2 Univariate linear regression

Contents

2.1 Representation: Single Features

Notation. For univariate linear regression, we have

$m :=\#$ training examples, $x :=$input variable (or features), $y :=$output (or target) variable

$(x, y) :=$one training example

$\left(x^{(i)}, y^{(i)}\right) := i^{th}$ training example

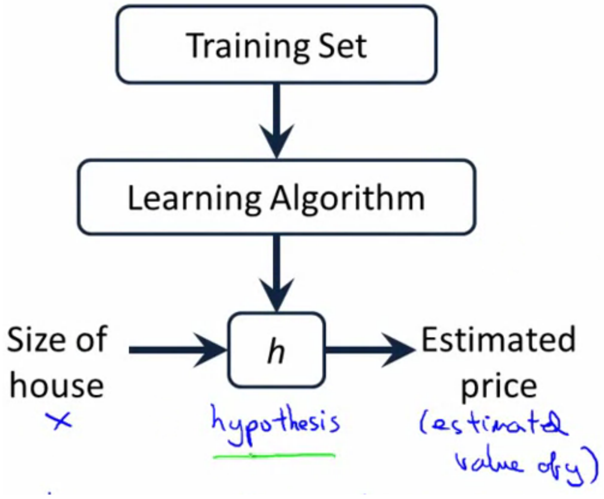


Figure 1: *Representation*: In univariate linear regression, our **hypothesis function** has the general form $h_\theta(x) = \theta_0 + \theta_1 x$. We determine the values for the parameters θ_0 and θ_1 by running our **training set** on a **learning algorithm**.

2.2 Evaluation: Cost Function for univariate linear regression

Idea: We use a cost function for learning the parameters: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training set. That is, minimise:

$$\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \iff \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Note. $\frac{1}{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ is equal to the means of the squares of the residual, $\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$.

Remark. The mean is halved as a convenience of the gradient descent computation (as the derivative term of the square function will cancel out with the $\frac{1}{2}$; note also that dividing by a constant $\frac{1}{2m}$ is a linear operator, and so it does not effect which parameters "score" best).

Definition 2.1. The **cost** (**squared error**) **function for univariate linear regression** of a training set is

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

We seek $\min_{h_\theta} J(\theta_0, \theta_1) \iff \min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$.

Note. The cost function uses *all* the data in the training set.

2.2.1 Cost Function Intuition

Example. ***Case*** $\theta_0 = 0$

Consider a simplified version of the straight line model, with $\theta_0 = 0$:

$$h_\theta(x) = \theta_1 x \text{ and } J(\theta_0, \theta_1) = J(\theta_1)$$

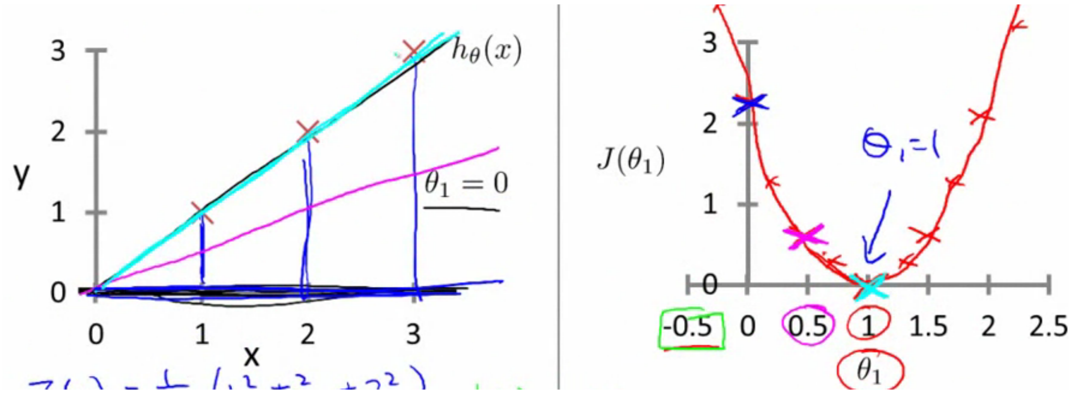


Figure 2: We get the following lines for:
$\theta_1 = 1$: Cyan line
$\theta_1 = 0.5$: Magenta line
$\theta_1 = 0$: Blue line
Observe $\theta_1 = 1$ minimises $J(\theta_1)$.

Example. ***More general case***

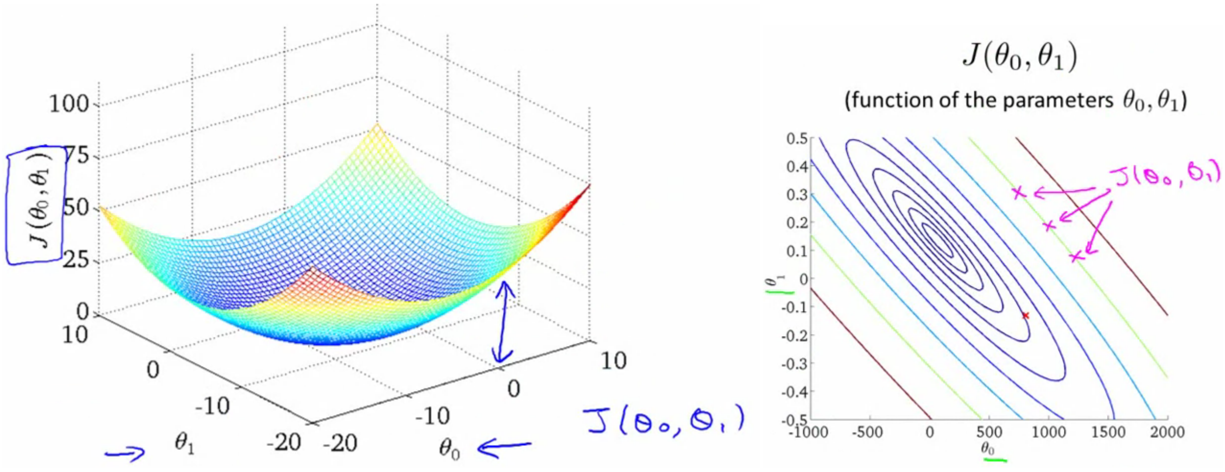


Figure 3: 3D representation and it's contour plot.

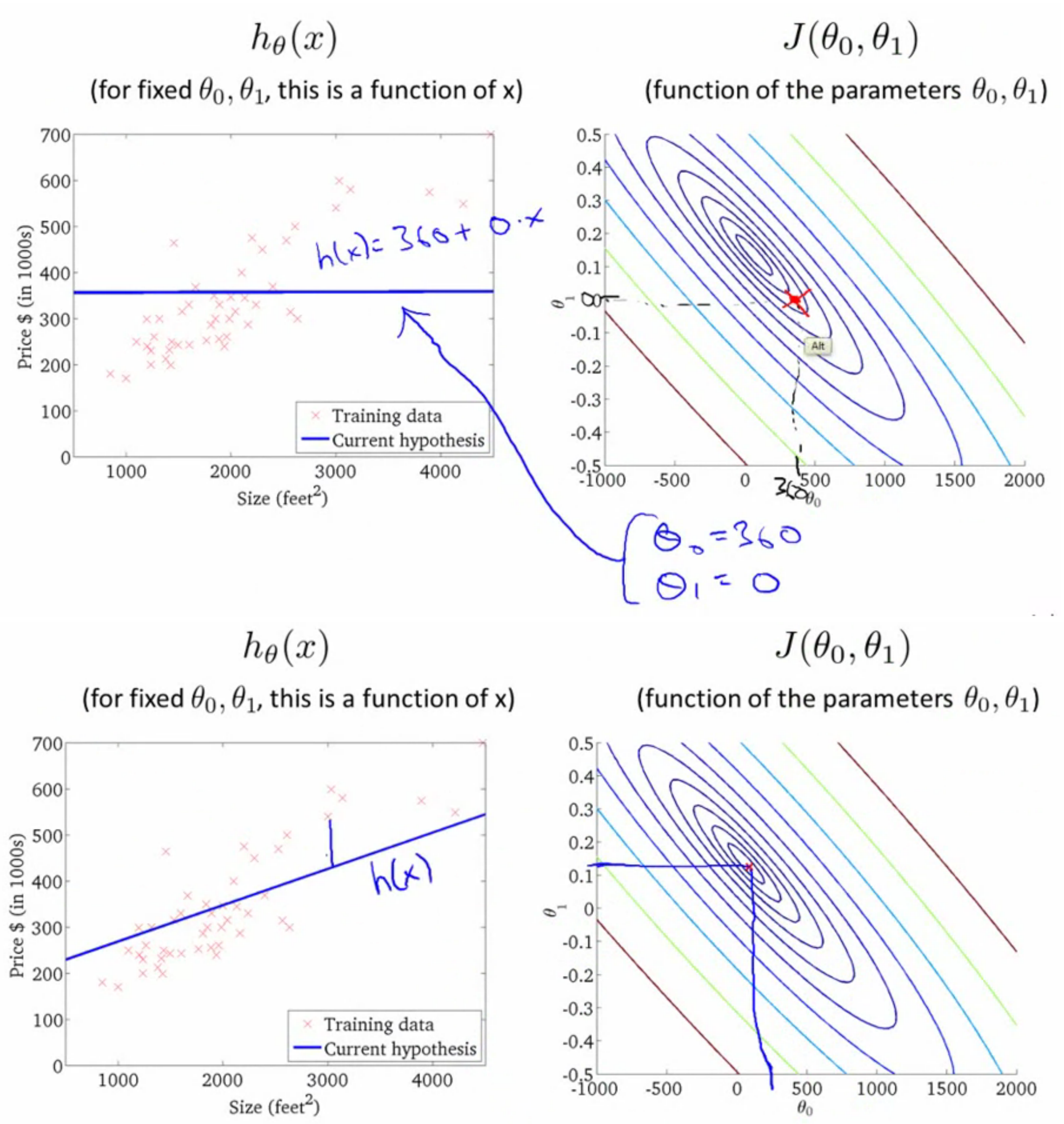


Figure 4: The closer the values of θ_0 and θ_1 to the minimum (in the contour plot), the better the line of best fit is.

Remark. We seek an algorithm that efficiently minimises the cost function J.

2.3 Optimisation: Gradient Descent univariate linear regression

Idea: You're at the top of a hill and you seek a route to the bottom by *always choosing the lowest step.*

```
Algorithm 1. Gradient Descent
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (simultaneously update $j = 0$ and $j = 1$)
}
for $\alpha$ some (positive) learning rate (aka. step size) and some initial guess for $\theta_0, \theta_1$.
```

Note. **Simultaneously update** means:

`temp0` $:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$; `temp1` $:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

θ_0 :=`temp0`; θ_1 :=`temp1`.

Remark. $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is equal to the slope of the tangent.

Remark. ***WARNING using gradient descent***

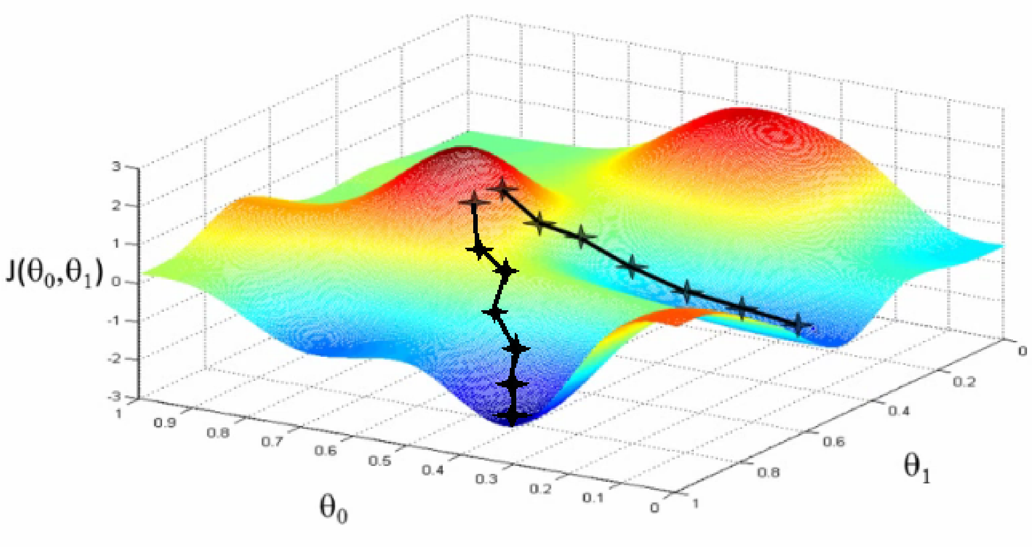
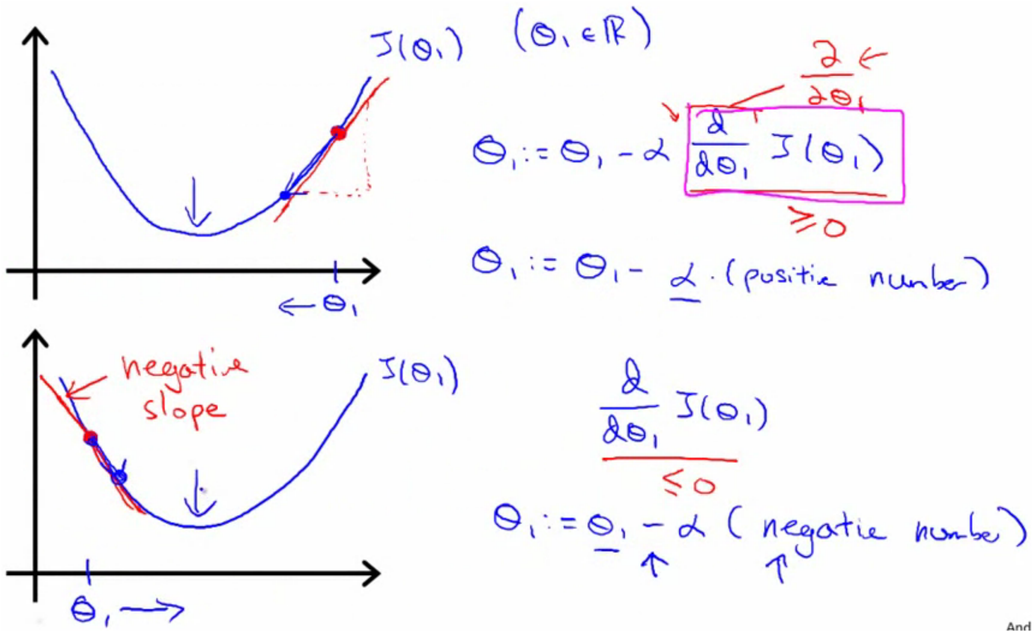


Figure 5: Where you start (your initial guess) can affect which (local) minimum you arrive at, and this may not necessarily be the global minimum. Observe if we are at a local minimum, one step of gradient descent does nothing.

2.3.1 Intuition behind gradient descent

Consider a 1D case: $J(\theta_1)$ for $\theta_1 \in \mathbb{R}$, so $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$. Thus



$J(\theta_1)$
$(\theta_1 \in \mathbb{R})$
$\frac{\partial}{\partial \theta_1}$
$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)}$
≥ 0
$\theta_1 := \theta_1 - \alpha \cdot$ (positie number)
$\leftarrow \theta_1$
negative slope
$J(\theta_1)$
$\frac{d}{d\theta_1} J(\theta_1)$
≤ 0
$\theta_1 := \theta_1 - \alpha$ (negatie number)
$\theta_1 \rightarrow$
And

Figure 6: Note, the derivative term always has the correct sign for sending us in the correct direction (towards the minimum). Also note, gradient descent automatically takes smaller steps with time (as we approach the local minimum, the derivative get smaller), so there is no need to decrease α over time.

Remark. ***Choosing a sensible learning rate,α.***

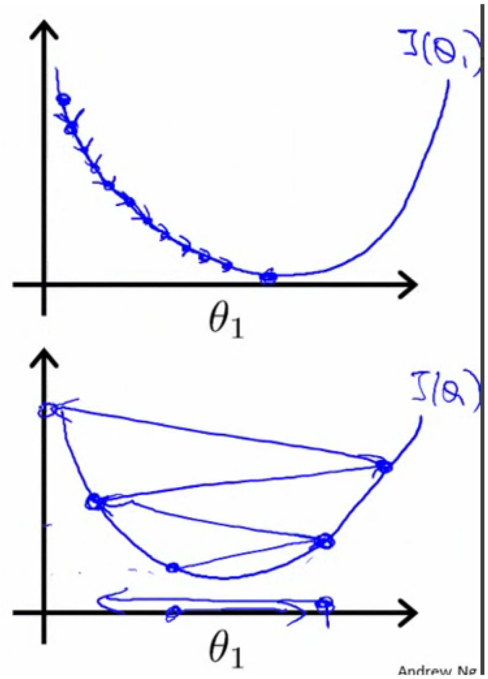


Figure 7: TOP: if α is too *small*, gradient descent can be slow.
BOTTOM: if α is too *large*, gradient descent can overshoot the minimum and can fail to converge (and can even diverge).

2.3.2 Applying the cost function for gradient descent - a first learning algorithm for linear regression

Observe by linearity of the derivative,

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J\left(\theta_0,\theta_1\right) &= \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2\\ &= \frac{1}{2m}\frac{\partial}{\partial\theta_j}\sum_{i=1}^{m}\left(\theta_0+\theta_1x^{(i)}-y^{(i)}\right)^2\end{aligned}$$

Also by the chain rule, we have

$$\begin{aligned}\frac{\partial}{\partial\theta_0}J\left(\theta_0,\theta_1\right) &= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)\\ \frac{\partial}{\partial\theta_1}J\left(\theta_0,\theta_1\right) &= \frac{1}{m}\sum_{i=1}^{m}x^{(i)}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)\end{aligned}$$

Hence

Algorithm 2. ***Batch Gradient Descent***

```
repeat until convergence {
$\theta_0 := \theta_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)\right]$
$\theta_1 := \theta_1 - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}x^{(i)}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)\right]$
(simultaneously update $j = 0$ and $j = 1$)
}
```

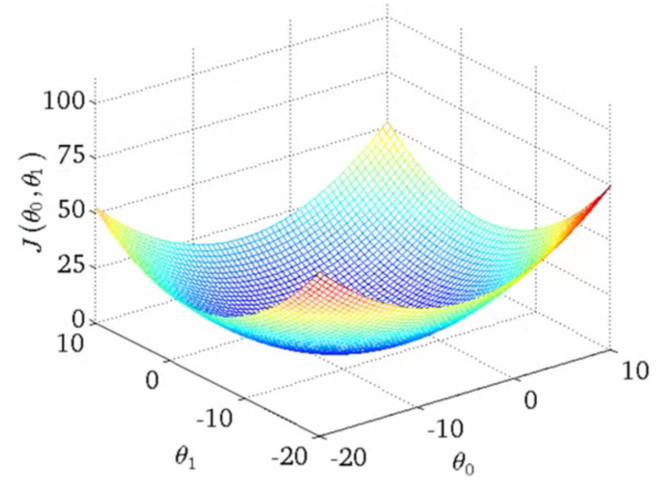


Figure 8: Recall that a problem with general gradient descent is that initial conditions can affect the (local) minimum you arrive at; however *the Cost Function for gradient descent is* always *a bowl shape* (a **convex function**), which has *one global minimum*.

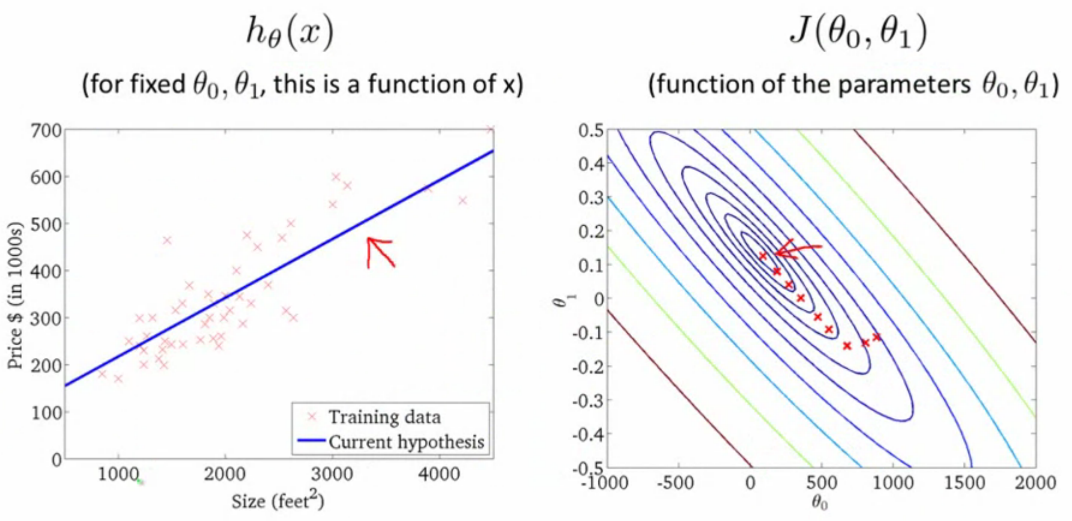


Figure 9: This results in a *fast approach* to the *global* minimum.

Remark. The "Batch" refers to the fact that over each step we look at all the training data (each step compute over m training examples). Some non-batch versions exist, which look at small data subsets (to use when m is too large).