Learning From Data
Lecture 9
Logistic Regression and Gradient Descent

Logistic Regression
Gradient Descent

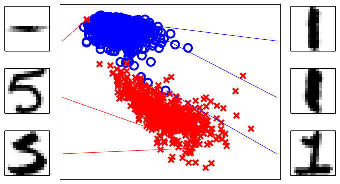
M. Magdon-Ismail
CSCI 4100/6100

RECAP: Linear Classification and Regression

The linear signal:

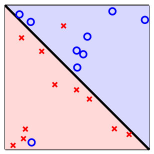
$$s = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

Good Features are Important

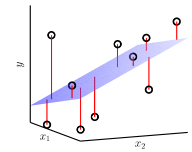


Before looking at the data, we can **reason** that symmetry and intensity should be good features **based on our knowledge of the problem**.

Algorithms



Linear Classification.
Pocket algorithm can tolerate errors
Simple and efficient



Linear Regression.
Single step learning:

$$\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathbf{y}$$

Very efficient $O(Nd^2)$ *exact* algorithm.

Predicting a Probability

Will someone have a heart attack over the next year?

age	62 years
gender	male
blood sugar	120 mg/dL40,000
HDL	50
LDL	120
Mass	190 lbs
Height	5′ 10″
…	…

Classification: Yes/No

Logistic Regression: Likelihood of heart attack logistic regression $\equiv y \in [0,1]$

$$h(\mathbf{x}) = \theta\left(\sum_{i=0}^{d} w_i x_i\right) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Predicting a Probability

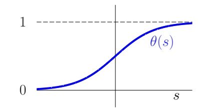
Will someone have a heart attack over the next year?

age	62 years
gender	male
blood sugar	120 mg/dL40,000
HDL	50
LDL	120
Mass	190 lbs
Height	5′ 10″
…	…

Classification: Yes/No

Logistic Regression: Likelihood of heart attack logistic regression $\equiv y \in [0,1]$

$$h(\mathbf{x}) = \theta\left(\sum_{i=0}^{d} w_i x_i\right) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$



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

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

The Data is Still Binary, ± 1

$$\mathcal{D} = (\mathbf{x}_1, y_1 = \pm 1), \cdots, (\mathbf{x}_N, y_N = \pm 1)$$

$\mathbf{x}_n$ $\leftarrow$ a person's health information

$y_n = \pm 1$ $\leftarrow$ **did** they have a heart attack or not

We cannot measure a *probability*.

We can only see the occurence of an event and try to *infer* a probability.

The Target Function is Inherently Noisy

$$f(\mathbf{x}) = \mathbb{P}[y = +1 \mid \mathbf{x}].$$

The data is generated from a *noisy* target function:

$$P(y \mid \mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

What Makes an h Good?

'fitting' the data means finding a good h

h is good if: $\begin{cases} h(\mathbf{x}_n) \approx 1 & \text{whenever } y_n = +1; \\ h(\mathbf{x}_n) \approx 0 & \text{whenever } y_n = -1. \end{cases}$

A simple error measure that captures this:

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}\left(h(\mathbf{x}_n) - \tfrac{1}{2}(1+y_n)\right)^2.$$

Not very convenient (hard to minimize).

The Cross Entropy Error Measure

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln(1 + e^{-y_n \cdot \mathbf{w}^{\intercal}\mathbf{x}})$$

It looks complicated and ugly $(\ln, e^{(\cdot)}, \ldots)$,

But,

- it is based on an intuitive probabilistic interpretation of h.
- it is very convenient and mathematically friendly ('easy' to minimize).

Verify: $y_n = +1$ encourages $\mathbf{w}^{\intercal}\mathbf{x}_n \gg 0$, so $\theta(\mathbf{w}^{\intercal}\mathbf{x}_n) \approx 1$; $y_n = -1$ encourages $\mathbf{w}^{\intercal}\mathbf{x}_n \ll 0$, so $\theta(\mathbf{w}^{\intercal}\mathbf{x}_n) \approx 0$;

The Probabilistic Interpretation

Suppose that $h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ closely captures $\mathbb{P}[+1|\mathbf{x}]$:

$$P(y \mid \mathbf{x}) = \begin{cases} \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = +1; \\ 1 - \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = -1. \end{cases}$$

The Probabilistic Interpretation

So, if $h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ closely captures $\mathbb{P}[+1|\mathbf{x}]$:

$$P(y \mid \mathbf{x}) = \begin{cases} \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = +1; \\ \theta(-\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = -1. \end{cases}$$

The Probabilistic Interpretation

So, if $h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ closely captures $\mathbb{P}[+1|\mathbf{x}]$:

$$P(y \mid \mathbf{x}) = \begin{cases} \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = +1; \\ \theta(-\mathbf{w}^{\mathrm{T}}\mathbf{x}) & \text{for } y = -1. \end{cases}$$

. . . or, more compactly,

$$P(y \mid \mathbf{x}) = \theta(y \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x})$$

The Likelihood

$$P(y \mid \mathbf{x}) = \theta(y \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Recall: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ are independently generated

Likelihood:
The probability of getting the $y_1, \ldots, y_N$ in $\mathcal{D}$ from the corresponding $\mathbf{x}_1, \ldots, \mathbf{x}_N$:

$$P(y_1, \ldots, y_N \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n).$$

The likelihood measures the probability that the data were generated if f were h.

Maximizing The Likelihood (why?)

$$
\begin{aligned}
& \max && \textstyle\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \max && \ln\left(\textstyle\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) \\
\equiv\ & \max && \textstyle\sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\Leftrightarrow\ & \min && -\tfrac{1}{N}\textstyle\sum_{n=1}^{N} \ln P(y_n \mid \mathbf{x}_n) \\
\equiv\ & \min && \tfrac{1}{N}\textstyle\sum_{n=1}^{N} \ln \tfrac{1}{P(y_n \mid \mathbf{x}_n)} \\
\equiv\ & \min && \tfrac{1}{N}\textstyle\sum_{n=1}^{N} \ln \tfrac{1}{\theta(y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n)} \qquad \leftarrow \text{we specialize to our "model" here} \\
\equiv\ & \min && \tfrac{1}{N}\textstyle\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n})
\end{aligned}
$$

$$
E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_n})
$$

How To Minimize $E_{\text{in}}(\mathbf{w})$

Classification – PLA/Pocket (iterative)

Regression – pseudoinverse (analytic), from solving $\nabla_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \mathbf{0}$.

Logistic Regression – analytic won't work.

Numerically/iteratively set $\nabla_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) \to \mathbf{0}$.

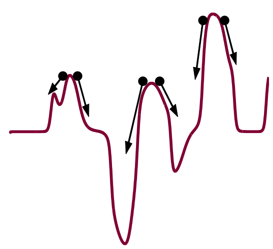
Finding The Best Weights - Hill Descent

Ball on a complicated hilly terrain

— rolls down to a *local valley*

↑

this is called a *local minimum*

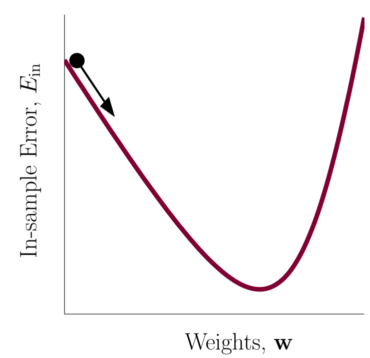


Questions:

How to get to the bottom of the deepest valey?

How to do this when we don't have gravity?

Our E_{in} Has Only One Valley



... because $E_{\text{in}}(\mathbf{w})$ is a **convex function** of $\mathbf{w}$. (So, who care's if it looks ugly!)

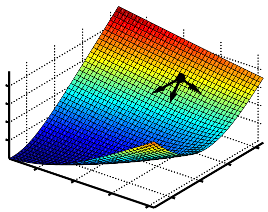
How to “Roll Down”?

Assume you are at weights $\mathbf{w}(t)$ and you take a step of size η in the direction $\hat{\mathbf{v}}$.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\hat{\mathbf{v}}$$

We get to pick $\hat{\mathbf{v}}$ ← what’s the best direction to take the step?

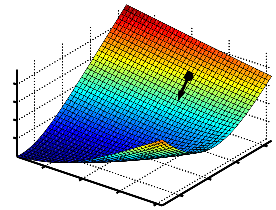
Pick $\hat{\mathbf{v}}$ to make $E_{\text{in}}(\mathbf{w}(t+1))$ as small as possible.



The Gradient is the Fastest Way to Roll Down

Approximating the change in E_{in}

$$\begin{aligned}
\Delta E_{\text{in}} &= E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t)) \\
&= E_{\text{in}}(\mathbf{w}(t) + \eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(t)) \\
&= \underbrace{\eta\,\nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}}}_{\text{minimized at } \hat{\mathbf{v}} = -\frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|}} + O(\eta^2) \qquad \text{(Taylor's Approximation)} \\
&\gtrapprox -\eta\|\nabla E_{\text{in}}(\mathbf{w}(t))\| \qquad \leftarrow \text{attained at } \hat{\mathbf{v}} = -\frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|}
\end{aligned}$$

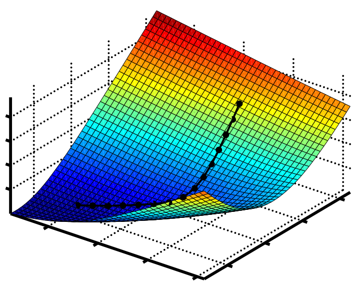


The best (steepest) direction to move is the negative gradient:

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

“Rolling Down” ≡ Iterating the Negative Gradient

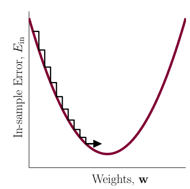
$\mathbf{w}(0)$
$\downarrow$ ← negative gradient
$\mathbf{w}(1)$
$\downarrow$ ← negative gradient
$\mathbf{w}(2)$
$\downarrow$ ← negative gradient
$\mathbf{w}(3)$
$\downarrow$ ← negative gradient
$\vdots$

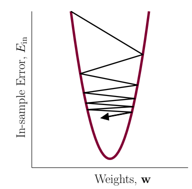


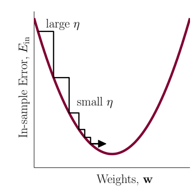
$\eta = 0.5$; 15 steps

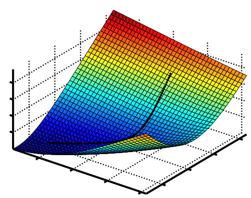
The ‘Goldilocks’ Step Size

η too small | **η too large** | **variable η_t – just right**

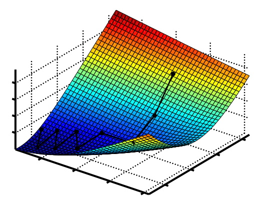




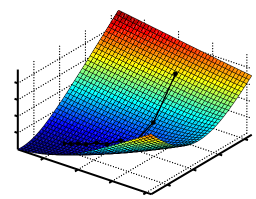




$\eta = 0.1$; 75 steps



$\eta = 2$; 10 steps



variable η_t; 10 steps

Fixed Learning Rate Gradient Descent

$$\eta_t = \eta \cdot \|\nabla E_{\text{in}}(\mathbf{w}(t))\|$$

$\|\nabla E_{\text{in}}(\mathbf{w}(t))\| \to 0$ when closer to the minimum.

$$\begin{aligned} \eta_t \hat{\mathbf{v}} &= -\eta_t \cdot \frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|} \\ &= -\eta \cdot \cancel{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|} \cdot \frac{\nabla E_{\text{in}}(\mathbf{w}(t))}{\cancel{\|\nabla E_{\text{in}}(\mathbf{w}(t))\|}} \end{aligned}$$

$$\eta_t \hat{\mathbf{v}} = -\eta \cdot \nabla E_{\text{in}}(\mathbf{w}(t))$$

Initialize at step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
  Compute the gradient
  $$\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t)). \longleftarrow \text{(Ex. 3.7 in LFD)}$$
  Move in the direction $\mathbf{v}_t = -\mathbf{g}_t$.
  Update the weights:
  $$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t.$$
  Iterate 'until it is time to stop'.
end for
Return the final weights.

Gradient descent can minimize any smooth function, for example

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathsf{T}}\mathbf{x}}) \qquad \leftarrow \text{logistic regression}$$

Stochastic Gradient Descent (SGD)

A variation of GD that considers only the error on one data point.

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \cdot \mathbf{w}^{\mathsf{T}}\mathbf{x}}) = \frac{1}{N}\sum_{n=1}^{N} e(\mathbf{w}, \mathbf{x}_n, y_n)$$

- Pick a random data point $(\mathbf{x}_*, y_*)$
- Run an iteration of GD on $e(\mathbf{w}, \mathbf{x}_*, y_*)$

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} e(\mathbf{w}, \mathbf{x}_*, y_*)$$

1. The 'average' move is the same as GD;
2. Computation: fraction $\frac{1}{N}$ cheaper per step;
3. Stochastic: helps escape local minima;
4. Simple;
5. Similar to PLA.

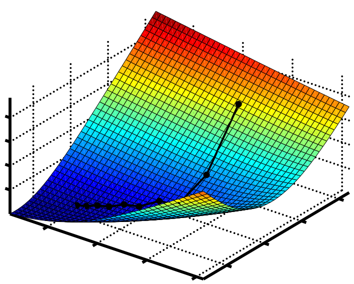
Logistic Regression:

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y_* \mathbf{x}_* \left(\frac{\eta}{1 + e^{y_* \mathbf{w}^{\mathsf{T}} \mathbf{x}_*}} \right)$$

(Recall PLA: $\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y_* \mathbf{x}_*$)

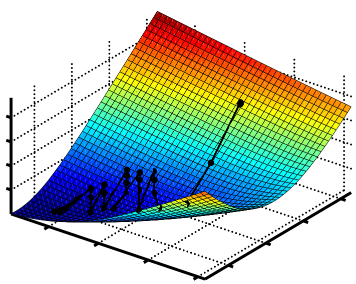
Stochastic Gradient Descent

GD



$\eta = 6$
10 steps
$N = 10$

SGD



$\eta = 2$
30 steps