Multiple Regression - Linear Regression with multiple features

Linear fit might not be the best fit to the data.

The data can be fit with a quadratic function or even a polynomial function.

- Simple Linear Regression - function f(x) = w(0) + w(1) *x; Model y(i) = w(0) + w(1)* x(i) + ε(i)
- Quadratic function - f(x) = w(0) + w(1) *x + w(2)* x^2
- Higher order polynomial - f(x) = w(0) + w(1) *x + w(2)* x^2 + ... + w(p) * x^p

Polynomial regression

Model:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$

treat as different **features**

feature 1 = 1 (constant)	*parameter* 1 = w_0
feature 2 = x	*parameter* 2 = w_1
feature 3 = x^2	*parameter* 3 = w_2
...	...
feature p+1 = x^p	*parameter* p+1 = w_p

In the polynomial regression each function of the input is treated as a separate feature.

Example : Detrending

- A dataset - log(house price) v/s Month
- Analysis captures : The house prices increase with time (linear relationship); The house prices lower in Nov-Dec (Seasonal);
- Therefore - Seasonal and Linear with time.

Model:

$$y_i = w_0 + w_1 t_i + w_2 \sin(2\pi t_i / 12 - \Phi) + \varepsilon_i$$

Linear trend

Seasonal component = Sinusoid with period 12 (resets annually)

Unknown phase/shift

Trigonometric identity: $\sin(a-b)=\sin(a)\cos(b)-\cos(a)\sin(b)$

$\rightarrow \sin(2\pi t_i / 12 - \Phi) = \sin(2\pi t_i / 12)\cos(\Phi) - \cos(2\pi t_i / 12)\sin(\Phi)$

Equivalently,

$$y_i = w_0 + w_1 t_i + w_2 \sin(2\pi t_i / 12) + w_3 \cos(2\pi t_i / 12) + \varepsilon_i$$

feature 1 = 1 (constant)

feature 2 = t

feature 3 = $\sin(2\pi t/12)$

feature 4 = $\cos(2\pi t/12)$

- The dataset is fit fit a 5th order polynomial.

Generic Model :

Model:

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i$$

j^{th} *feature*

j^{th} *regression coefficient* or *weight*

General notation

Output: y ← scalar

Inputs: $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[d])$

d-dim vector

Notational conventions:

- $\mathbf{x}[j]$ = j^{th} input (*scalar*)
- $h_j(\mathbf{x})$ = j^{th} feature (*scalar*)
- $\mathbf{x}_i$ = input of i^{th} data point (*vector*)
- $\mathbf{x}_i[j]$ = j^{th} input of i^{th} data point (*scalar*)

More on notation

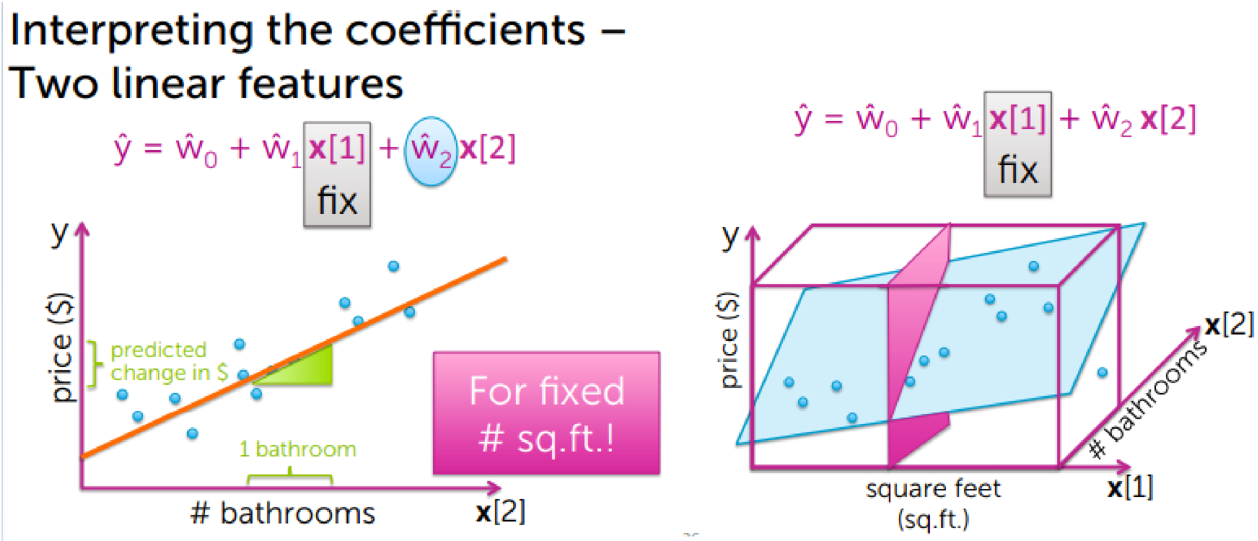
observations $(\mathbf{x}_i, y_i)$: N

inputs $\mathbf{x}[j]$: d

features $h_j(\mathbf{x})$: D

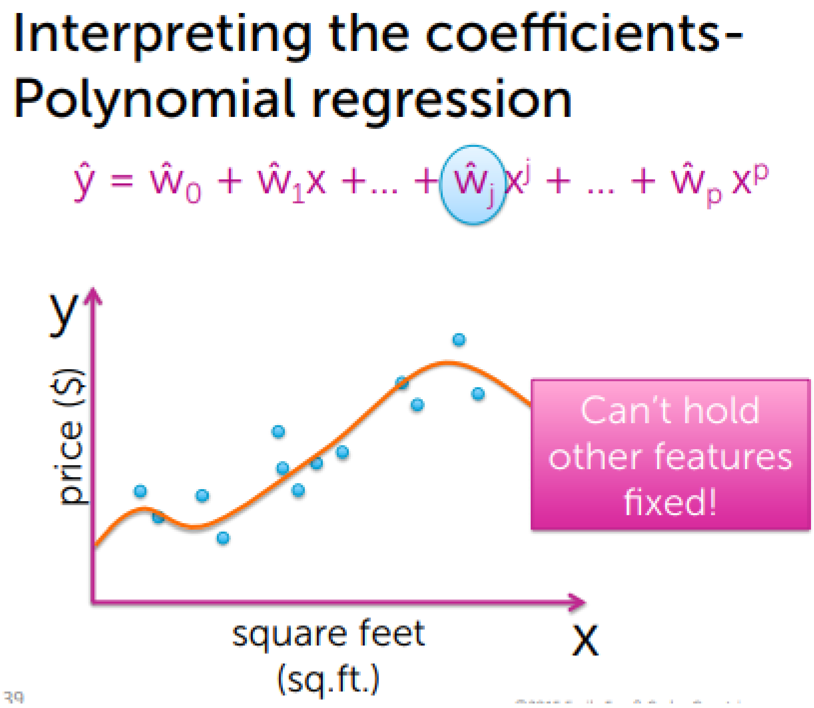
Interpreting the coefficients:

- While a model has multiple features the when interpreting the coefficient need to consider fix all the other inputs to the model and look at the focus feature.
- In case a house data - depends on features -> sqft of house and # of bathrooms.
- x[1] -> sqft, x[2] -> # bathrooms



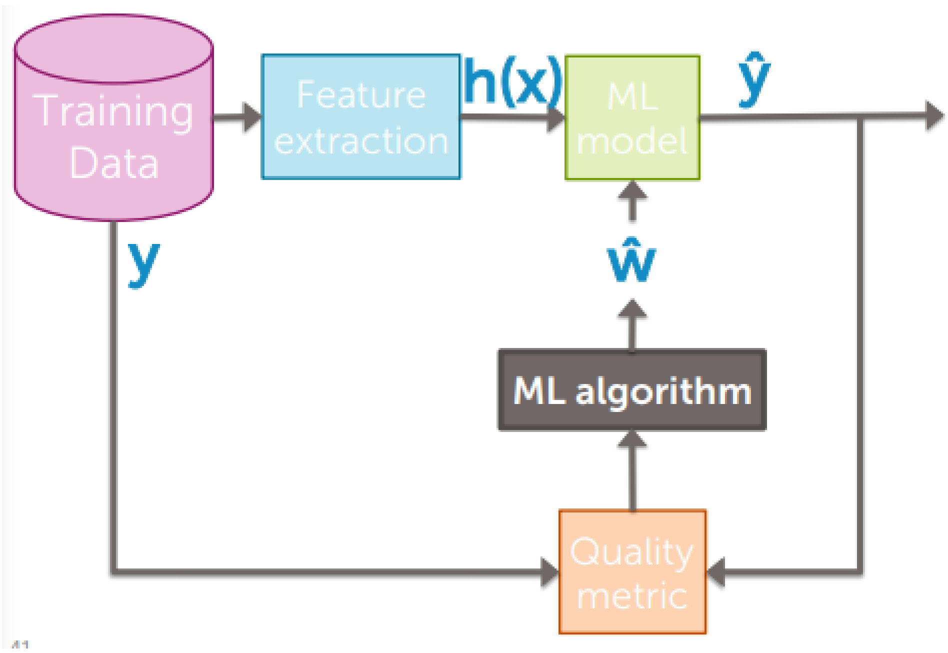
Interpreting the coefficients – Multiple linear features

$$\hat{y} = \hat{w}_0 + \hat{w}_1 \underset{\text{fix}}{\mathbf{x}[1]} + \underset{\text{fix}}{\dots} + \hat{w}_j \mathbf{x}[j] + \underset{\text{fix}}{\dots} + \hat{w}_d \underset{\text{fix}}{\mathbf{x}[d]}$$



In case of polynomial regression, since the model consist of powers of a single feature it is not possible to hold the other values still while focussing on one co-efficient.

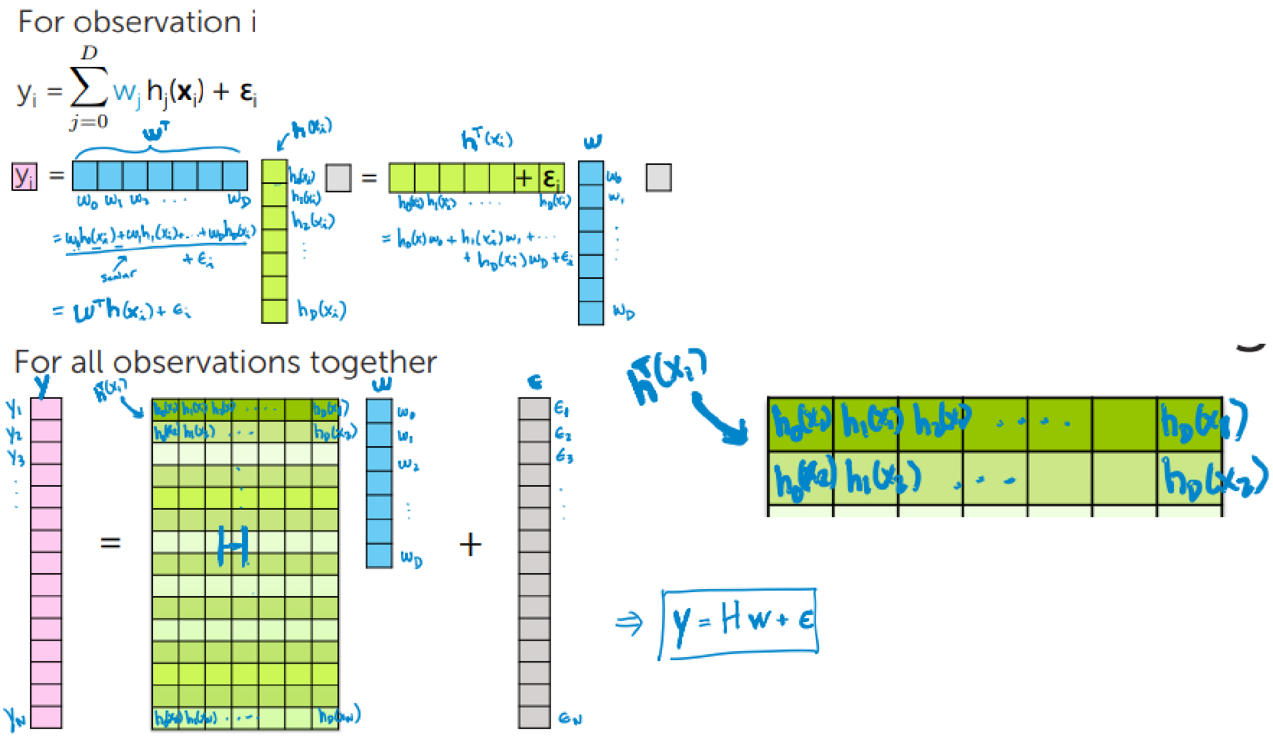
Algorithms associated with Multiple Regression:



1. Closed form solution
2. Gradient descent

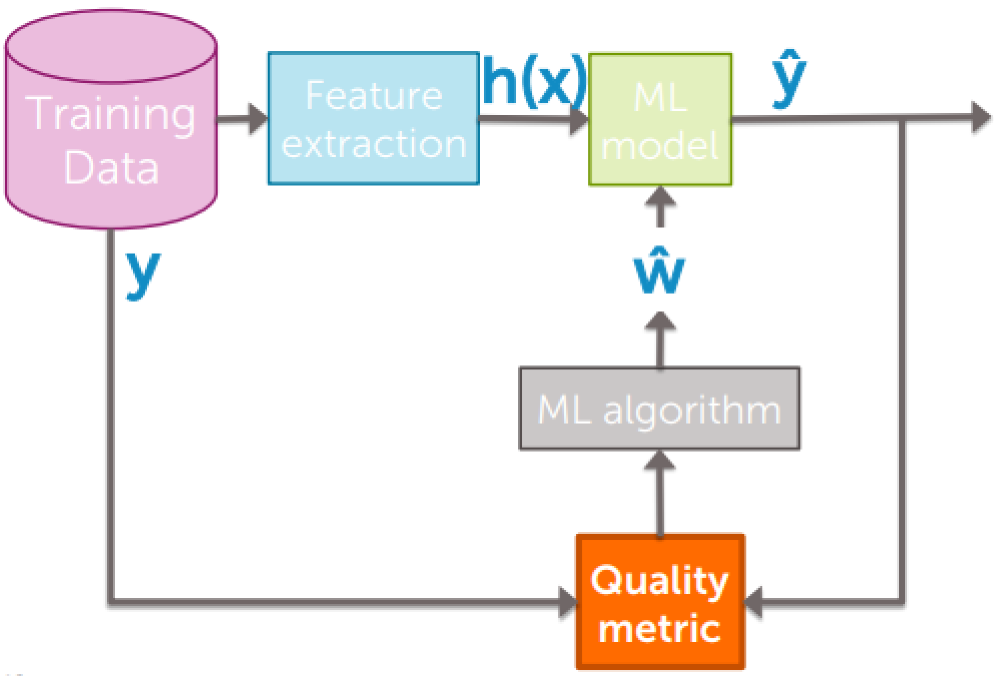
Step 1 : Rewrite in matrix notation:

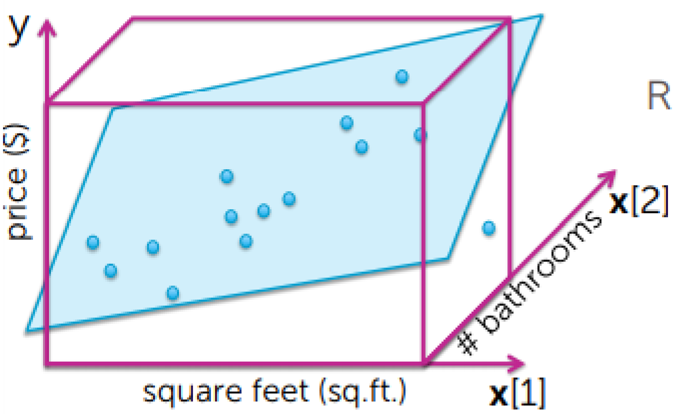
- w(j) -> parameters / co-efficients; -> vector
- h(j) -> features of input; -> vector



- The green-box - is a stack of all features per observation.

Step 2 : Compute the cost:





$$RSS(\mathbf{w}) = \sum_{i=1}^{N}\Big(y_i - \underbrace{h^T(x_i)\,\mathbf{w}}_{\hat{y}_i(\mathbf{w})}\Big)^2$$

$$\hat{y}_i = \underbrace{\begin{bmatrix} h_0(x_i) & h_1(x_i) & \cdots & h_D(x_i) \end{bmatrix}}_{h^T(x_i)} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}}_{\mathbf{w}}$$

$$RSS(\mathbf{w}) = \sum_{i=1}^{N}(y_i - h(\mathbf{x}_i)^T\mathbf{w})^2$$
$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

$$\hat{y} = Hw$$

$$(y - Hw) = (y - \hat{y}) = \begin{bmatrix} residual_1 \\ residual_2 \\ \vdots \\ residual_N \end{bmatrix}$$

$$residual_i = y_i - \hat{y}_i$$

$$\begin{bmatrix} residual_1 & residual_2 & residual_3 & \ldots & residual_N \end{bmatrix} \begin{bmatrix} residual_1 \\ residual_2 \\ residual_3 \\ \ldots \\ residual_N \end{bmatrix} = (residual_1^2 + residual_2^2 + \ldots + residual_N^2)$$
$$= \sum_{i=1}^{N} residual_i^2$$
$$\triangleq RSS(w)$$

Step 3 : Take the gradient:

Gradient of RSS

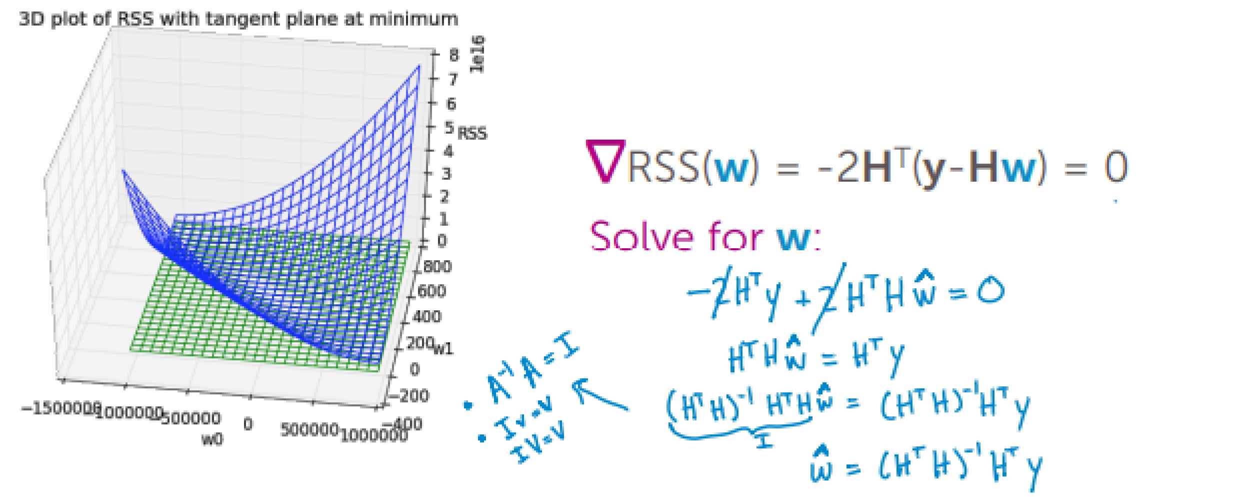
$$\nabla RSS(\mathbf{w}) = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]$$
$$= -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

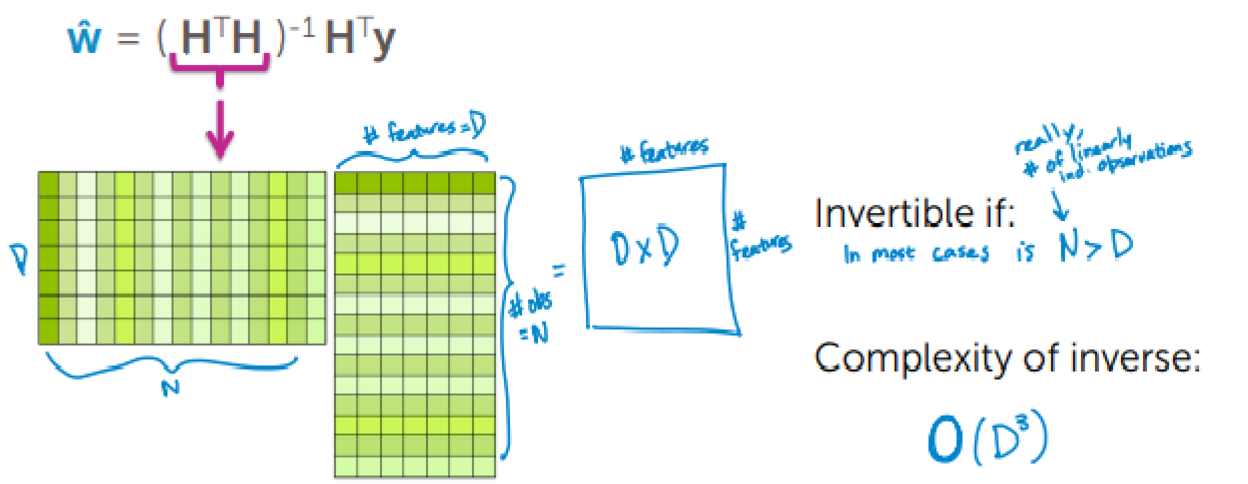
Why? By analogy to 1D case:

$$\frac{d}{dw}\underbrace{(y-hw)(y-hw)}_{\text{scalars}} = \frac{d}{dw}(y-hw)^2 = 2\cdot(y-hw)^1(-h)$$
$$= -2h(y-hw)$$

Approach - 1 : Closed form

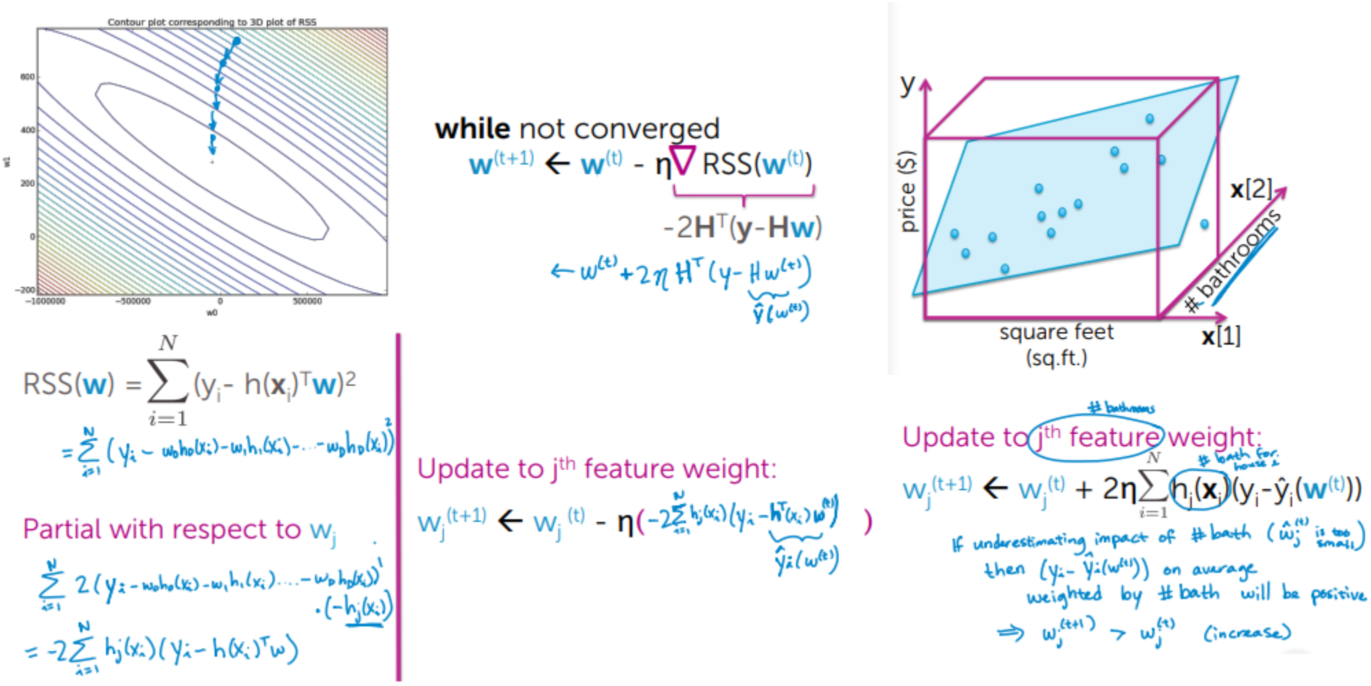
- Set the gradient = 0;





- H(T) * H -> matrix result is square matrix of dimension -> D x D;
- The above matrix D x D is invertible if the N > D;
- Where N -> number of observations; D -> number of features;

Approach - 2 : Gradient Descent



- initialize w(1) = 0 (randomly) @ t=1
- while ||∇ RSS(w(t))|| > ε

```
- for j=0,...,D
- partial[j] = -2 * Σ[i = 1<-> N] h(j) (x(i)) (y(i) - y(hat)(i)(w(t)))
- w(j)(t+1) <-- w(j)(t) - η * partial[j]
- t <-- t + 1
```

Quiz :

1 point

1. Which of the following is **NOT a linear** regression model. *Hint: remember that a linear regression model is always linear in the parameters, but may use non-linear features.*

- ( ) $y = w_0 + w_1 x$
- ( ) $y = w_0 + w_1 x^2$
- ( ) $y = w_0 + w_1 \log(x)$
- (•) $y = w_0 w_1 + \log(w_1) x$

---

1 point

2. Your estimated model for predicting house prices has a large positive weight on 'square feet living'. This implies that if we remove the feature 'square feet living' and refit the model, the new predictive performance will be **worse** than before.

- ( ) True
- (•) False

---

1 point

3. *Complete the following:* Your estimated model for predicting house prices has a positive weight on 'square feet living'. You then add 'lot size' to the model and re-estimate the feature weights. The new weight on 'square feet living' [_________] be positive.

- ( ) will not
- ( ) will definitely
- (•) might

1 point

4. If you double the value of a given feature (i.e. a specific column of the feature matrix), what happens to the least-squares estimated coefficients for every **other** feature? (assume you have no other feature that depends on the doubled feature i.e. no interaction terms).

- [ ] They double
- [ ] They halve
- [x] They stay the same
- [ ] It is impossible to tell from the information provided

---

1 point

5. Gradient descent/ascent is...

- [ ] A model for predicting a continuous variable
- [x] An algorithm for minimizing/maximizing a function
- [ ] A theoretical statistical result
- [ ] An approximation to simple linear regression
- [ ] A modeling technique in machine learning

1 point

6. Gradient descent/ascent allows us to...

- [ ] Predict a value based on a fitted function
- [ ] Estimate model parameters from data
- [x] Assess performance of a model on test data

---

1 point

7. Which of the following statements about step-size in gradient descent is/are **TRUE** (select all that apply)

- [ ] It's important to choose a very small step-size
- [ ] The step-size doesn't matter
- [x] If the step-size is too large gradient descent may not converge
- [x] If the step size is too small (but not zero) gradient descent may take a very long time to converge

---

1 point

8. Let's analyze how many computations are required to fit a multiple linear regression model *using the closed-form solution* based on a data set with 50 observations and 10 features. In the videos, we said that computing the inverse of the 10x10 matrix $H^T H$ was on the order of D^3 operations. Let's focus on forming this matrix **prior** to inversion. How many multiplications are required to form the matrix $H^T H$?

Please enter a number below.

5000

- 8th question -> Matrix multiplication between -> 2*3 x 3*2 matrix has -> 3 *2* 2 = 12 multiplications performed;
- Matrix H x H (T) = (N *D) X (D* N) => N * N;
- Matrix -> 50 observation (N), 10 features (D) =>
- H(T) *H => # multiplication => D*N x N*D => N*D^2 = 50 * 10^2 = 5000;

1 point

9. More generally, if you have D features and N observations what is the total complexity of computing $(H^T H)^{-1}$?

- ○ $O(D^3)$
- ○ $O(ND^3)$
- ◉ $O(ND^2 + D^3)$
- ○ $O(ND^2)$
- ○ $O(N^2D + D^3)$
- ○ $O(N^2D)$

- Mutlipltication complexity -> N*D^2;
- Inversion complexity -> D^3
- Total complexity = O(N*D^2 + D^3);

In [ ]:

```

```