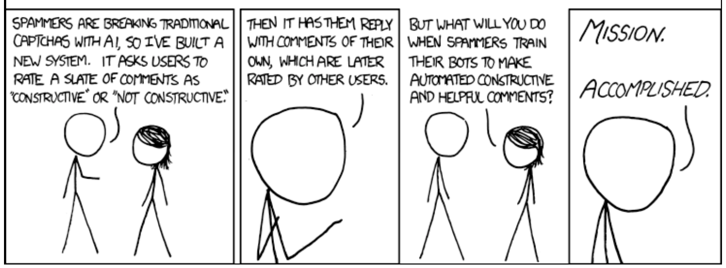
Linear models: Logistic regression

Chapter 3.3



Predicting probabilities

Objective: learn to predict a probability P(y | x) for a binary classification problem using a linear classifier

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

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

For positive examples P(y = +1 | x) = 1 whereas P(y = +1 | x) = 0 for negative examples.

2

Predicting probabilities

Objective: learn to predict a probability P(y | x) for a binary classification problem using a linear classifier

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

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

We'll assume a particular form for f(x).

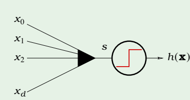
Can we assume that f(x) is linear?

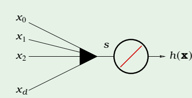
3

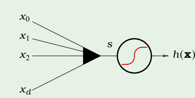
Another linear model

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

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$

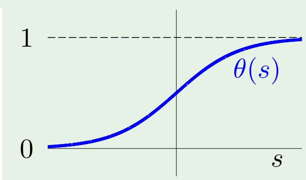






The logistic function (aka squashing function):

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

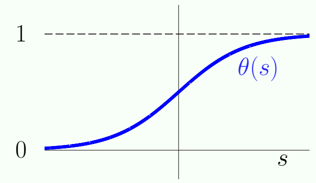


4

Properties of the logistic function

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

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



Predicting probabilities

Fitting the data means finding a good hypothesis h

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

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

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

Predicting probabilities

Fitting the data means finding a good hypothesis h

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

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

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

More compactly: $P(y \mid \mathbf{x}) = \theta(y \cdot \mathbf{w}^{\mathsf{T}}\mathbf{x})$

Is logistic regression really linear?

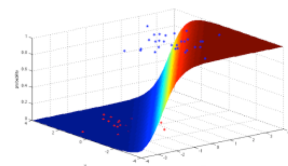
$$P(y = +1|\mathbf{x}) = \frac{\exp(\mathbf{w}^{\mathsf{T}}\mathbf{x})}{\exp(\mathbf{w}^{\mathsf{T}}\mathbf{x}) + 1}$$

$$P(y = -1|\mathbf{x}) = 1 - P(y = +1|\mathbf{x}) = \frac{1}{\exp(\mathbf{w}^{\mathsf{T}}\mathbf{x}) + 1}$$

To figure out how the decision boundary looks like consider:

$$\ln \frac{P(y = +1|\mathbf{x})}{P(y = -1|\mathbf{x})} = \mathbf{w}^{\mathsf{T}}\mathbf{x}$$

i.e. linear!



Maximum likelihood

We will find **w** using the principle of maximum likelihood.

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

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

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

9

Maximizing the likelihood

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

10

Maximizing the likelihood

Summary: maximizing the likelihood is equivalent to

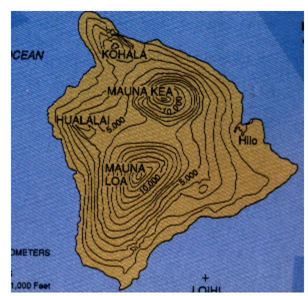
minimize $$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{\ln\left(1 + e^{-y_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right)}_{\mathrm{e}\left(h(\mathbf{x}_n), y_n\right)}$$

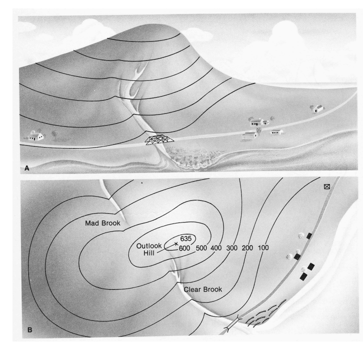
Cross entropy error

11

Digression: gradient descent

Topographical maps can give us some intuition about how to optimize a cost function



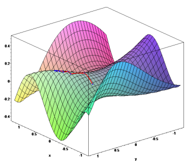


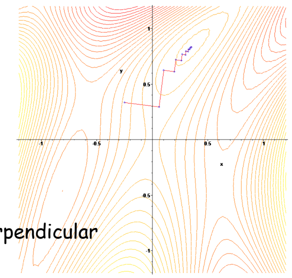
http://www.csus.edu/indiv/s/slaymaker/archives/geol10l/shield1.jpg http://www.sir-ray.com/touro/IMG_0001_NEW.jpg

12

Digression: gradient descent

Given a function E(**w**), the gradient is the direction of steepest ascent
Therefore to minimize E(**w**), take a step in the direction of the negative of the gradient





Notice that the gradient is perpendicular to contours of equal E(**w**)

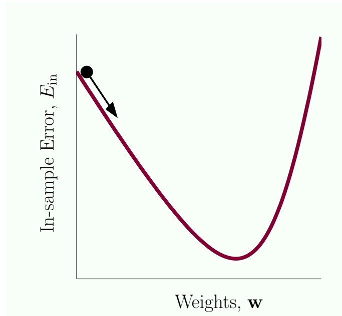
Images from http://en.wikipedia.org/wiki/Gradient_descent

Gradient descent

Gradient descent is an iterative process

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

How to pick $\hat{\mathbf{v}}$?



Gradient descent

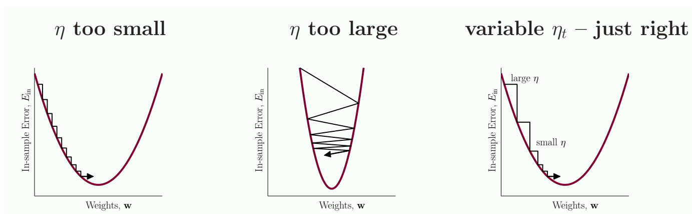
The gradient is the best direction to take to optimize E_{in}(**w**):

$$\begin{aligned}\Delta E_{\text{in}} &= E_{\text{in}}(\mathbf{w}(t+1)) - E_{\text{in}}(\mathbf{w}(t)) \\ &= E_{\text{in}}(\mathbf{w}(t) + \eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(t)) \\ &= \eta\, \underbrace{\nabla E_{\text{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}}} + O(\eta^2)\end{aligned}$$

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

Choosing the step size

The choice of the step size affects the rate of convergence:



Let's use a variable learning rate:

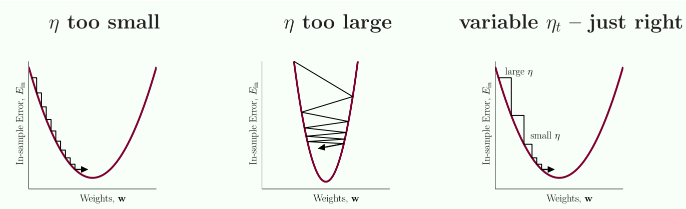
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t\hat{\mathbf{v}}$$

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

When approaching the minimum: $||\nabla E_{\text{in}}(\mathbf{w}(t))|| \to 0$

Choosing the step size

The choice of the step size affects the rate of convergence:



Let's use a variable learning rate:

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

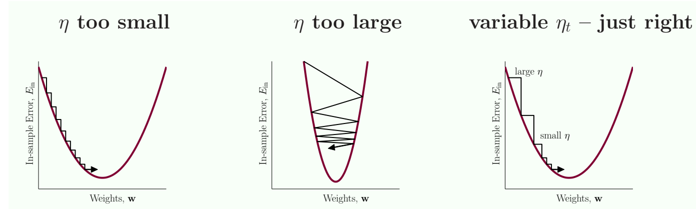
$$\eta_t = \eta \cdot ||\nabla E_{\text{in}}(\mathbf{w}(t))||$$

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

17

The final form of gradient descent

The choice of the step size affects the rate of convergence:



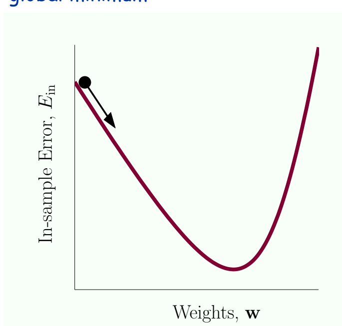
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}}(\mathbf{w}(t))$$

18

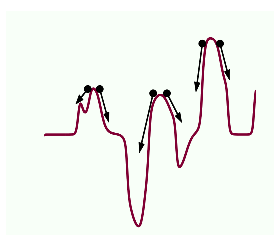
Logistic regression using gradient descent

We will use gradient descent to minimize our error function.

Fortunately, the logistic regression error function has a single global minimum:



So we don't need to worry about getting stuck in local minima



19

Logistic regression using gradient descent

Putting it all together:

Initialize at step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
  Compute the gradient
  $$\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t)).$$
  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.

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

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

20

Logistic regression

Comments:

- Assumptions: i.i.d. data and specific form of P(y | **x**). In practice logistic regression is solved by faster methods than gradient descent
- There is an extension to multi-class classification

21

Stochastic gradient descent

Variation on gradient descent that considers the error for a single training example:

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

Pick a random data point $(\mathbf{x}_*, y_*)$

Run an iteration of GD on $e(\mathbf{w}, \mathbf{x}_*, y_*)$

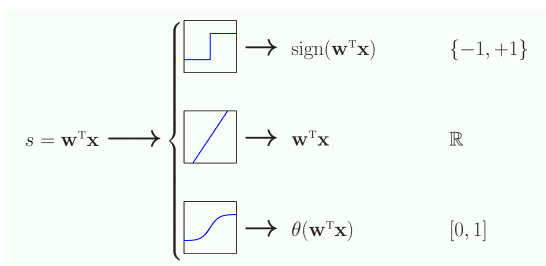
$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} e(\mathbf{w}, \mathbf{x}_*, y_*)$$

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

22

Summary of linear models

Linear methods for classification and regression:



More to come!

23