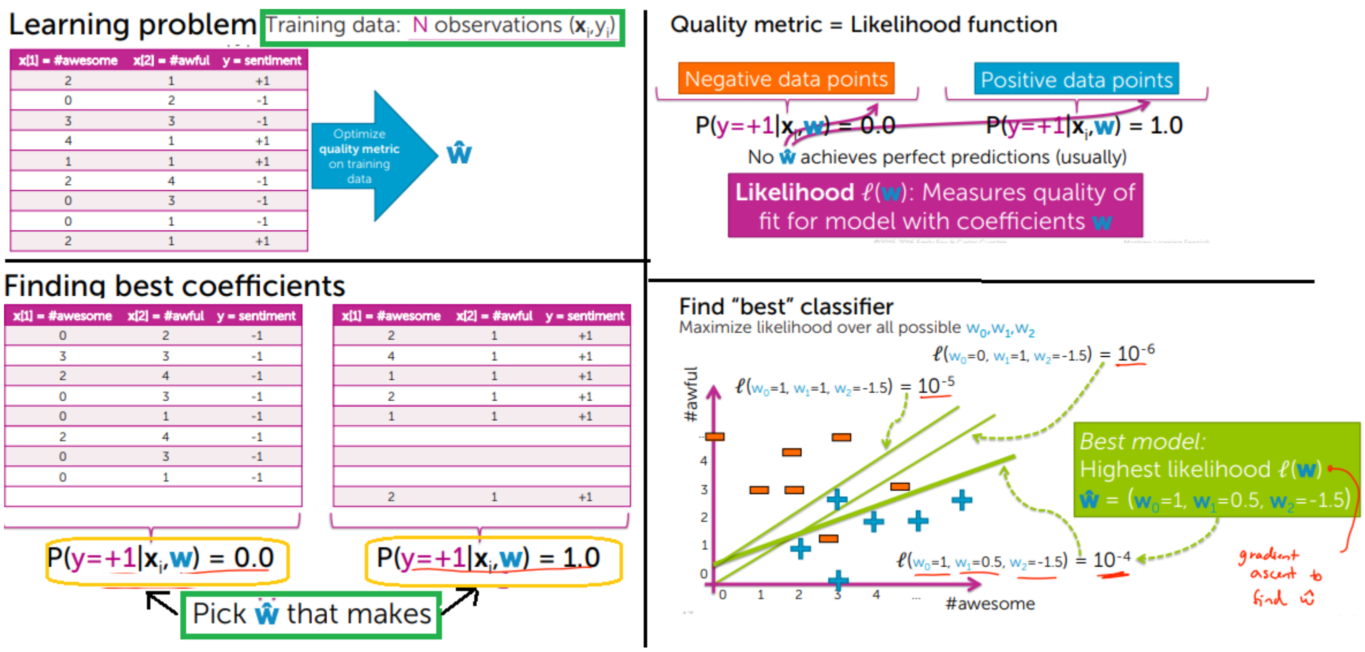
Learning Linear Classifiers

Quality metric for logistic regression : Maximum likelihood estimation

- **Likelihood l(w)** - Measures the quality of the fit of the model with coefficients w.
- It is a small number, but the number closest to 1 is the best fit.
- Higher the probability of a observation , better is it's likelihood and certainity.
  - Consider a datatset with features -> x1-> #awesome, x2 -> #awful, and y -> sentiment.
  - Need to **Find the best coefficients** The ones with high probability. The single table is split into two tables -> one table with **positive y labels** and another with **negative y labels**.
  - For the respective tables need to capture parameters/coefficients (w-hat) with high positive or negative probabilities. P range [0,1];
    - P(y=+1|xi,w) = 0.0 for y = -1;negative predictions.
    - P(y=-1|xi,w) = 1.0 for y = +1; positive predictions.
  - Practically no w-hat achieves prefect predictions usually like 0 or 1. Therefore the probabilities of coefficients very close to 0 or 1 must be considered.

The "best classifier" maximizes likelihood over all possible w0, w1, w2.



Data likelihood

- **Quality metrics : Probability of data**.

Case I. Input x1

- x[1] #awesome = 2 | x[2] #awful = 1 | y sentiment = +1
- If the model is good , should predict y-hat1 = +1;
- For the prediction to be possible, Probability **y=+1** given xi and w must be **maximum**.

Case II. Input x2

- x[1] #awesome = 0 | x[2] #awful = 2 | y sentiment = -1
- If the model is good , should predict y-hat2 = -1;

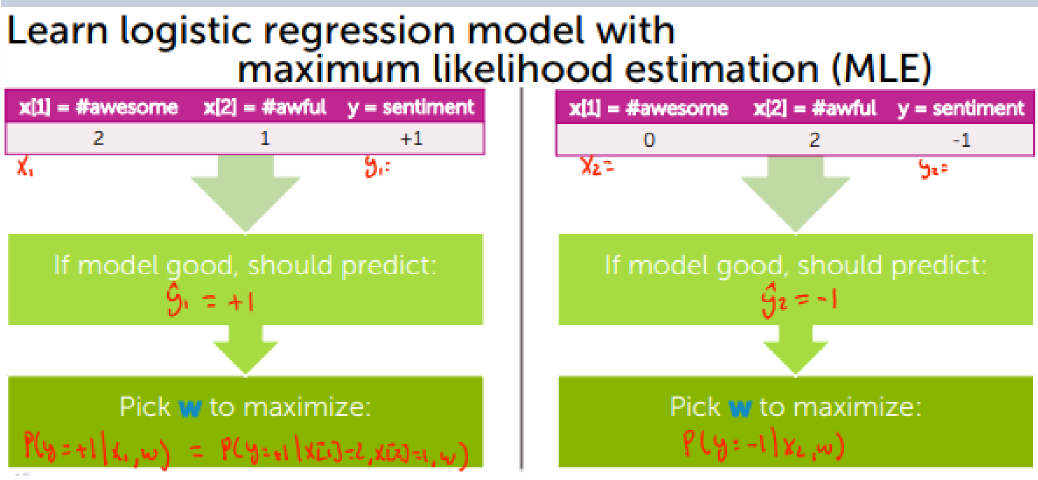
- For the predictions to be possible , Probability **y=-1** given xi and w must be **maximum**.

Maximizing likelihood

- For a given set of obervations, the model must provide a maximum probability that matches the values of the output label in the datatset.
    - if y = +1, then 0.5 < P(y=+1|xi, w) <= 1; Closer the Probability to 1, better the likelihood and certainity that it is negative.
    - if y = -1, then 0 <= P(y=-1|xi, w) <0.5; Closer the probability to 0, better the likelihood and certainity that it is negative.
    - Positive examples y=+1 - maximize the probability of y=+1.
    - Negative examples y=-1 - maximize the probability of y=-1.

The maximum likelihood of the dataset to obtain 'Single Measure of Quality' it is achieved - by multiplying the individual probabilities since rows are 'independent entities(multiply)'.

Goal is to optimize the likelihood by making the product of probabilities as large as possible.



Data point	x[1]	x[2]	y	Choose w to maximize
$\mathbf{x}_1, y_1$	2	1	y_1=+1	P(y=+1|**x**[1]=2, **x**[2]=1,**w**)
$\mathbf{x}_2, y_2$	0	2	-1	P(y=-1|**x**[1]=0, **x**[2]=2,**w**)
$\mathbf{x}_3, y_3$	3	3	-1	P(y=-1|**x**[1]=3, **x**[2]=3,**w**)
$\mathbf{x}_4, y_4$	4	1	+1	P(y=+1|**x**[1]=4, **x**[2]=1,**w**)

Num. of data points → N

$$\ell(w) = \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

← pick w to make this fn. as large as possible

$$\ell(\mathbf{w}) = \underbrace{\underbrace{P(y=+1|\mathbf{x}[1]=2, \mathbf{x}[2]=1,\mathbf{w})}_{P(y_1|\mathbf{x}_1,\mathbf{w})}\ \underbrace{P(y=-1|\mathbf{x}[1]=0, \mathbf{x}[2]=2,\mathbf{w})}_{P(y_2|\mathbf{x}_2,\mathbf{w})}\ \underbrace{P(y=-1|\mathbf{x}[1]=3, \mathbf{x}[2]=3,\mathbf{w})}_{P(y_3|\mathbf{x}_3,\mathbf{w})}\ \underbrace{P(y=+1|\mathbf{x}[1]=4, \mathbf{x}[2]=1,\mathbf{w})}_{P(y_4|\mathbf{x}_4,\mathbf{w})}}$$

Finding the best linear classifier with 'Gradient ascent'

ML Algorithm -> Gradient Ascent

- The quality metrics - likelihood -> product of the probabilities of the true labels given the input sentence and coefficients.
- The goal is to maximize the likelihood function over all possible parameters.
- It has no closed-form solution.
- It has only **Gradient Ascent**.

Gradient Ascent

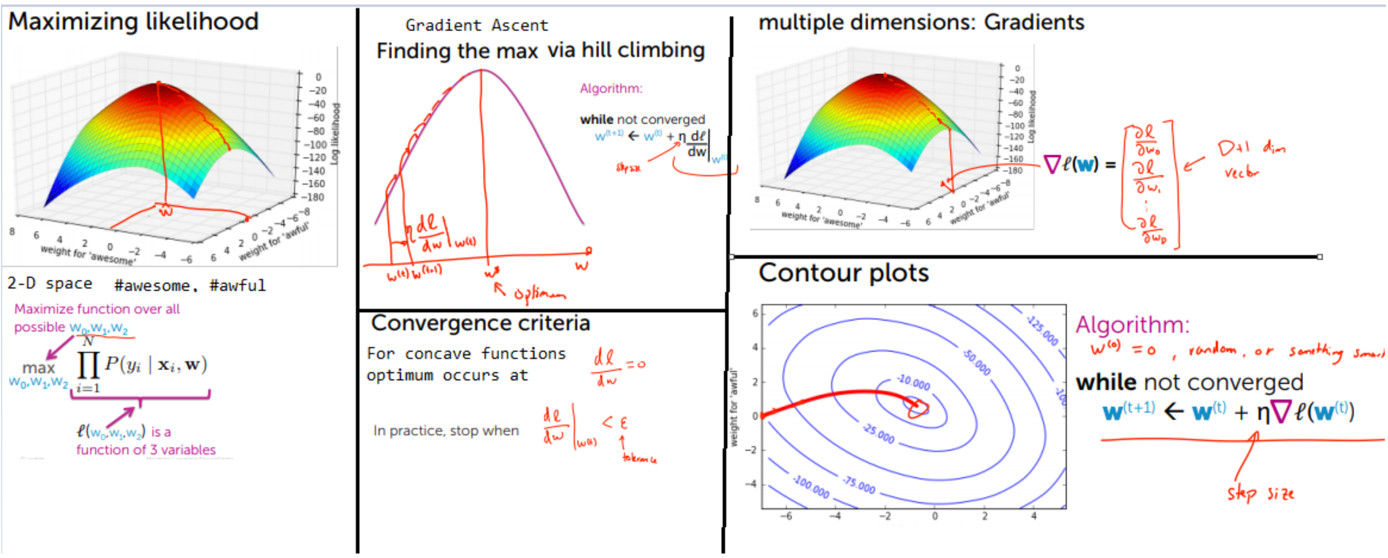
- It is a concave function and the optimum is reached when the derivative is 0. This occurs at the peak of the curve.
- **Algorithm : while not converged w(t+1) <- w(t) + η * (dl/dw)**.
  - (For **2-d space** it is the **derivative** of the **likelihood function**, for **higher dimension space** - the **gradient** of the **likelihood function** must be taken).

Convergence criteria :

- For concave function the optimum occurs at dl/dw = 0 (peak of the curve);
- In practice, the algorithm is stopped at a tolerance point (epsilon - ε) - since it is difficult to achieve 0. dl/dw < ε.

Contour plots

- It is the 2-D representation of the gradient ascent/ descent algorithm. Representation.
- The optimum is achieved when **magnitude of the coefficients/parameters decreases** and the **likelihood function reaches the optimum**.



Derivative of (log-) likelihood

- dl(w) / dwj = Σ(i=1---N) (feature values) * (truth - prediction);
  - truth -> Indicator function -> 1 if yi = +1, else 0 when yi = -1.

Derivative of (log-)likelihood

Sum over data points

Feature value

Difference between truth and prediction

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - \underbrace{P(y = +1 \mid \mathbf{x}_i, \mathbf{w})}_{\text{predict } x_i \text{ is positive}}\Big)$$

Indicator function:

$$\mathbb{1}[y_i = +1] = \begin{cases} 1 & \text{if } y_i = +1 \\ 0 & \text{if } y_i = -1 \end{cases}$$

Example - Computing derivative

- Consider the initial coefficients with values
  - w0(t) = 0, w1(t) = 1, w2(t) = -2
- The derivative is individually calculated for each row.
  - example x[1] = 2, x[2] = 1, y = +1, P(y=+1|x, w) = 0.5
    - Contribution to derivative w = feature *(|y=+1| - P)
      - feature -x[1] = 2;
      - |y=+1| = 1(if the y =+1 then 1; if y = -1 then 0)
    - Contribution to derivative w = 2 * (1 - 0.5) = 1;
  - example x[1] = 4, x[2] = 1, y = +1, P(y=+1|x, w) = 0.88
    - Contribution to derivative w = 4(1 - 0.88) = 0.48
- Total derivative = sum of all individual derivatives = 0.5 + 0.48 = 0.98
- Algorithm : w1(t+1) <- w1(t) + η *(dl/dw) = 1 + 0.1* 0.98 = 1.098; (η = 0.1);
- updating a particular parameter w1. Therefore the feature 1 was considered.

$$\frac{\partial \ell(\mathbf{w}^{(t)})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\Big)$$

$w^{(t)}$:

$w_0^{(t)}$	0
$w_1^{(t)}$	1
$w_2^{(t)}$	-2

$\frac{\partial \ell}{\partial w_1}$

$h_1(x)$ = # awesome

x[1]	x[2]	y	P(y=+1|x_i,w)	Contribution to derivative for w_1
2	1	+1	0.5	2(1-0.5) = 1
0	2	-1	0.02	0(0-0.02) = 0
3	3	-1	0.05	3(0-0.05) = -0.15
4	1	+1	0.88	4(1-0.88) = 0.48

Total derivative:

$$\frac{\partial \ell(w^{(t)})}{\partial w_1} = 1 + 0 - 0.15 + 0.48 = 1.33$$

$$w_1^{(t+1)} = w_1^{(t)} + \eta \frac{\partial \ell(w^{(t)})}{\partial w_1} \quad \Big| \quad \eta = 0.1$$

$$= 1 + 0.1 \times 1.33 = 1.133$$

Interpretation of the Derivative of (log-) likelihood

- Consider the derivative of likelihood equation. Assume feature value (hj(xi) = 1)

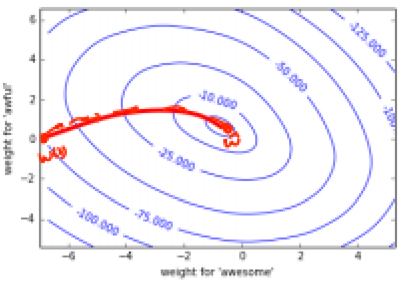
- In case the **labelled output y** and **prediction** are the same, then the difference between the **truth and prediction = 0**.
  - yi = +1 & P(y=+1|xi, w) = 1, remains the same.
  - yi = -1 & P(y=+1|xi, w) = 0, remains the same.
- In case the **labelled output y** and **prediction** are different, the there is a difference term that leads to an increase or decrease in the next iteration parameters value.
  - yi = +1 & P(y=+1|xi, w) = 0, [truth - prediction] = 1 - 0 = 1;
    - therefore the wj must increase. Score(xi) must be larger.
    - P(y=+1|xi, w)-> increases.
  - yi = -1 & P(y=+1|xi, w) = 1, [truth - prediction] = 0 - 1 = -1;
    - therefore the wj must decrease. Score(xi) must be smaller.
    - P(y=+1|xi, w)-> decreases.

Sum over data points — Feature value — Difference between truth and prediction

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\underbrace{\left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\right)}_{\Delta_i}$$

If $h_j(\mathbf{x}_i)=1$:	$P(y=+1 \mid \mathbf{x}_i,\mathbf{w}) \approx 1$	$P(y=+1 \mid \mathbf{x}_i,\mathbf{w}) \approx 0$
$y_i=+1$	$\Delta_i = (1-1) \approx 0$ ↳ don't change anything)	$\Delta_i \approx 1 \Rightarrow$ increase w_j $\Rightarrow$ Score(x_i) becomes larger $\Rightarrow P(y=+1 \mid x_i, w)$ increases
$y_i=-1$	$\Delta_i = -1 \Rightarrow w_j$ to decrease $\Rightarrow$ Score(x_i) decreases $\Rightarrow P(y=+1 \mid x_i, w)$ decrease	$\Delta_i \approx 0 \Rightarrow$ don't change anything

Gradient Ascent - Logistic Regression



init $\mathbf{w}^{(1)}=0$ (or randomly, or smartly), t=1

while $\|\nabla \ell(\mathbf{w}^{(t)})\| > \varepsilon$ ← tolerance

$\frac{1}{1+e^{-\mathbf{w}^{(t)T} h(x_i)}}$

for j=0,...,D ← coefficient

partial[j] = $\sum_{i=1}^{N} h_j(\mathbf{x}_i)\left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\right)$

$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta$ partial[j]

t ← t + 1

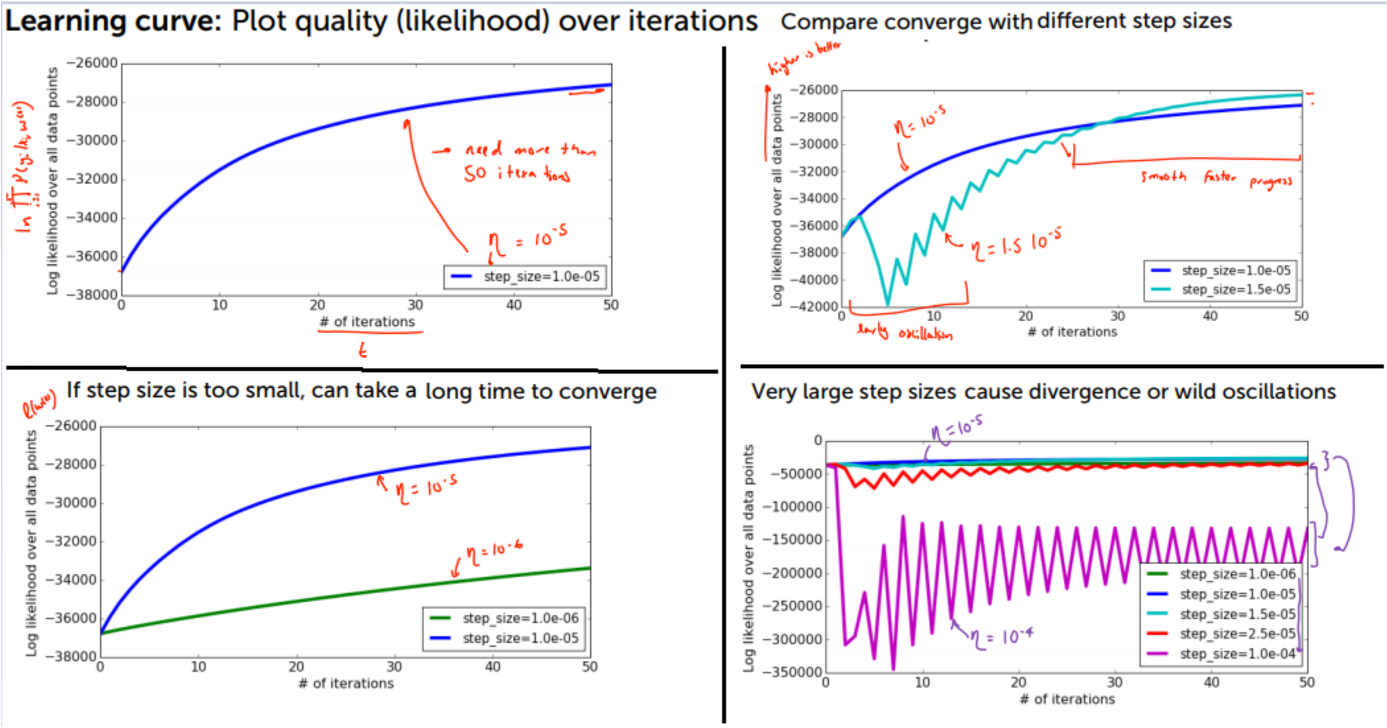
step size $\quad \frac{\partial \ell(w^{(t)})}{\partial w_j}$

Choosing step-size η

- Picking the step-size requires a lot of trail and error.
- Try several values that are exponentially spaced -> 10^-5, 10^-4, etc. **Plot learning curves - to see the convergence**.
  - find one η that is too small (smooth but moving too slowly).
  - find one η that is too large (oscillations and divergence).
- Try values in between to find the 'best' η.
- **Advanced tip -> can try 'step-sizes' that decrease with iterations.**.

Choosing η : η(t) = η(0) / t ;

- η(0) -> constant step size derived from desired optimum function.
- t -> number of iterations.
- η1 = η0/1, ..., η10 = η0/10,..



Deriving gradient of logistic regression (Advanced)

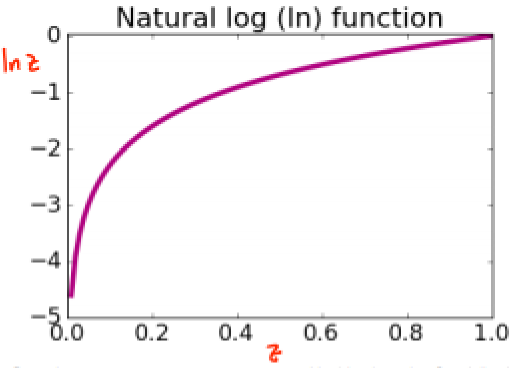
- Goal - choose coefficients **w** maximizing likelihood.

$$\ell(\mathbf{w}) = \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

- Math is simplified by using log-likelihood - taking the natural log -ln/log-e.

$$\ell\ell(\mathbf{w}) = \ln \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

natural log



Graph f(x) = ln(2x).

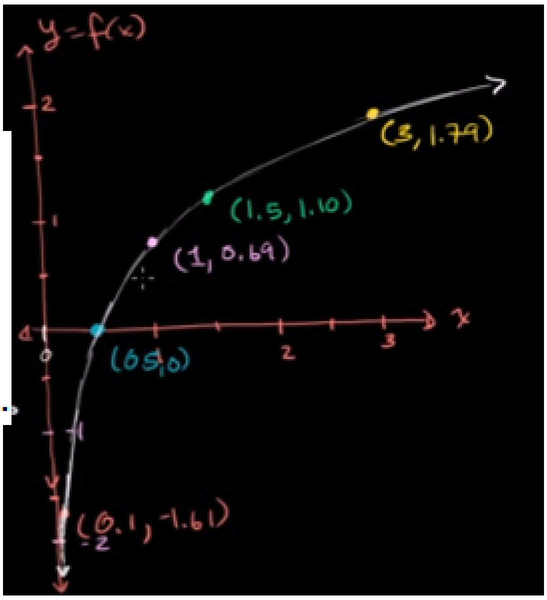
x	$y=f(x)$
0.1	-1.61
0.5	0
1	0.69
1.5	1.10
3	1.79

The natural log function increases with the input magnitude and decreases with decrease in input, but it approaches 0 but never becomes 0.

$2x > 0$

$x > 0 \Leftarrow$ Domain

$$e^{-1,000,000,000} = \frac{1}{e^{1,000,000,000}}$$



- In log the product becomes sum and divison becomed subtraction.
- Log doesn't change the maximum. Since the goal is to maximize the likelihood. **w-hat = ln(w-hat)**.

Step-1 : Using log to turn products into sums

The log of the product of likelihoods becomes the sum of the logs:

$$\ell\ell(\mathbf{w}) = \ln \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{N} \ln P(y_i \mid x_i, w)$$

Step-2 : Rewrite log-likelihood (Introduce indicators -> sign)

$$\ell\ell(\mathbf{w}) = \sum_{i=1}^{N} \ln P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

$$= \sum_{i=1}^{N} \left[\mathbb{1}[y_i = +1] \ln P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) + \mathbb{1}[y_i = -1] \ln P(y = -1 \mid \mathbf{x}_i, \mathbf{w})\right]$$

Step-3 : Values for probabilities

$P(y=+1|\mathbf{x},\mathbf{w}) + P(y=-1|\mathbf{x},\mathbf{w}) = 1$ $\quad P(y=+1|\mathbf{x},\mathbf{w}) = \dfrac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$

$$P(y = -1 \mid \mathbf{x}, \mathbf{w}) = \frac{e^{-\mathbf{w}^\top h(\mathbf{x})}}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$$

Step-4 : Plugging in logistic function 1-datapoint

$$\ell\ell(\mathbf{w}) = \mathbb{1}[y_i = +1] \ln P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) + \mathbb{1}[y_i = -1] \ln P(y = -1 \mid \mathbf{x}_i, \mathbf{w})$$

$$= \mathbb{1}[y_i = +1] \ln \frac{1}{1+e^{-w^T h(x_i)}} + (1 - \mathbb{1}[y_i = +1]) \ln \frac{e^{-w^T h(x_i)}}{1+e^{-w^T h(x_i)}}$$

$$= -\mathbb{1}[y_i = +1] \ln(1 + e^{-w^T h(x_i)}) + (1 - \mathbb{1}[y_i = +1])[-w^T h(x_i) - \ln(1 + e^{-w^T h(x_i)})]$$

$$= -(1 - \mathbb{1}[y_i = +1]) w^T h(x_i) - \ln(1 + e^{-w^T h(x_i)})$$

$\ln e^a = a$

$\mathbb{1}[y_i = -1] = 1 - \mathbb{1}[y_i = +1]$

$\ln \frac{1}{1+e^{-w^T h(x_i)}} = -\ln(1 + e^{-w^T h(x_i)})$

$\ln \frac{e^{-w^T h(x_i)}}{1+e^{-w^T h(x_i)}} = \underbrace{\ln e^{-w^T h(x_i)}}_{-w^T h(x_i)} - \ln(1 + e^{-w^T h(x_i)})$

Step-5 : Gradient for 1-datapoint

$$\ell\ell(\mathbf{w}) = -(1 - \mathbb{1}[y_i = +1])\mathbf{w}^\top h(\mathbf{x}_i) - \ln\left(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}\right)$$

$$\frac{\partial \ell\ell}{\partial w_j} = -(1 - \mathbb{1}[y_i = +1]) \frac{\partial}{\partial w_j} w^T h(x_i) - \frac{\partial}{\partial w_j} \ln(1 + e^{-w^T h(x_i)})$$

$$= -(1 - \mathbb{1}[y_i = +1]) h_j(x_i) + h_j(x_i) P(y = -1 \mid x_i, w)$$

$$= h_j(x_i)\left[\mathbb{1}[y_i = +1] - P(y = +1 \mid x_i, w)\right]$$

$\frac{\partial}{\partial w_j} w^T h(x_i) = h_j(x_i)$

$\frac{\partial}{\partial w_j} \ln(1 + e^{-w^T h(x_i)})$

$= -h_j(x_i) \underbrace{\frac{e^{-w^T h(x_i)}}{1 + e^{-w^T h(x_i)}}}_{P(y=-1|x_i,w)}$

Step-6 : Gradient for all datapoints

- Gradient for one data point:

$$h_j(\mathbf{x}_i)\left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\right)$$

- Adding over data points:

$$\frac{\partial \ell\ell}{\partial w_j} = \sum_{i=1}^{N} h_j(x_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid x_i, w)\right)$$

Quiz

1.

(True/False) A linear classifier can only learn positive coefficients.

- ◯ True
- ◉ False

2.

(True/False) In order to train a logistic regression model, we find the weights that maximize the likelihood of the model.

- ◉ True
- ◯ False

3.

(True/False) The data likelihood is the product of the probability of the inputs $\mathbf{x}$ given the weights $\mathbf{w}$ and response y.

- ◯ True
- ◉ False

Given a particular weights vector, a training observation, and its true label, the logistic regression model specifies a probability $P(y_{\text{pred}} = y_{\text{true}} \mid \mathbf{w}, \mathbf{x})$. We want to maximize all of these probabilities. The likelihood is the product of these probabilities.

4.

Questions 4 and 5 refer to the following scenario.

Consider the setting where our inputs are 1-dimensional. We have data

x	y
2.5	+1
0.3	-1
2.8	+1
0.5	+1

and the current estimates of the weights are $w_0 = 0$ and $w_1 = 1$. (w_0: the intercept, w_1: the weight for x).

Calculate the likelihood of this data. Round your answer to 2 decimal places.

0.23

$$P(y_1 = +1|x_1, w)P(y_2 = -1|x_2, w)P(y_3 = +1|x_3, w)P(y_4 = +1|x_4, w)$$
$$= \frac{1}{1+e^{-2.5}} \frac{e^{-0.3}}{1+e^{-0.3}} \frac{1}{1+e^{-2.8}} \frac{1}{1+e^{-0.5}}$$
$$= 0.230765 \cdots$$

In [1]:

```
import numpy as np

dummy_feature_matrix = np.array([[1.,2.5], [1.,0.3], [1.,2.8], [1.,0.5]])
dummy_coefficients = np.array([0., 1.])
sentiment = np.array([1., -1., 1., 1.])

def predict_probability(feature_matrix, coefficients):
    # Take dot product of feature_matrix and coefficients
    # YOUR CODE HERE
    scores = np.dot(feature_matrix, coefficients)

    # Compute P(y_i = +1 | x_i, w) using the link function
    # YOUR CODE HERE
    predictions = 1. / (1 + np.exp(-scores))

    # return predictions
    return predictions

def compute_data_likelihood(sentiment, probability):
    indicator = (sentiment==+1)
    print "Indicator: ", indicator

    print "Probability of +1: ", probability

    # probability of (-1)= (1 - probability of +1)
    probability[~indicator] = 1 - probability[~indicator]
    print "Maximum likelihood: ", probability
    return np.prod(probability)


probability = predict_probability(dummy_feature_matrix, dummy_coefficients)
print probability

data_likelihood = compute_data_likelihood(sentiment, probability)
print data_likelihood
```

```
[ 0.92414182  0.57444252  0.94267582  0.62245933]
Indicator:  [ True False  True  True]
Probability of +1:  [ 0.92414182  0.57444252  0.94267582  0.62245933]
Maximum likelihood:  [ 0.92414182  0.42555748  0.94267582  0.62245933]
0.230765141474
```

5.

Refer to the scenario given in Question 4 to answer the following:

Calculate the derivative of the log likelihood with respect to w_1. Round your answer to 2 decimal places.

0.37

$$\begin{aligned}\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_1} &= \sum_{i=1}^{4} h_1(\mathbf{x}_i)\Big(\mathbf{1}[y_i = +1] - P(y_i = +1|\mathbf{x}_i, \mathbf{w})\Big) \\ &= 2.5\left(1 - \frac{1}{1+e^{-2.5}}\right) + 0.3\left(0 - \frac{1}{1+e^{-0.3}}\right) \\ &\quad + 2.8\left(1 - \frac{1}{1+e^{-2.8}}\right) + 0.5\left(1 - \frac{1}{1+e^{-0.5}}\right) \\ &= 0.366591\cdots\end{aligned}$$

In [2]:

```
def compute_derivative_log_likelihood(feature_vector, sentiment, probability):
    """ Compute derivative of feature vector
    - In this case, the feature vector with respect to w1
    """
    indicator = (sentiment==+1)
    print "Indicator: ", indicator

    # Contribution to derivative for w1
    contribution =  feature_vector * (indicator - probability)
    print "Contribution: ", contribution

    return np.sum(contribution)

probability = predict_probability(dummy_feature_matrix, dummy_coefficients)
print probability

# In this case, the feature vector (dummy_feature_matrix[:, 1]) with respect to w1
compute_derivative_log_likelihood(dummy_feature_matrix[:, 1], sentiment, probability)
```

```
[ 0.92414182  0.57444252  0.94267582  0.62245933]
Indicator:  [ True False  True  True]
Contribution:  [ 0.18964545 -0.17233276  0.16050769  0.18877033]
```

Out[2]:

```
0.36659072192551606
```

6.

Which of the following is true about gradient ascent? Select all that apply.

- [x] It is an iterative algorithm
- [ ] It only updates a few of the parameters, not all of them
- [x] It finds the maximum by "hill climbing"

In [ ]: