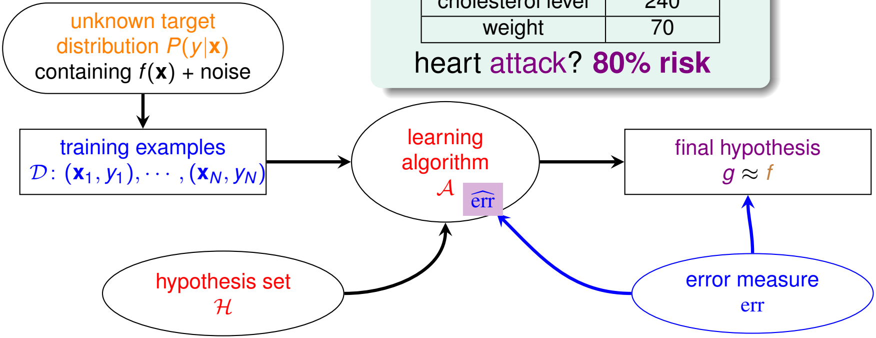
$$\text{ideal } f(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \tfrac{1}{2}\right) \in \{-1, +1\}$$

because of classification err

Heart Attack Prediction Problem (2/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart attack? **80% risk**



'soft' binary classification:

$$f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$$

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$
\begin{aligned}
&\left(\mathbf{x}_1, y'_1 = 0.9 = P(+1|\mathbf{x}_1)\right)\\
&\left(\mathbf{x}_2, y'_2 = 0.2 = P(+1|\mathbf{x}_2)\right)\\
&\quad\vdots\\
&\left(\mathbf{x}_N, y'_N = 0.6 = P(+1|\mathbf{x}_N)\right)
\end{aligned}
$$

actual (noisy) data

$$
\begin{aligned}
&\left(\mathbf{x}_1, y_1 = \circ \sim P(y|\mathbf{x}_1)\right)\\
&\left(\mathbf{x}_2, y_2 = \times \sim P(y|\mathbf{x}_2)\right)\\
&\quad\vdots\\
&\left(\mathbf{x}_N, y_N = \times \sim P(y|\mathbf{x}_N)\right)
\end{aligned}
$$

same data as hard binary classification,
different **target function**

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\begin{array}{l}
(\mathbf{x}_1, y'_1 = 0.9 = P(+1|\mathbf{x}_1)) \\
(\mathbf{x}_2, y'_2 = 0.2 = P(+1|\mathbf{x}_2)) \\
\quad\vdots \\
(\mathbf{x}_N, y'_N = 0.6 = P(+1|\mathbf{x}_N))
\end{array}$$

actual (noisy) data

$$\begin{array}{l}
(\mathbf{x}_1, y'_1 = 1 = [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_1)]\!]) \\
(\mathbf{x}_2, y'_2 = 0 = [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_2)]\!]) \\
\quad\vdots \\
(\mathbf{x}_N, y'_N = 0 = [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_N)]\!])
\end{array}$$

same data as hard binary classification,
different **target function**

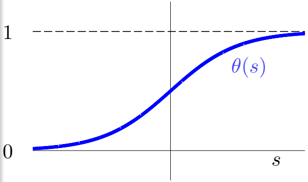
Logistic Hypothesis

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of patient', calculate a weighted 'risk score':

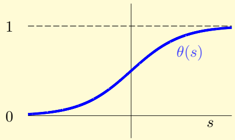
$$s = \sum_{i=0}^{d} w_i x_i$$

- convert the score to estimated probability by logistic function $\theta(s)$



logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

Logistic Function



$\theta(-\infty) = 0$; $\theta(0) = \frac{1}{2}$; $\theta(\infty) = 1$

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

—smooth, monotonic, sigmoid function of s

logistic regression: use

$$h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

to approximate target function $f(\mathbf{x}) = P(+1|\mathbf{x})$

Fun Time

Logistic Regression and Binary Classification

Consider any logistic hypothesis $h(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$ that approximates $P(y|\mathbf{x})$. 'Convert' $h(\mathbf{x})$ to a binary classification prediction by taking $\text{sign}\left(h(\mathbf{x}) - \frac{1}{2}\right)$. What is the equivalent formula for the binary classification prediction?

1. $\text{sign}\left(\mathbf{w}^T\mathbf{x} - \frac{1}{2}\right)$
2. $\text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$
3. $\text{sign}\left(\mathbf{w}^T\mathbf{x} + \frac{1}{2}\right)$
4. none of the above

Reference Answer: 2

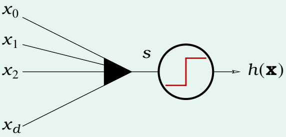
When $\mathbf{w}^T\mathbf{x} = 0$, $h(\mathbf{x})$ is exactly $\frac{1}{2}$. So thresholding $h(\mathbf{x})$ at $\frac{1}{2}$ is the same as thresholding $(\mathbf{w}^T\mathbf{x})$ at 0.

Three Linear Models

linear scoring function: $s = \mathbf{w}^T\mathbf{x}$

linear classification

$h(\mathbf{x}) = \text{sign}(s)$

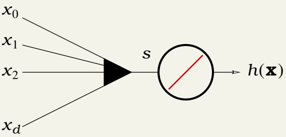


plausible err = 0/1

(small flipping noise)

linear regression

$h(\mathbf{x}) = s$

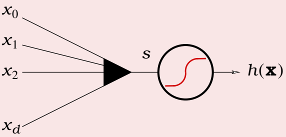


friendly err = squared

(easy to minimize)

logistic regression

$h(\mathbf{x}) = \theta(s)$



err = ?

how to define

$E_{\text{in}}(\mathbf{w})$ for logistic regression?

Likelihood

target function
$f(\mathbf{x}) = P(+1|\mathbf{x})$ $\Leftrightarrow$ $P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$

probability that f generates $\mathcal{D}$

$$P(\mathbf{x}_1)P(\circ|\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)P(\times|\mathbf{x}_2) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)P(\times|\mathbf{x}_N)$$

likelihood that h generates $\mathcal{D}$

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

- if $h \approx f$,
  then likelihood(h) $\approx$ probability using f
- probability using f usually **large**

Likelihood

target function $f(\mathbf{x}) = P(+1|\mathbf{x})$ $\Leftrightarrow$ $P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$

probability that f generates $\mathcal{D}$	likelihood that h generates $\mathcal{D}$
$P(\mathbf{x}_1)f(\mathbf{x}_1) \times$	$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$
$P(\mathbf{x}_2)(1 - f(\mathbf{x}_2)) \times$	$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$
$\ldots$	$\ldots$
$P(\mathbf{x}_N)(1 - f(\mathbf{x}_N))$	$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$

- if $h \approx f$,
  then likelihood(h) $\approx$ probability using f
- probability using f usually **large**

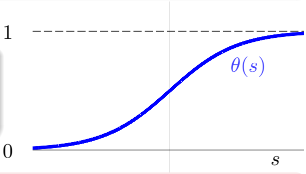
Likelihood of Logistic Hypothesis

likelihood(h) $\approx$ (probability using f) $\approx$ **large**

$$g = \operatorname*{argmax}_{h} \ \text{likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times \ldots P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

$$\text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n \mathbf{x}_n)$$

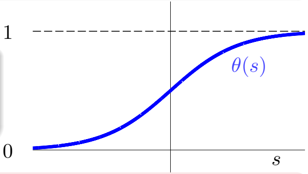
Likelihood of Logistic Hypothesis

likelihood(h) $\approx$ (probability using f) $\approx$ **large**

$$g = \underset{h}{\text{argmax}} \ \text{likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(+\mathbf{x}_1) \times P(\mathbf{x}_2)h(-\mathbf{x}_2) \times \ldots P(\mathbf{x}_N)h(-\mathbf{x}_N)$$

$$\text{likelihood(logistic } h) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{h} \quad \text{likelihood(logistic } h) \propto \prod_{n=1}^{N} h(y_n \mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \text{likelihood}(\mathbf{w}) \propto \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \ln \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

$$\min_{\mathbf{w}} \quad \frac{1}{N}\sum_{n=1}^{N} - \ln \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

$$\theta(s) = \frac{1}{1+\exp(-s)} \quad : \quad \min_{\mathbf{w}} \quad \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

$$\Longrightarrow \quad \min_{\mathbf{w}} \quad \underbrace{\frac{1}{N}\sum_{n=1}^{N} \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

$\mathrm{err}(\mathbf{w}, \mathbf{x}, y) = \ln\left(1+\exp(-y\mathbf{w}\mathbf{x})\right)$:
cross-entropy error

Fun Time

The four statements below help us understand more about the cross-entropy error $\mathrm{err}(\mathbf{w}, \mathbf{x}, y) = \ln\left(1 + \exp(-y\mathbf{w}^T\mathbf{x})\right)$. Consider $\mathbf{w}^T\mathbf{x} \neq 0$. Which statement is not true?

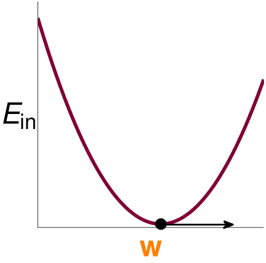
1. For any $\mathbf{w}, \mathbf{x}$, and y, $\mathrm{err}(\mathbf{w}, \mathbf{x}, y) > 0$.
2. For any $\mathbf{w}, \mathbf{x}$, and y, $\mathrm{err}(\mathbf{w}, \mathbf{x}, y) < 1126$.
3. When $y = \mathrm{sign}\left(\mathbf{w}^T\mathbf{x}\right)$, $\mathrm{err}(\mathbf{w}, \mathbf{x}, y) < \ln 2$.
4. When $y \neq \mathrm{sign}\left(\mathbf{w}^T\mathbf{x}\right)$, $\mathrm{err}(\mathbf{w}, \mathbf{x}, y) \geq \ln 2$.

Reference Answer: 2

1126, really? :-) You are highly encouraged to plot the curve of err with respect to some fixed y and some varying score $s = \mathbf{w}^T\mathbf{x}$ to know more about the error measure. After plotting, it is easy to see that err is not bounded above, and the other three choices are correct.

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, twice-differentiable, **convex**
- how to minimize? locate **valley**

$$\text{want } \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$$

first: derive $\nabla E_{\text{in}}(\mathbf{w})$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln \left(\underbrace{1 + \exp(\overbrace{-y_n \mathbf{w}^T \mathbf{x}_n}^{\bigcirc})}_{\square} \right)$$

$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N}\sum_{n=1}^{N} \left(\frac{\partial \ln(\square)}{\partial \square}\right) \left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right) \left(\frac{\partial - y_n \mathbf{w}^T \mathbf{x}_n}{\partial w_i}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\qquad \right) \left(\qquad \right) \left(\qquad \right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right) \left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N} \theta(\bigcirc)\left(-y_n x_{n,i}\right)
\end{aligned}$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln \Bigg(\underbrace{1 + \exp(\overbrace{-y_n \mathbf{w}^T \mathbf{x}_n}^{\bigcirc})}_{\square} \Bigg)$$

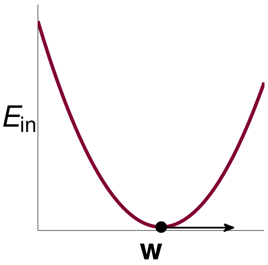
$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N}\sum_{n=1}^{N} \left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial - y_n \mathbf{w}^T \mathbf{x}_n}{\partial w_i}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\frac{1}{\square}\right)\left(\exp(\bigcirc)\right)\left(-y_n x_{n,i}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N} \theta(\bigcirc)\left(-y_n x_{n,i}\right)
\end{aligned}$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\Big(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\Big)$$

$$\text{want } \nabla E_{\text{in}}(\mathbf{w}) = \boxed{\frac{1}{N}\sum_{n=1}^{N} \theta\Big(-y_n \mathbf{w}^T \mathbf{x}_n\Big)\Big(-y_n \mathbf{x}_n\Big)} = \mathbf{0}$$



scaled θ-weighted sum of $-y_n\mathbf{x}_n$

- all $\theta(\cdot) = 0$: only if $y_n \mathbf{w}^T \mathbf{x}_n \gg 0$
  —linear separable $\mathcal{D}$
- weighted sum = $\mathbf{0}$:
  non-linear equation of $\mathbf{w}$

closed-form solution? no :-(

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\operatorname{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \left[\!\left[\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n \right]\!\right] y_n \mathbf{x}_n$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{1}_{\eta} \cdot \underbrace{\left(\left[\!\left[\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n \right]\!\right] \cdot y_n \mathbf{x}_n \right)}_{\mathbf{v}}$$

when stop, return last $\mathbf{w}$ as g

choice of $(\eta, \mathbf{v})$ and stopping condition defines **iterative optimization approach**

Fun Time

Consider the gradient $\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$. That is, each example $(\mathbf{x}_n, y_n)$ contributes to the gradient by an amount of $\theta\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)$. For any given $\mathbf{w}$, which example contributes the most amount to the gradient?

1. the example with the smallest $y_n \mathbf{w}^T \mathbf{x}_n$ value
2. the example with the largest $y_n \mathbf{w}^T \mathbf{x}_n$ value
3. the example with the smallest $\mathbf{w}^T \mathbf{x}_n$ value
4. the example with the largest $\mathbf{w}^T \mathbf{x}_n$ value

Reference Answer: 1

Using the fact that θ is a monotonic function, we see that the example with the smallest $y_n \mathbf{w}^T \mathbf{x}_n$ value contributes to the gradient the most.

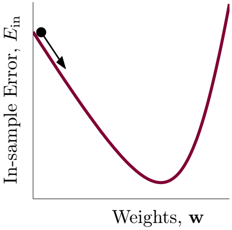
Iterative Optimization

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

- PLA: $\mathbf{v}$ comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll '**downhill**'?
  - direction $\mathbf{v}$: (assumed) of unit length
  - step size η: (assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

Linear Approximation

a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} \quad E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v})$$

- still non-linear optimization, now **with constraints**
  —not any easier than $\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w})$
- local approximation by linear formula makes problem easier

$$E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx E_{\text{in}}(\mathbf{w}_t) + \eta\mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)$$

  if η really small (Taylor expansion)

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \quad \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

Gradient Descent

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

- optimal $\mathbf{v}$: opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

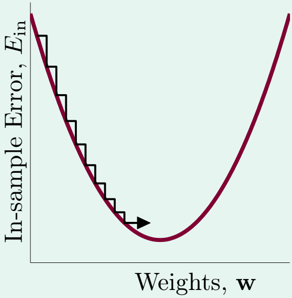
$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

- gradient descent: for **small** η, $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$

gradient descent:
a simple & popular optimization tool

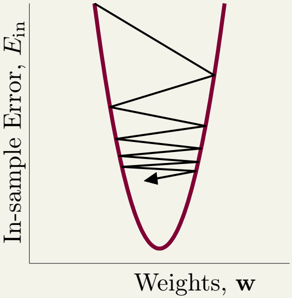
Choice of η

too small



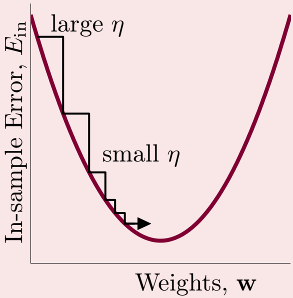
too slow :-(

too large



too unstable :-(

just right



use changing η

η better be **monotonic of** $\|\nabla E_{\text{in}}(\mathbf{w}_t)\|$

Simple Heuristic for **Changing** η

η better be **monotonic of** $\|\nabla E_{\text{in}}(\mathbf{w}_t)\|$

- if red $\eta \propto \|\nabla E_{\text{in}}(\mathbf{w}_t)\|$ by ratio purple η

$$\begin{aligned} \mathbf{w}_{t+1} \quad \leftarrow \quad & \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|} \\ & \qquad \| \\ & \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t) \end{aligned}$$

- call purple η the fixed learning rate

fixed learning rate gradient descent:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

Putting Everything Together

Logistic Regression Algorithm

initialize $\mathbf{w}_0$

For $t = 0, 1, \cdots$

1. compute

$$\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N} \sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

2. update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) = 0$ or enough iterations

return last $\mathbf{w}_{t+1}$ as g

similar time complexity to **pocket** per iteration

Fun Time

If $\mathbf{w}_0 = \mathbf{0}$, and take $\eta = 0.1$. What is $\mathbf{w}_1$ in the logistic regression algorithm?

1. $+0.1 \cdot \frac{1}{N} \sum_{n=1}^{N} y_n \mathbf{x}_n$
2. $-0.1 \cdot \frac{1}{N} \sum_{n=1}^{N} y_n \mathbf{x}_n$
3. $+0.05 \cdot \frac{1}{N} \sum_{n=1}^{N} y_n \mathbf{x}_n$
4. $-0.05 \cdot \frac{1}{N} \sum_{n=1}^{N} y_n \mathbf{x}_n$

Reference Answer: 3

You can do a simple substitution using the fact that $\theta(0) = \frac{1}{2}$. This result shows that a scaled average of $y_n \mathbf{x}_n$ is somewhat 'one-step' better than the zero vector.

Summary

1. When Can Machines Learn?
2. Why Can Machines Learn?
3. **How** Can Machines Learn?

Lecture 9: Linear Regression

Lecture 10: Logistic Regression

- Logistic Regression Problem
  $P(+1|\mathbf{x})$ as target and $\theta(\mathbf{w}^T\mathbf{x})$ as hypotheses
- Logistic Regression Error
  cross-entropy (negative log likelihood)
- Gradient of Logistic Regression Error
  θ-weighted sum of data vectors
- Gradient Descent
  roll downhill by $-\nabla E_{\text{in}}(\mathbf{w})$

- **next: linear model'S' for classification**

4. How Can Machines Learn Better?