Machine Learning
For Natural Language Processing

Abdelhak Mahmoudi
abdelhak.mahmoudi@um5.ac.ma

2020

Content

1. Introduction
2. Machine Learning

   1. Supervised Learning, 2. Unsupervised Learning
3. Natural Language Processing

   1. Regular Expressions, 2. Tokenization, 3. Character Encoding, 4. Part-of-Speech Tagging, 5. Chunking, 6. Stemming and Lemmatization, 7. Parsing, 8. Named Entity Recognition, 9. Topic Segmentation
4. Introduction to Deep Learning for NLP

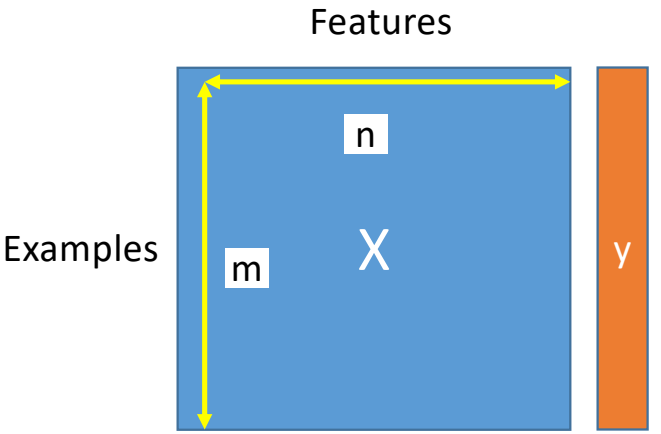
   1. Sequence models, 2. Embeddings, 3. BERT models

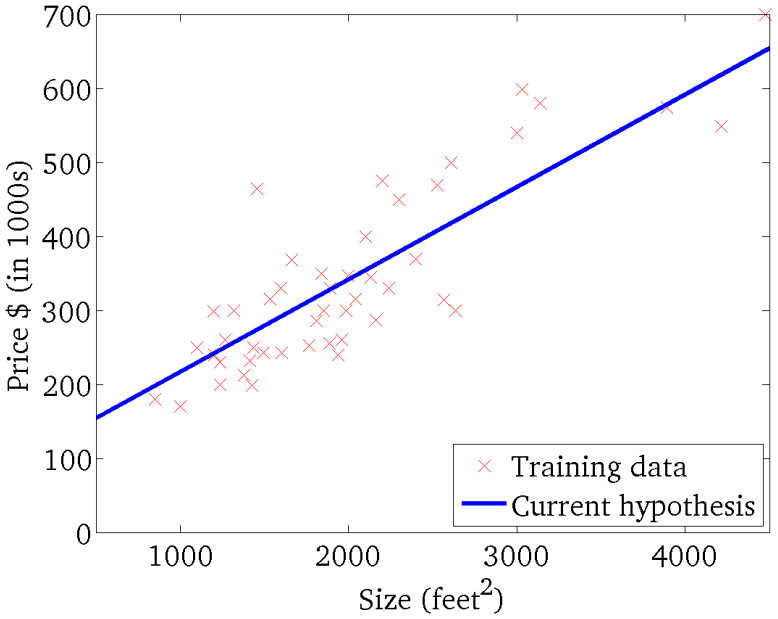
Supervised Learning

- **Linear Regression**
- Logistic Regression
- Support Vector Machines
- Trees (Decision and Regression)
- Random Forests
- Boosting
- Artificial Neural Networks

Linear Regression

- The output y is continuous
- Fit X with a line $y = w_0 + w_1 x$
- The best line is the line with minimum loss $\mathrm{L}(w)$
- Solved using Normal Equations
  - $W = (X^T X)^{-1} X^T y$
  - But not for big X !
- Find W iteratively using gradient descent





Gradient Descent

(Batch) GD

X = data_input

Y = data_output

W = initialize_parameters()

for it **in** range(num_iterations):

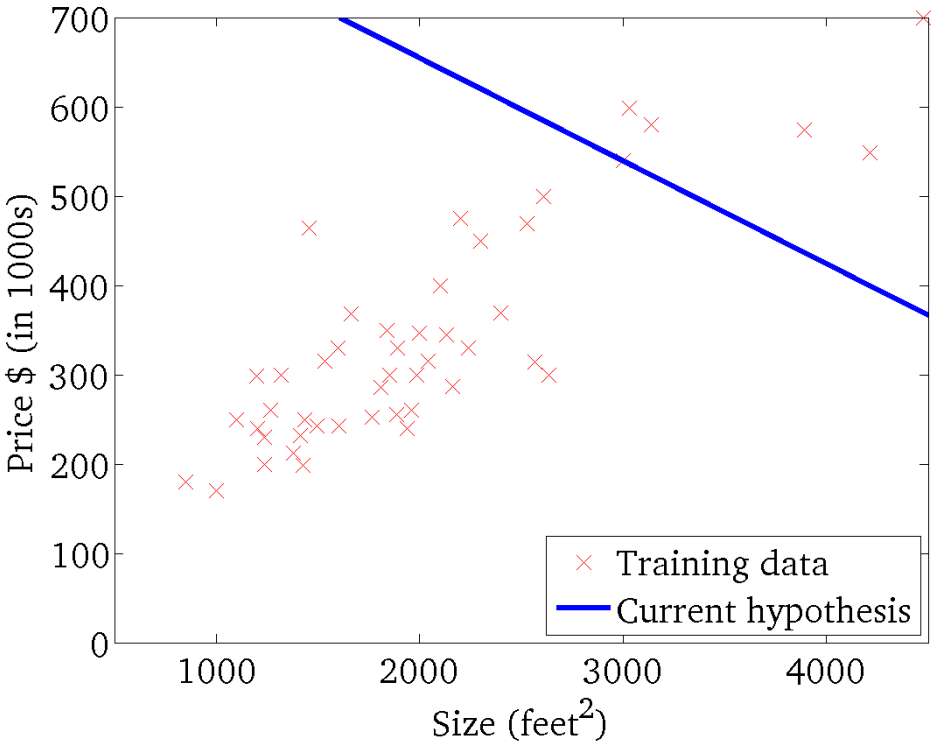
    Yhat = h(X, W)

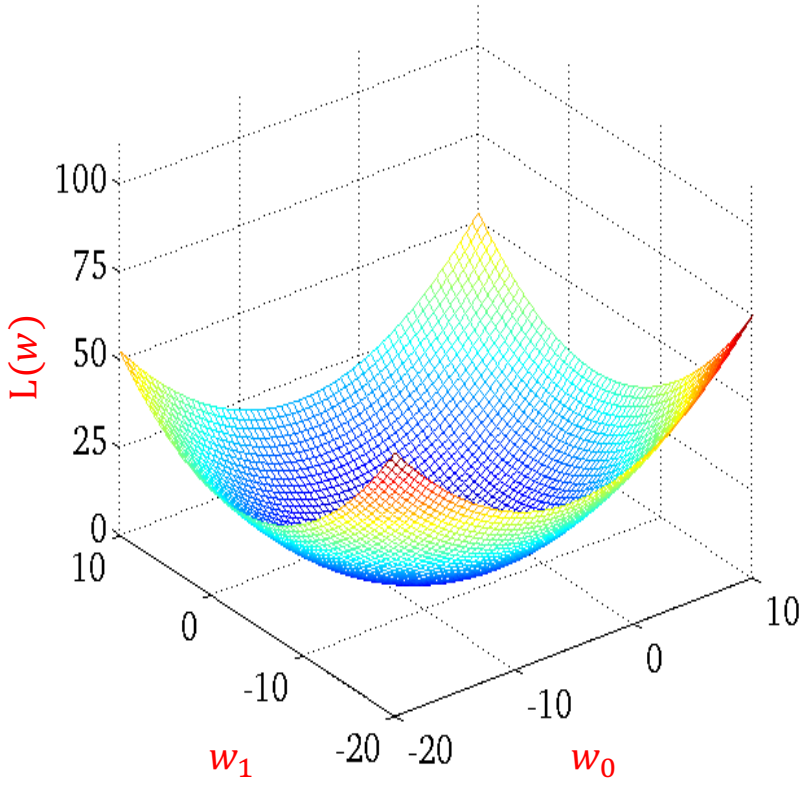
    L = loss(Yhat, Y)

    dW = gradient(L(W))

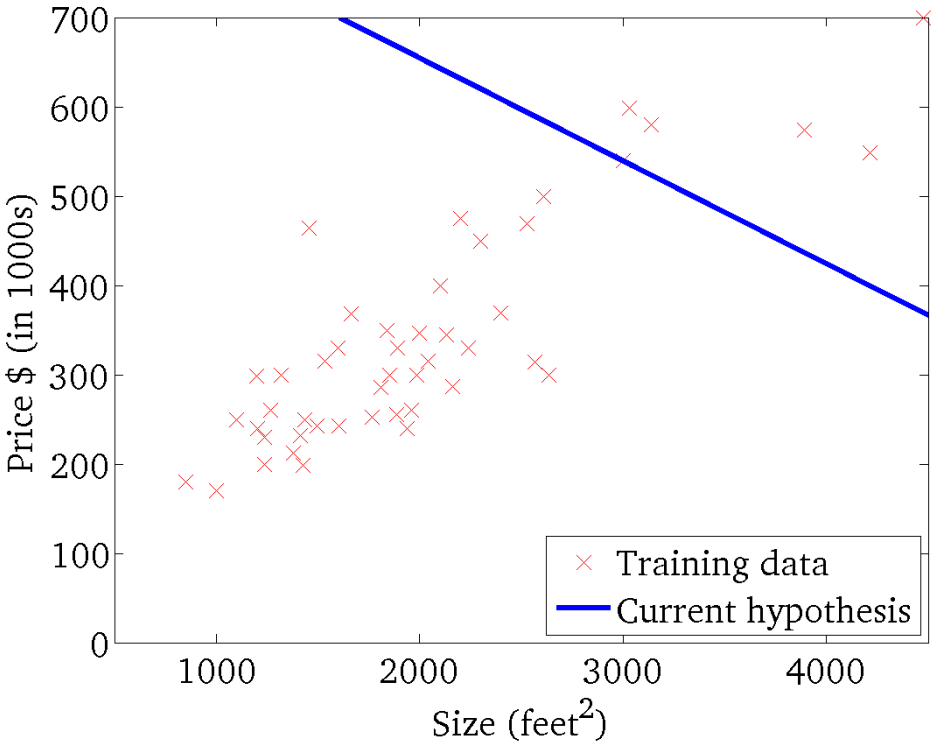
    $W = W - \alpha \, dW$

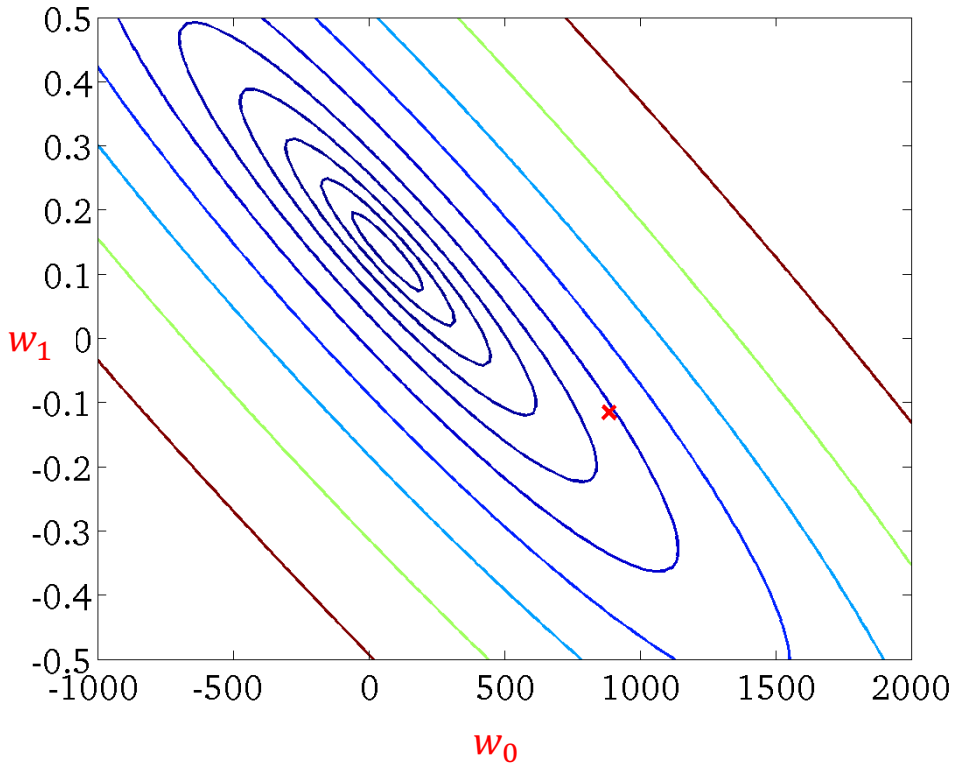
Linear Regression



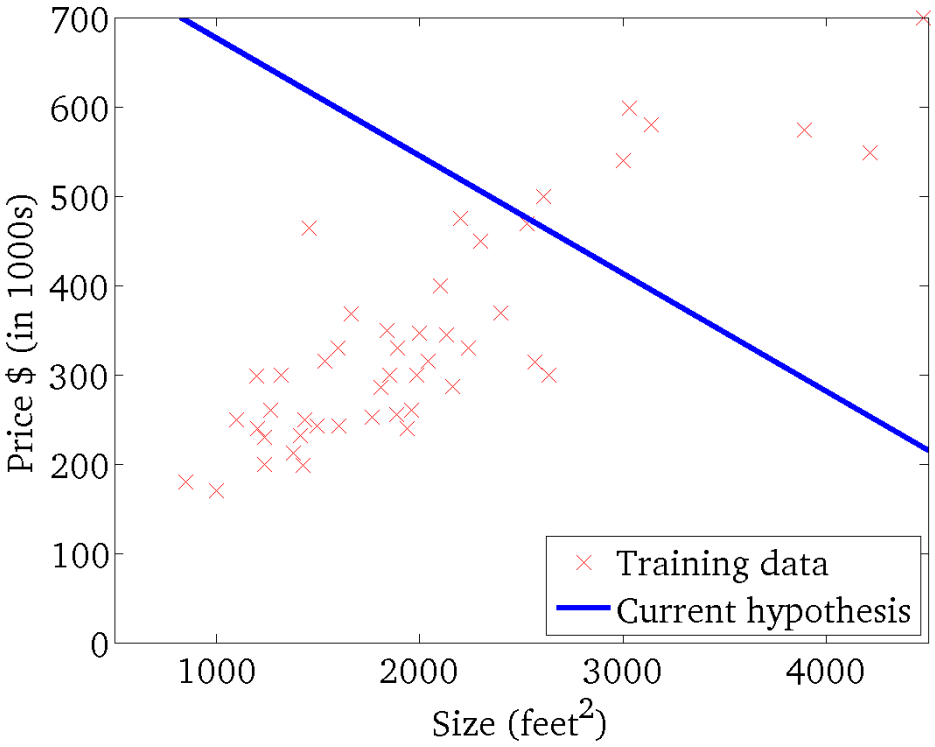


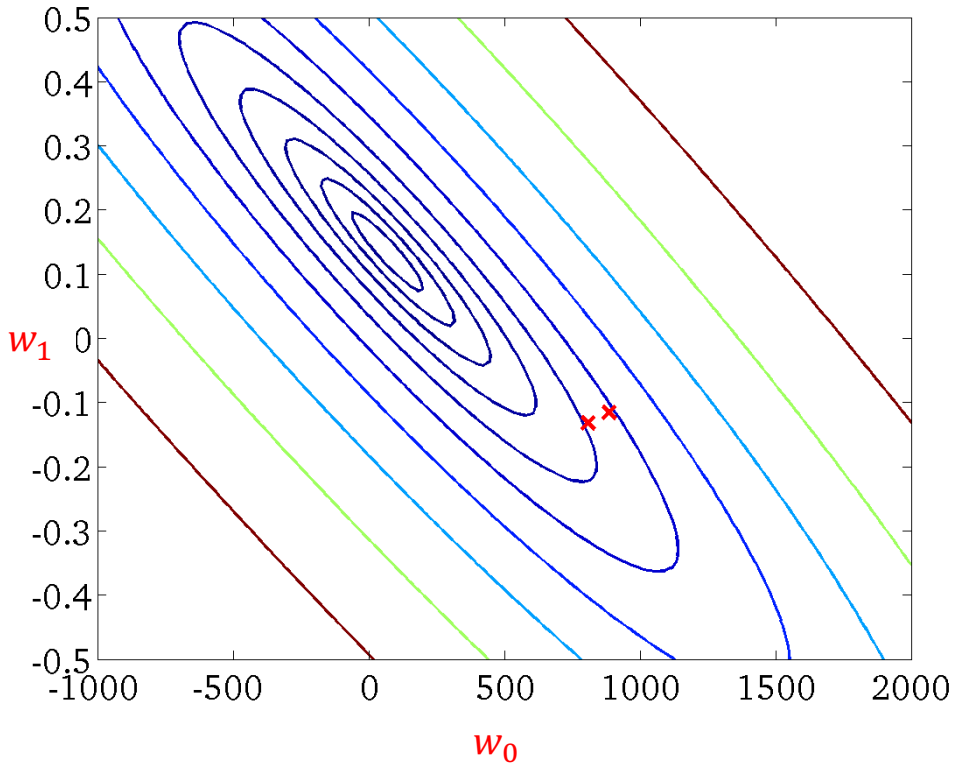
Linear Regression



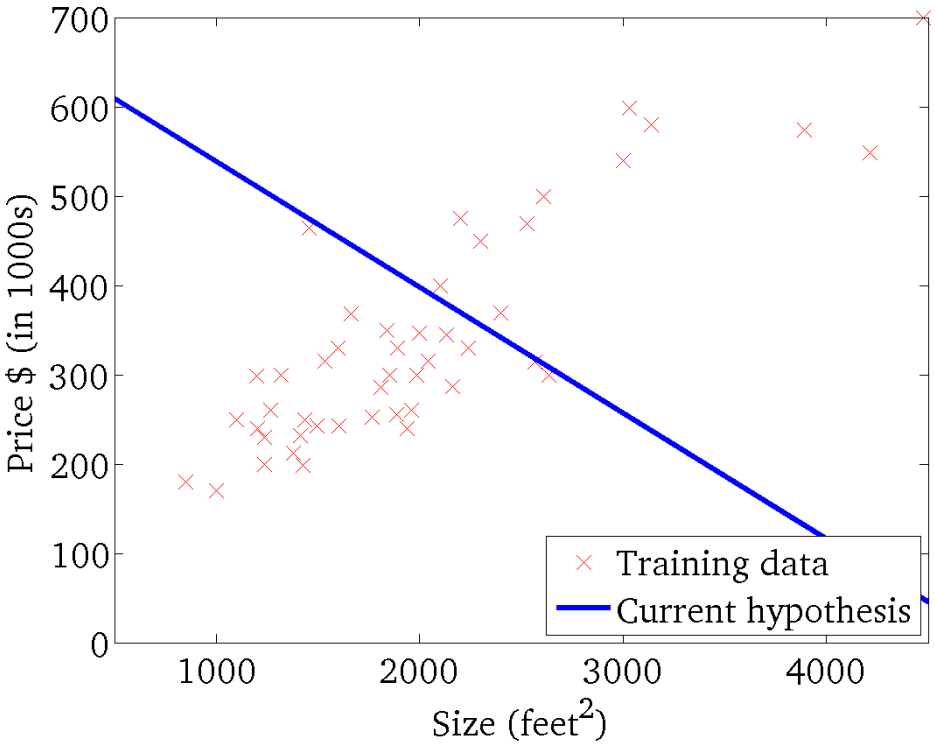


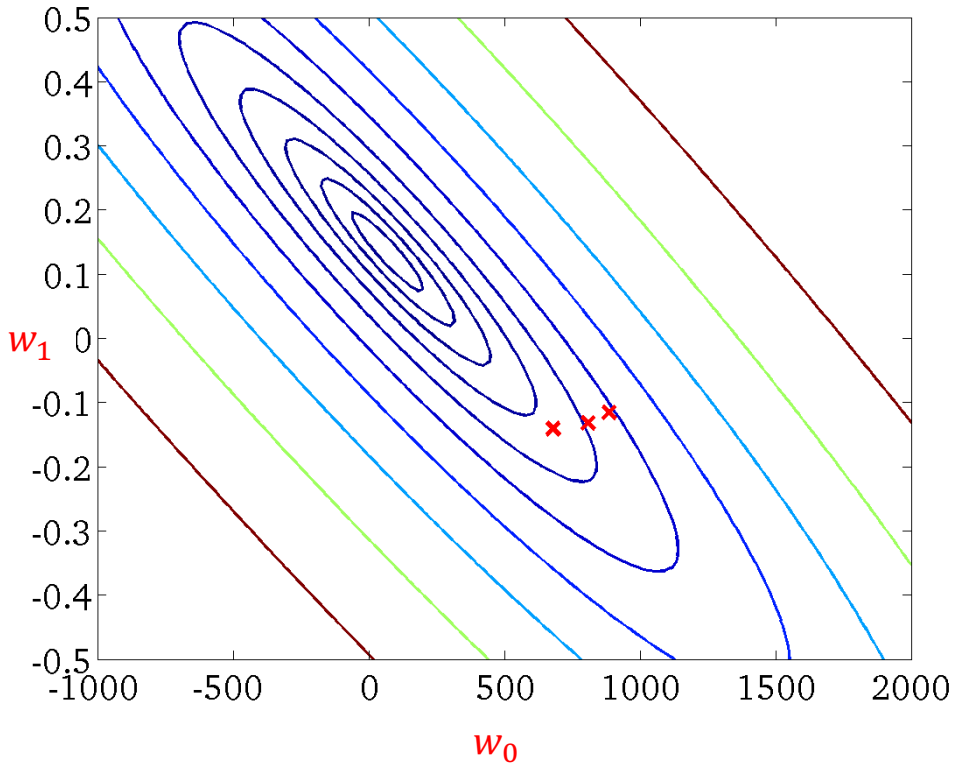
Linear Regression



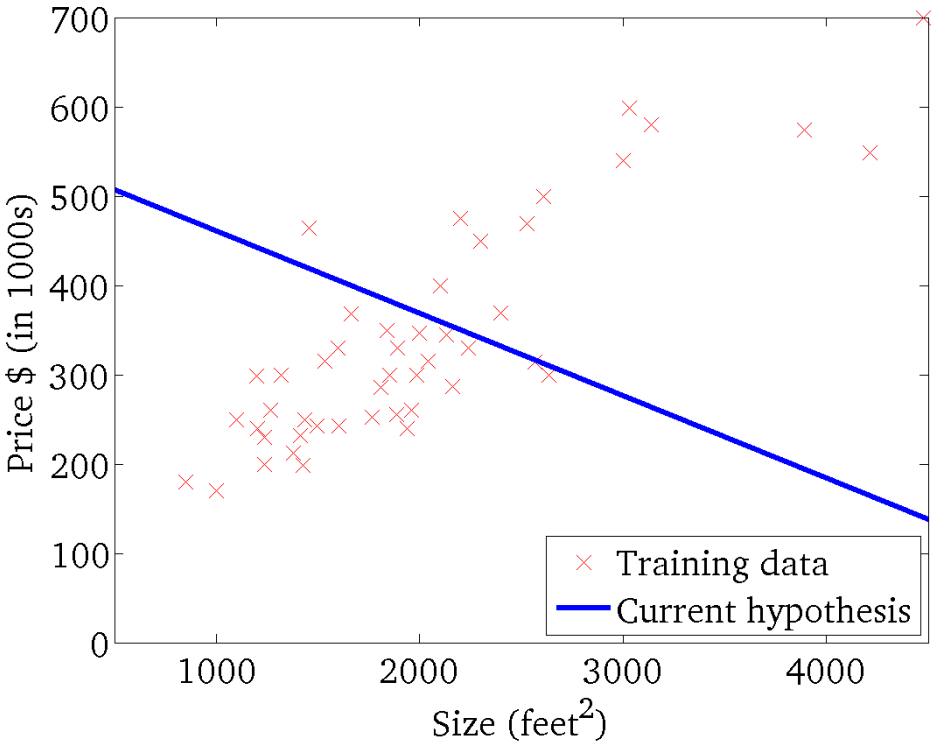


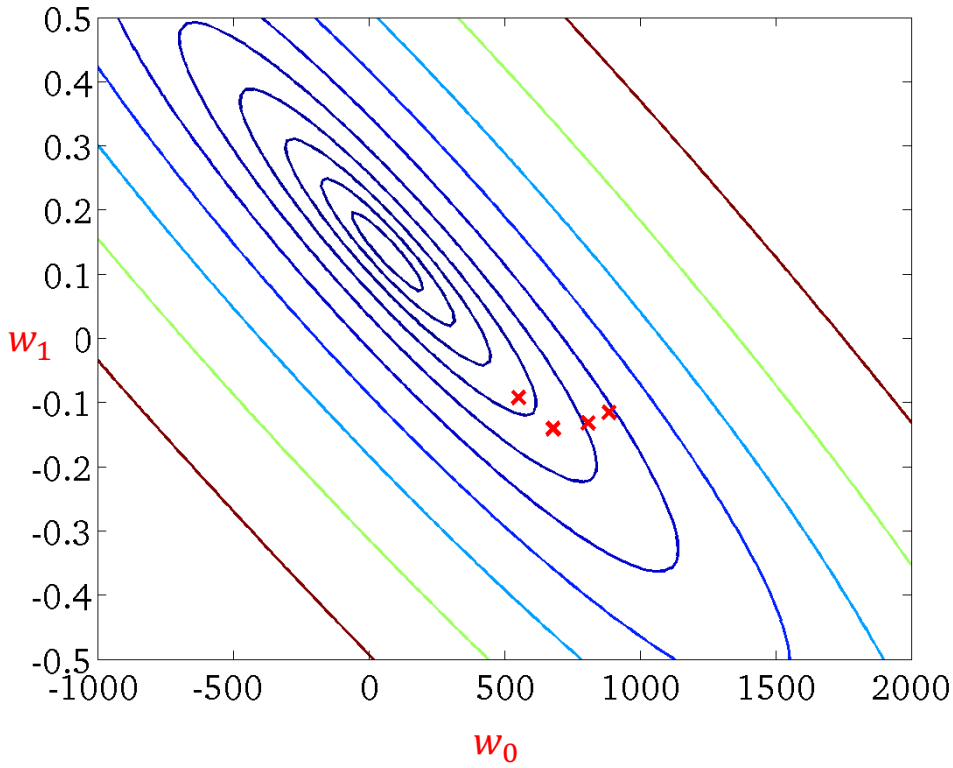
Linear Regression



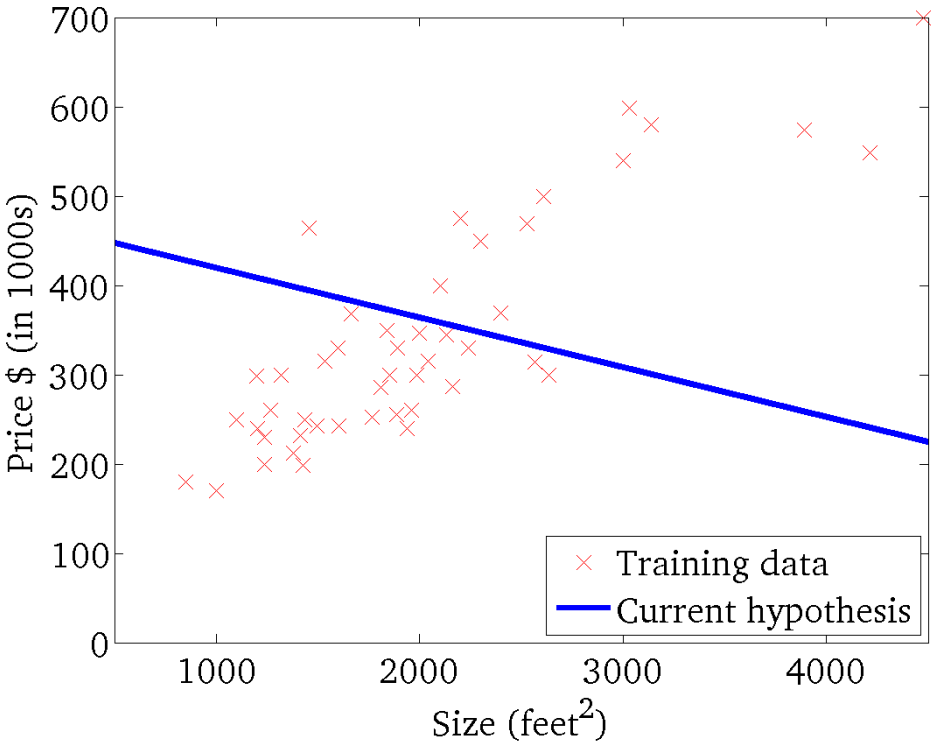


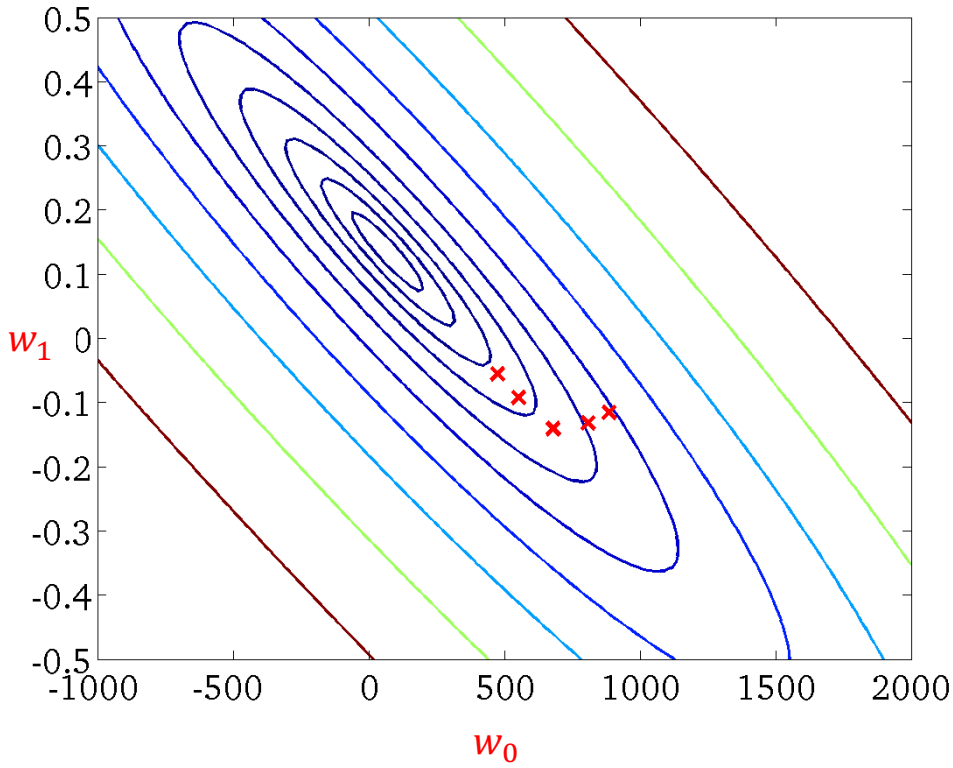
Linear Regression



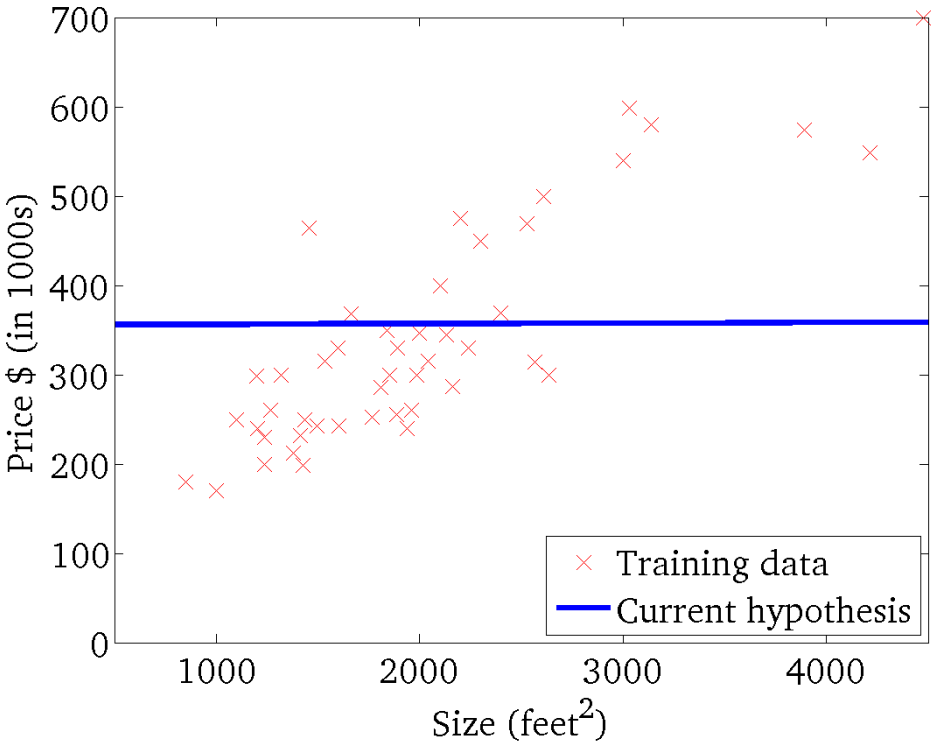


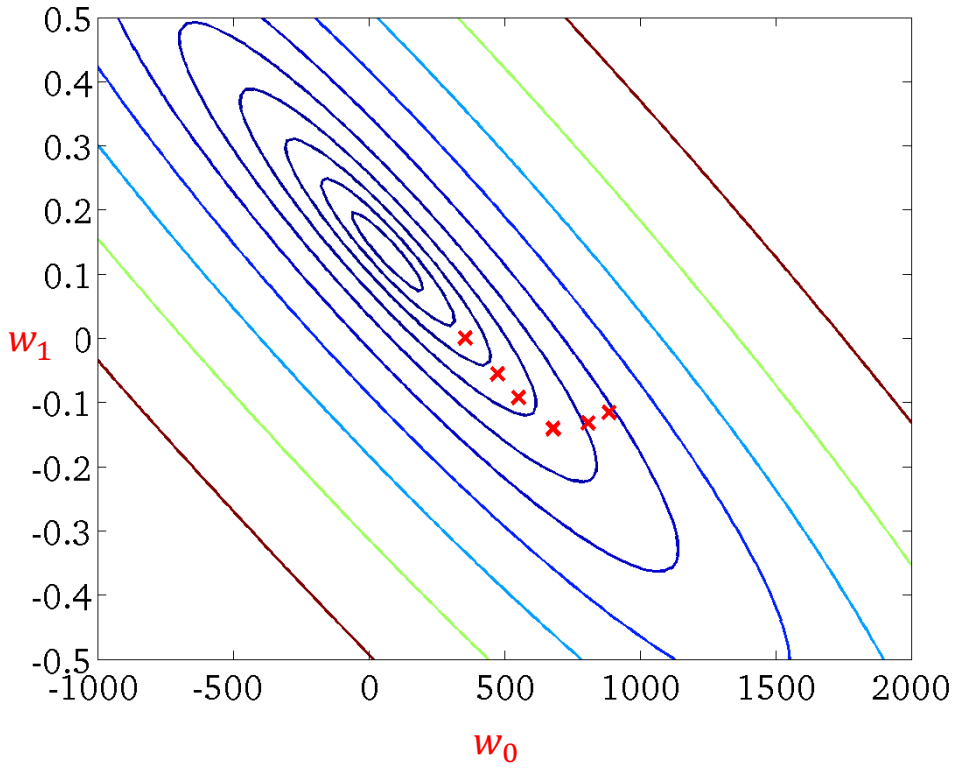
Linear Regression



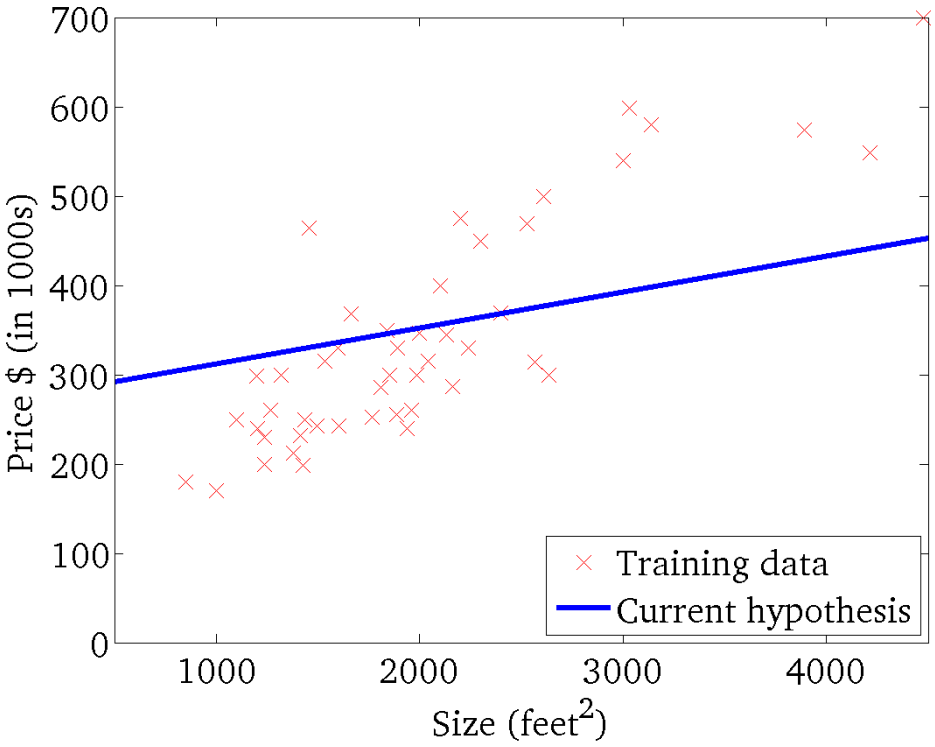


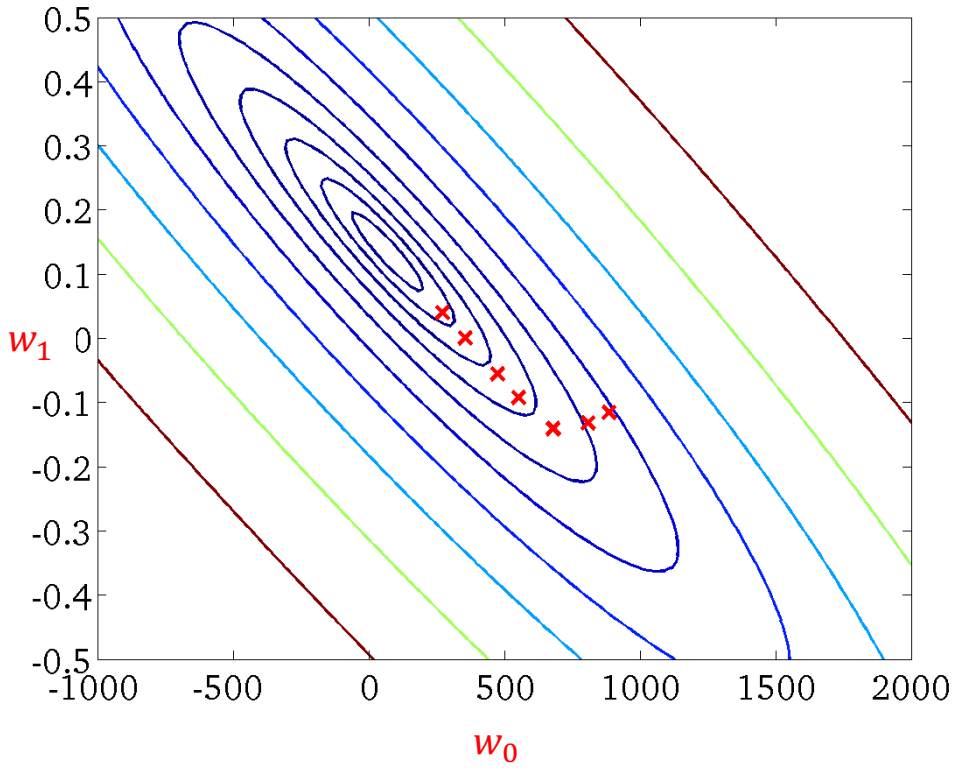
Linear Regression



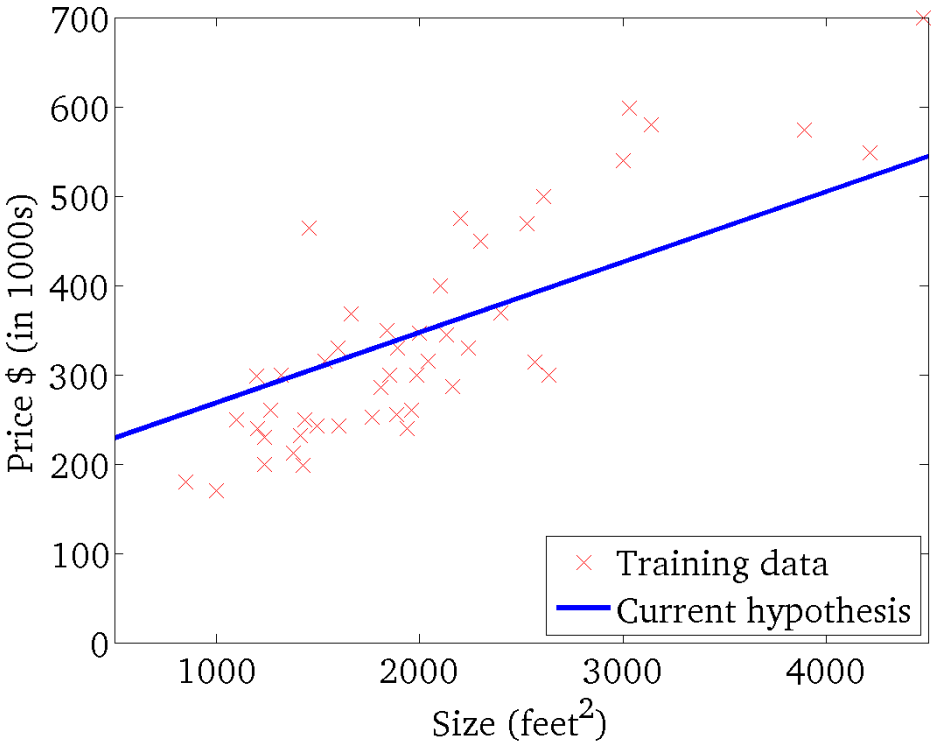


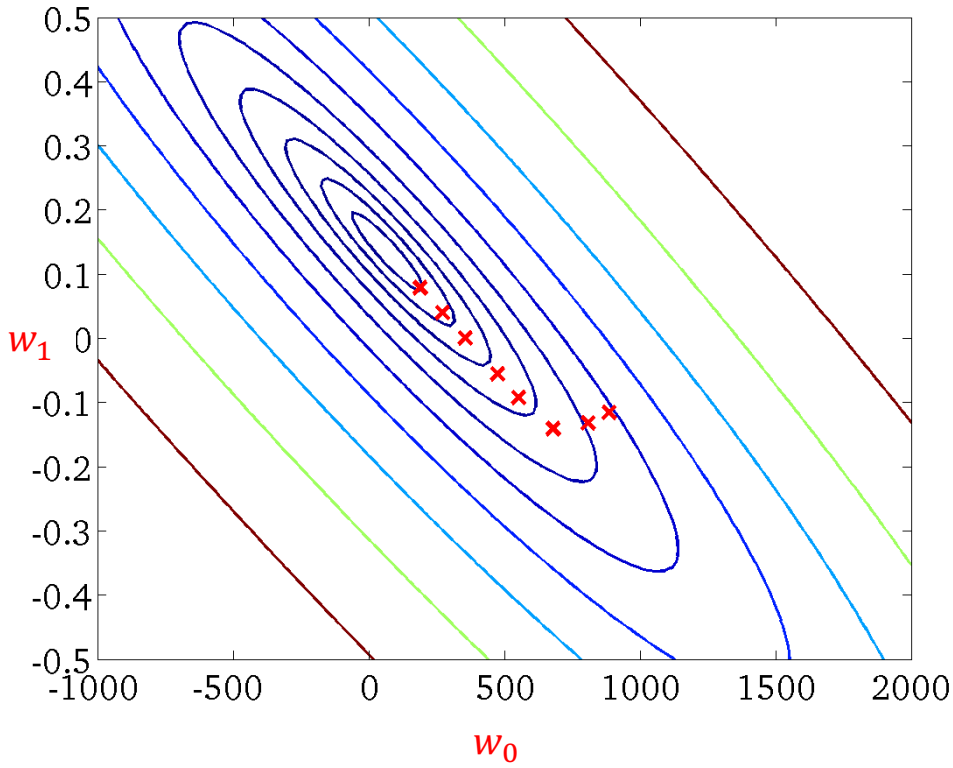
Linear Regression



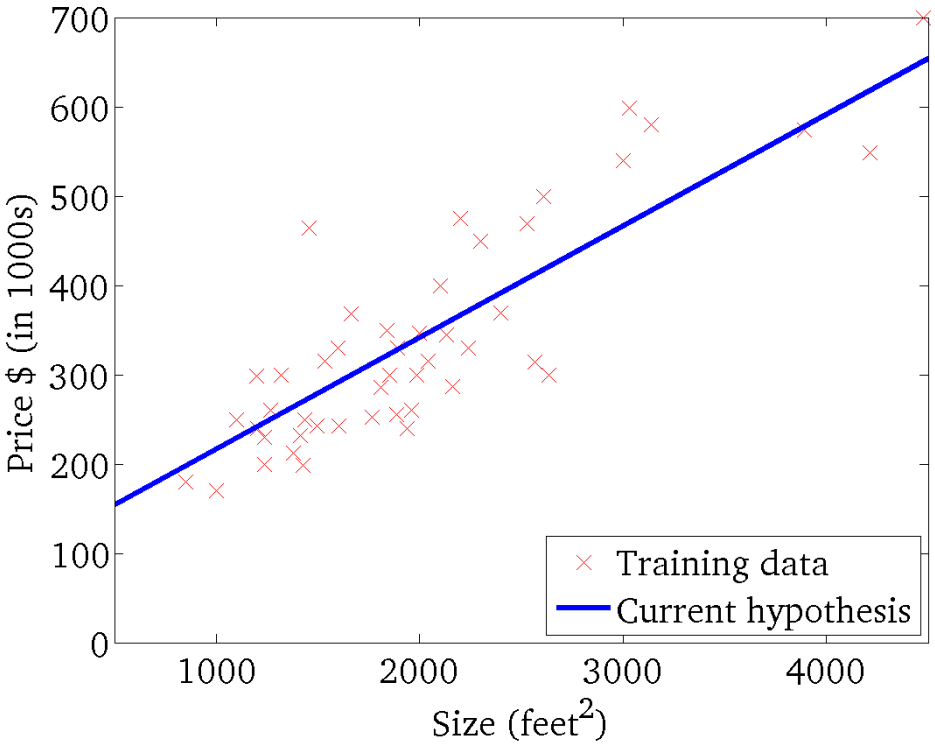


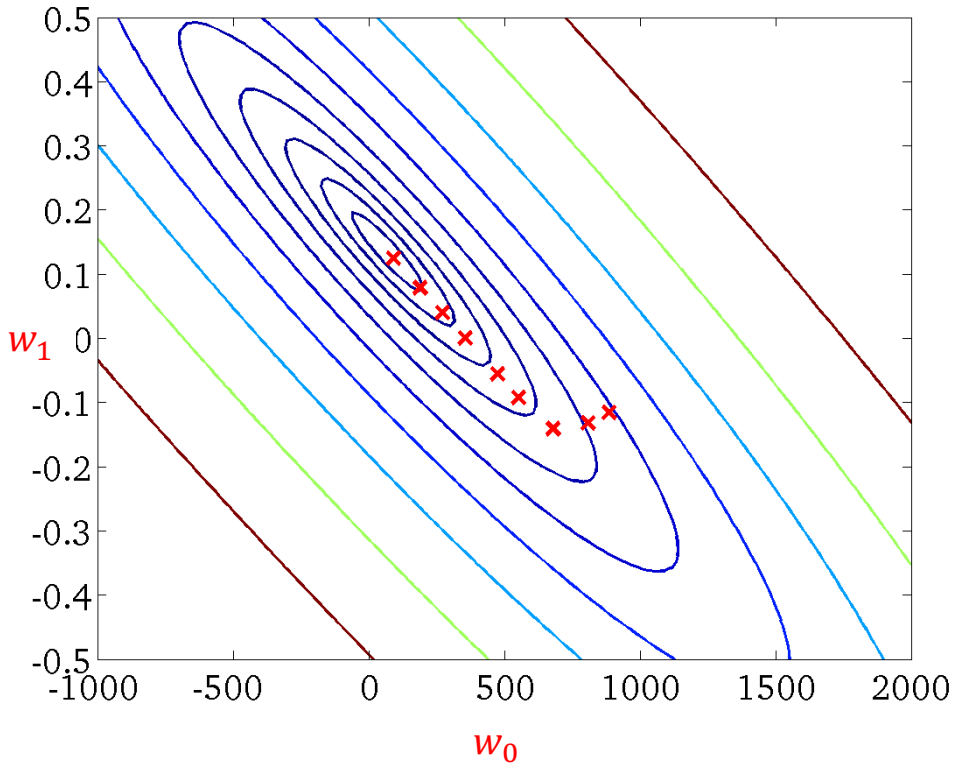
Linear Regression





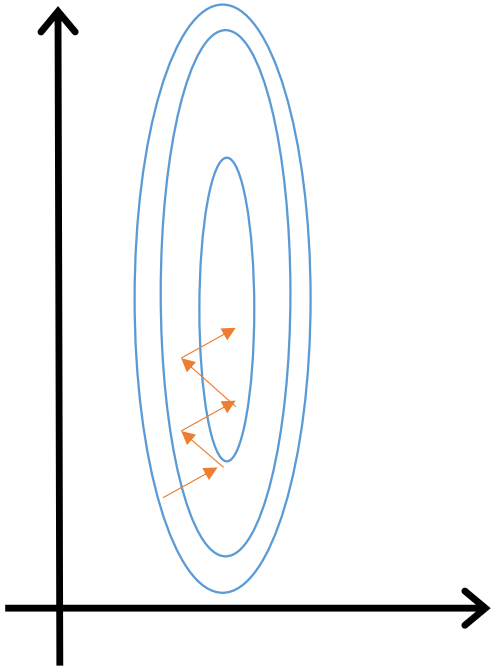
Linear Regression





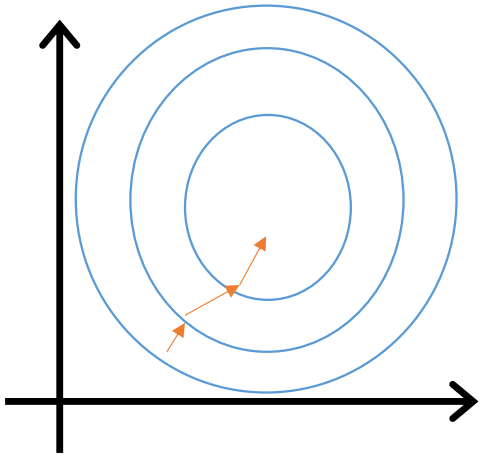
Feature Scaling

Problem: features are not on a similar scale



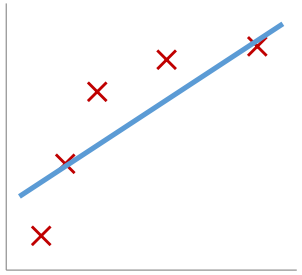
Solution: Mean Normalization

$$\frac{x_j - \mu_j}{\sigma_j} \qquad -1 \leq x_j \leq 1$$

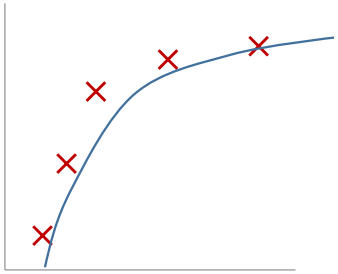


Overfitting vs. Underfitting

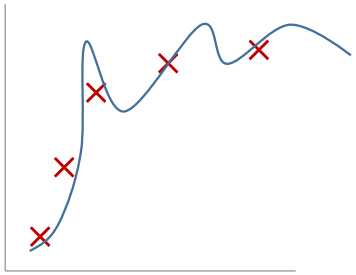
Underfitting
High Bias
Low variance



Trade-off

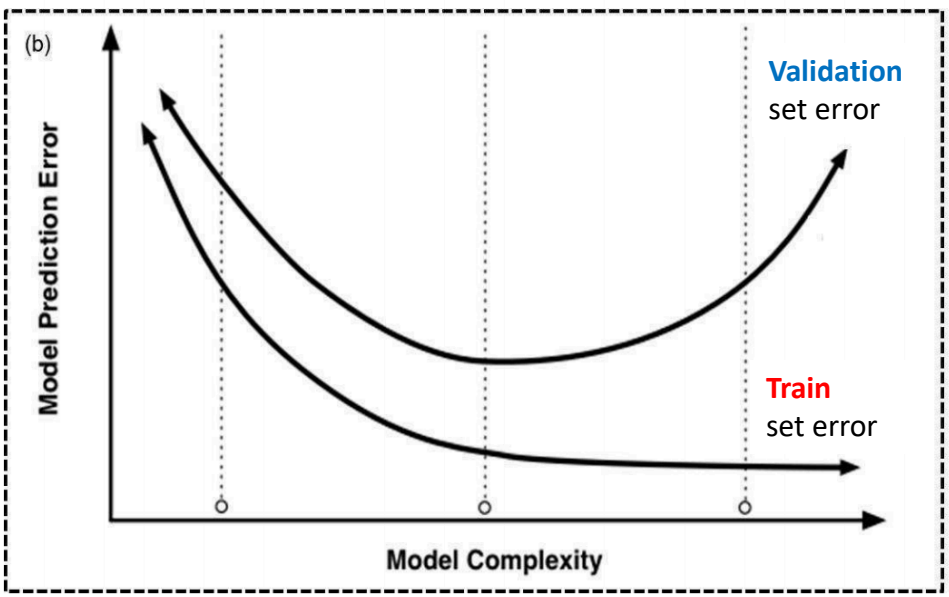


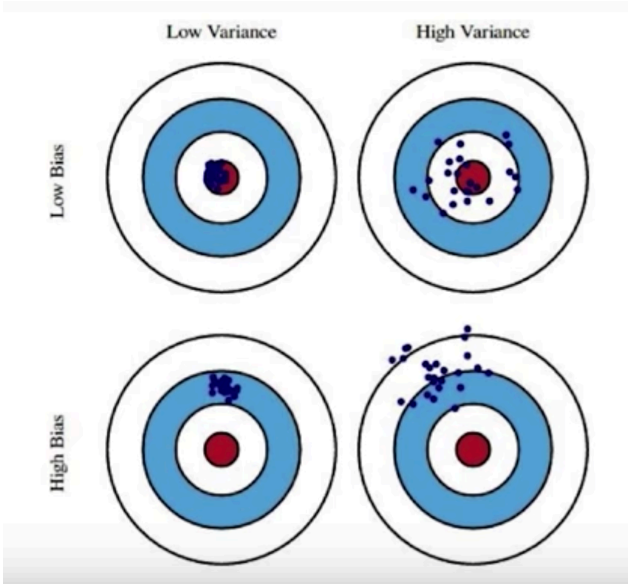
Overfitting
Low Bias
High Variance



Bias-Variance Tradeoff

Expected error (Human or Bayes optimal): **0%**	**Train** set error	1%	15%	15%	0.5%
	Validation set error	11%	16%	30%	1%
		High variance	High bias	High bias High variance	Low bias Low variance



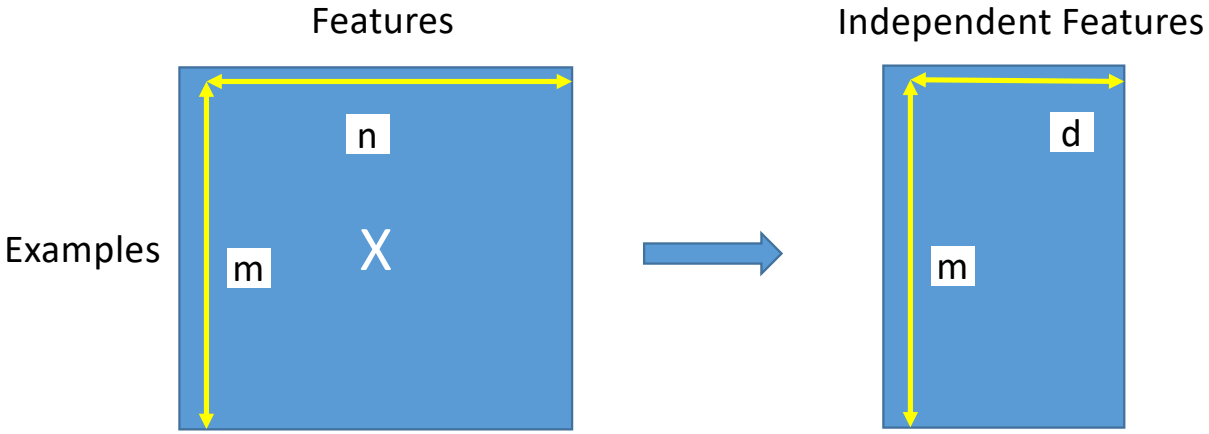


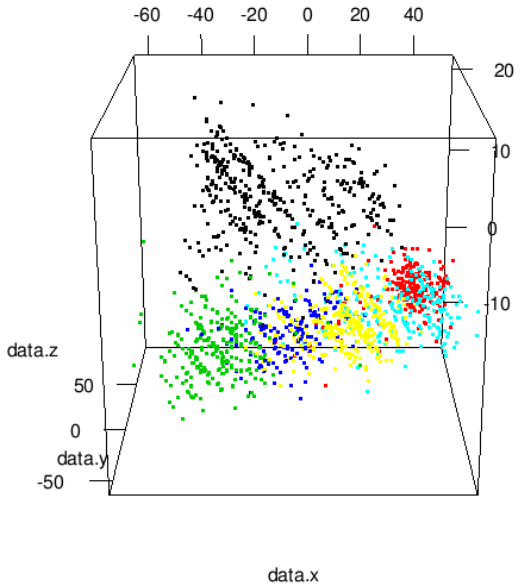
Address Overfitting

- Detect Overfitting
  - Performance analysis (Cross-Validation)
- Avoid Overfitting
  - Fewer features (Feature Selection, Dimensionality Reduction)
  - Constraint the model (Regularization : minimum loss $\mathrm{L}(w) + \lambda\, ww^T$)
  - Model Selection (Tune hyper-parameters using Grid Search)

Dimensionality Reduction

- Reducing the number of features





Regularization: Ridge Regression (L_2 norm)

Linear Regression

$$\hat{y} = h_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

$$\underset{w}{\text{minimize}} \quad L(y, \hat{y})$$

L_2 norm

$$\underset{w}{\text{minimize}} \; L(y, \hat{y}) + \lambda \sum_{j=1}^{n} w_j^2$$

Do not regularize for j=0

if λ is set to be extremely large, then w_j have to be very small.

→Algorithm results in underfitting

→Gradient Descent will fail to converge

Training $w_0 = 1, w_1 = 2, w_2 = 0.01$

Test $w_0 = 1, w_1 = 2, w_2 = 0$

Regularization: LASSO Regression (L_1 norm)

Linear Regression

$$\hat{y} = h_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

- LASSO: Least Absolute Shrinkage and Selection Operator
- LASSO is not differentiable for every value of w, but performs best feature selection

$$\underset{w}{\text{minimize}} \quad L(y, \hat{y})$$

L_1 norm

$$\underset{w}{\text{minimize}} \quad L(y, \hat{y}) + \lambda \sum_{j=1}^{n} |w_j|$$

Do not regularize for j=0

Training $\quad w_0 = 1, w_1 = 2, w_2 = 0$

Test $\quad w_0 = 1, w_1 = 2, w_2 = 0$

Model Selection

- Hyper-Parameters Tuning
  - λ : regularization hyper-parameter
  - d: degree of polynomial
  - Etc.
- Grid Search
- Randomized search

Supervised Learning

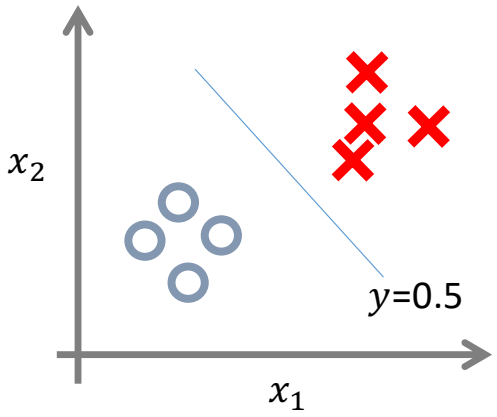
- Linear Regression
- **Logistic Regression**
- Support Vector Machines
- Trees (Decision and Regression)
- Random Forests
- Boosting
- Artificial Neural Networks

Logistic Regression

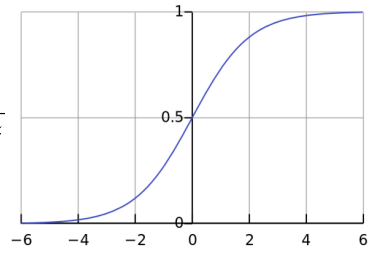
- The output y is discrete
- Classify X with a line $y = \boldsymbol{g}(w_0 + w_1 x_1 + w_2 x_2)$
- The best line is the one with minimum loss

$$\mathrm{L}(w) = \frac{1}{m}\sum_{i=1}^{m}\left[\hat{y}^{(i)}\log(y^{(i)}) + (1-\hat{y}^{(i)})\log(1-y^{(i)})\right]$$

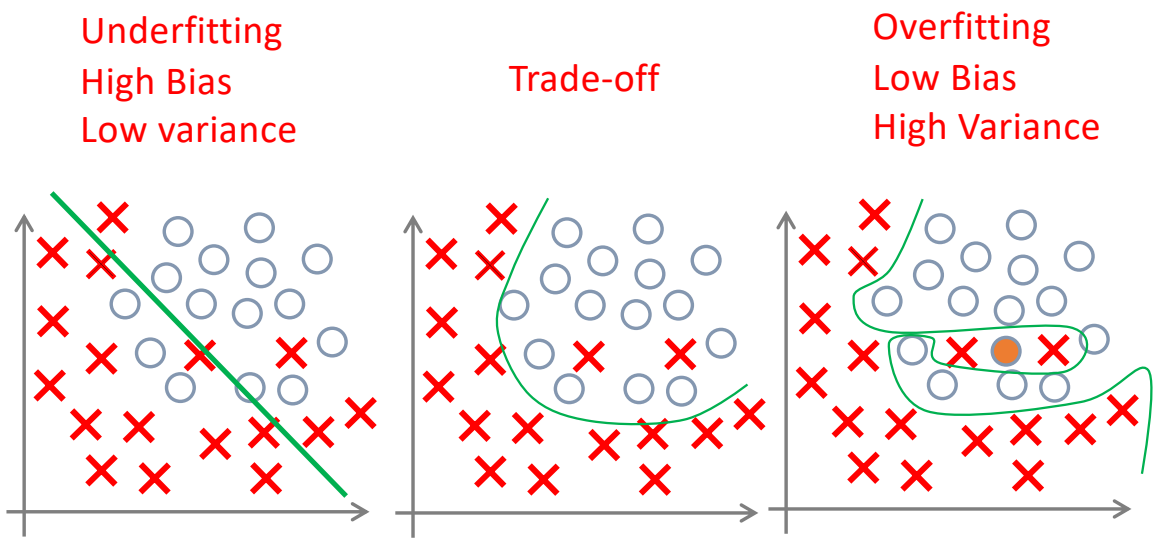
- Solved with gradient descent



$$g(z) = \frac{1}{1+e^{-z}}$$



Overfitting vs. Underfitting



Linear and Logistic Regression

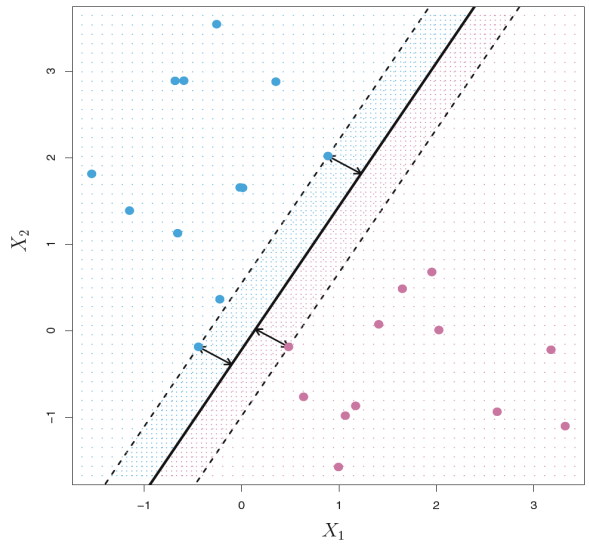
- Hyper-Parameters Tuning
  - λ : regularization hyper-parameter
  - d: degree of polynomial

Supervised Learning

- Linear Regression
- Logistic Regression
- **Support Vector Machines**
- Trees (Decision and Regression)
- Random Forests
- Boosting
- Artificial Neural Networks

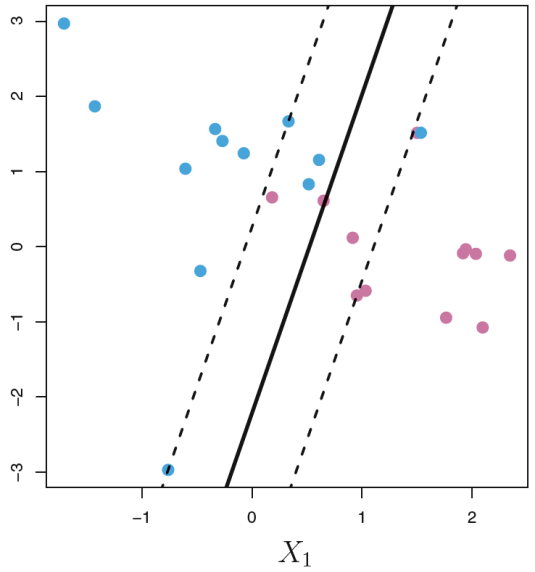
Support Vector Machines

Maximum Margin Classifier



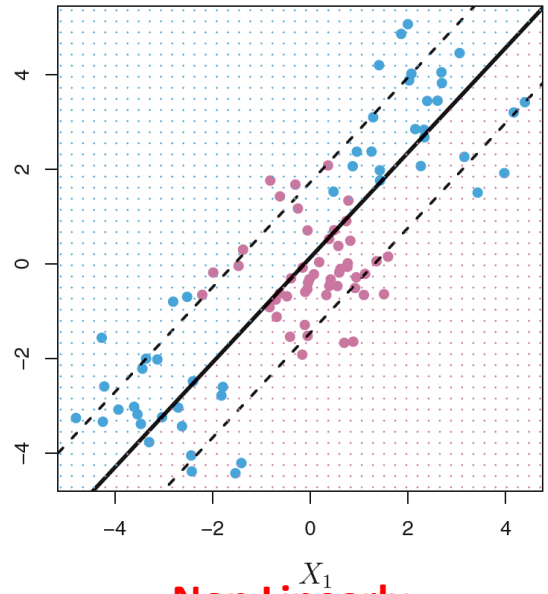
Linearly separable

Soft Margin Classifier



Slightly Linearly separable

Support Vector Machines



Non Linearly separable

Maximal Margin Classifier

- 2D: line

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

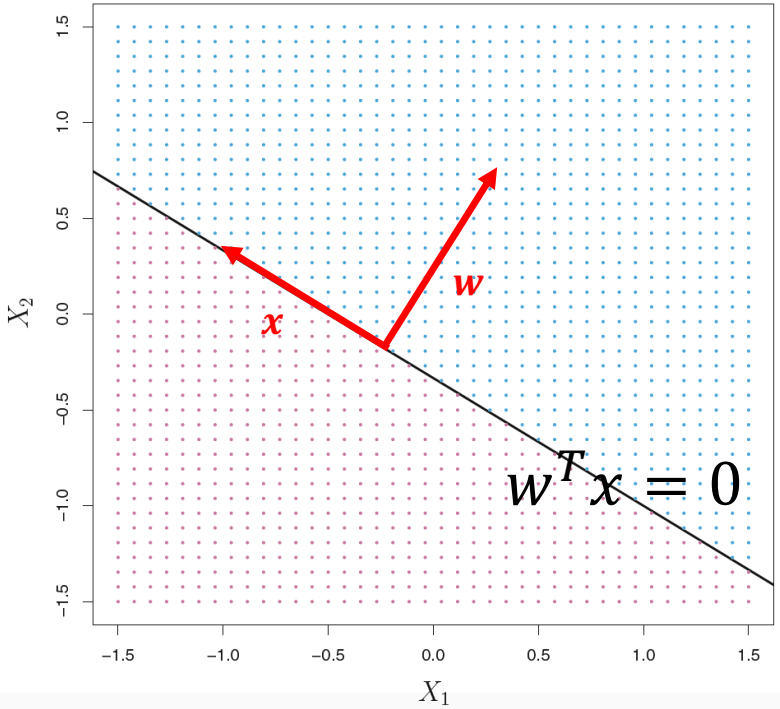
- 3D: plan

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$$

- nD: Hyperplane

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = 0$$

$$w^T x = 0$$



Maximal Margin Classifier

A separating hyperplane has the properties that:
for all i = 1, ... m.

$$w^T x^{(i)} > 0 \;\; if \;\; y^{(i)} = +1$$

$$w^T x^{(i)} < 0 \;\; if \;\; y^{(i)} = -1$$

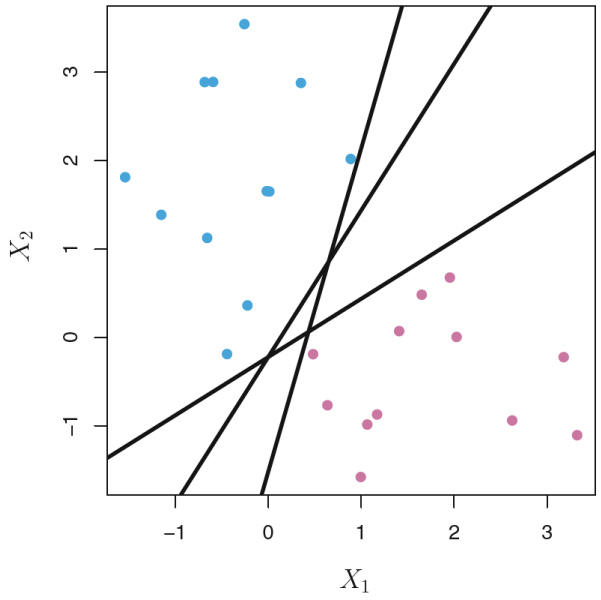
Equivalently

$$y^{(i)}(w^T x^{(i)}) > 0$$

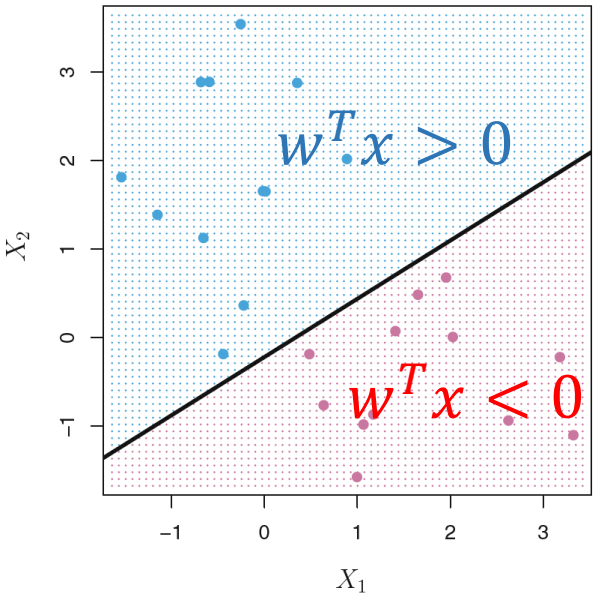
$y^{(i)}$={1,-1}

2 classes

$$w \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_n \end{bmatrix} \quad x^{(i)} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{bmatrix}$$



Many hyperplanes
Which is the best?



Maximal Margin Classifier

The optimization problem

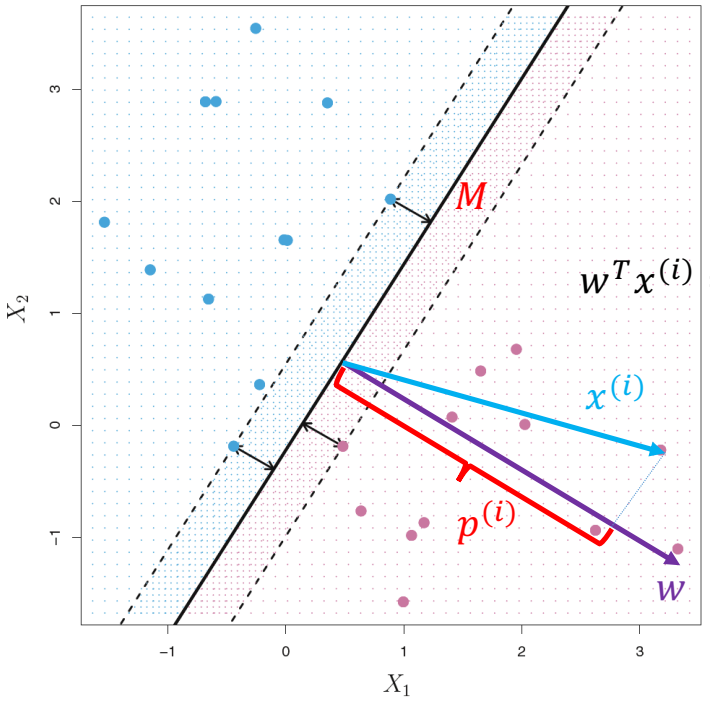
$$\underset{w}{\text{maximize}}\, M$$

Subject to: $\|w\| = \sum_{j=1}^{n} w_j^2 = 1,$

$$y^{(i)}(w^T x^{(i)}) \geq M, \forall\, i = 1 \ldots m$$

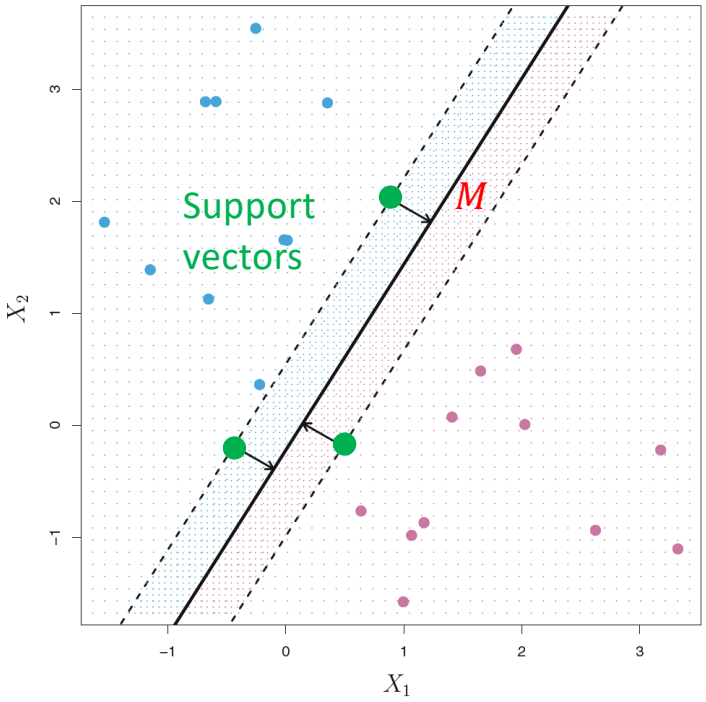
These equations ensure that each example is on the correct side of the hyperplane and at least a distance M from it.

Intuitively, pick the hyperplane with Maximum margin



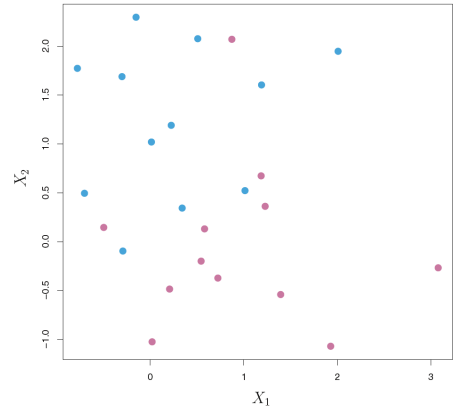
Maximal Margin Classifier

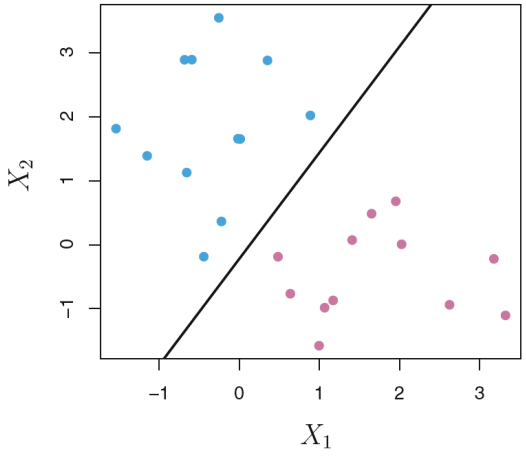
- Support vectors: examples supporting the margin (equidistant from the maximal margin hyperplane)
- If Support vectors were moved slightly, then the maximal margin hyperplane would move as well.
- The non-support vectors have no impact on the hyperplane !

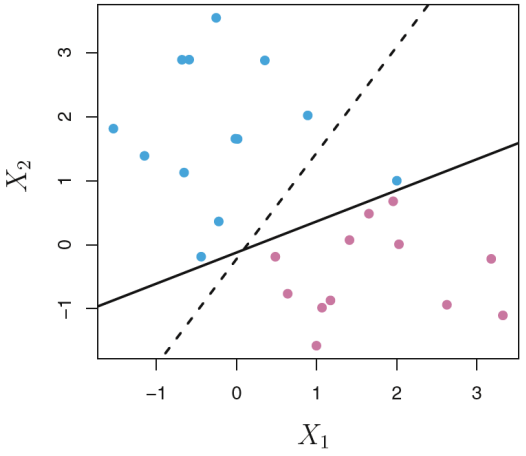


Soft Margin Classifier

- The Non-linearly separable case
- Soft margin: can be violated by some of the training examples.
- It could be better to misclassify a few training examples in order to do a better job in classifying the remaining ones.







Soft Margin Classifier

The optimization problem

$$\underset{w,\epsilon}{\text{maximize}}\, M$$

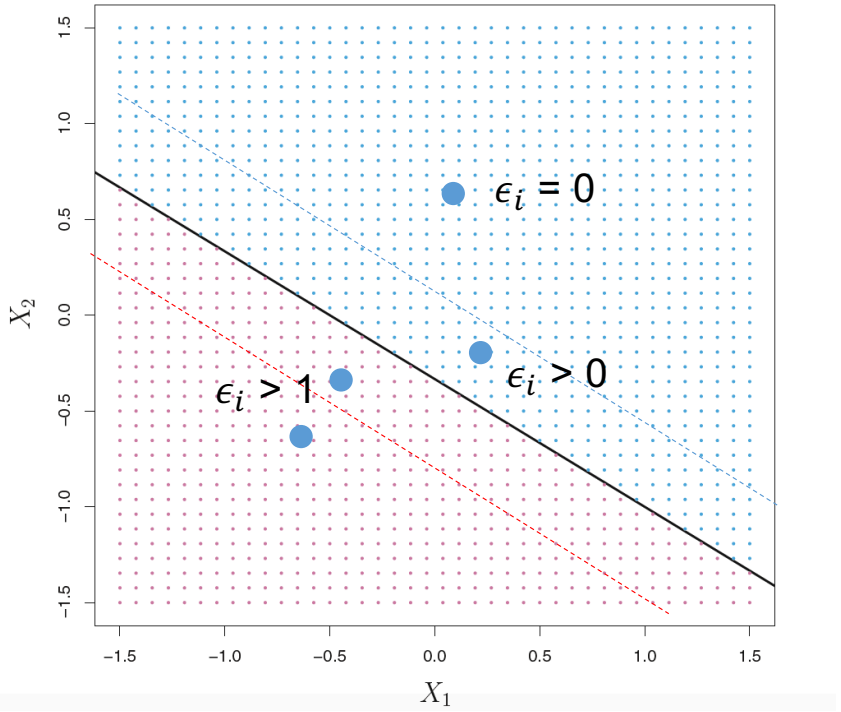
Subject to: $\|w\| = \sum_{j=1}^{n} w_j^2 = 1,$

$$y^{(i)}(w^T x^{(i)}) \geq M(1-\epsilon_i), \forall\, i = 1 \ldots m$$

$$\epsilon_i \geq 0, \quad \|\epsilon\| = \sum_{i=1}^{m} \epsilon_i^2 \leq C,$$

slack variables

Hyper parameter ≥ 0



- If ϵ_i = 0, then example i is on the correct side of the margin,
- If ϵ_i> 0, then example i is on the wrong side of the margin.
- If ϵ_i> 1, then it is on the wrong side of the hyperplane.

Soft Margin Classifier

The optimization problem

$$\underset{w,\epsilon}{\text{maximize}}\, M$$

Subject to: $\|w\| = \sum_{j=1}^{n} w_j^2 = 1,$

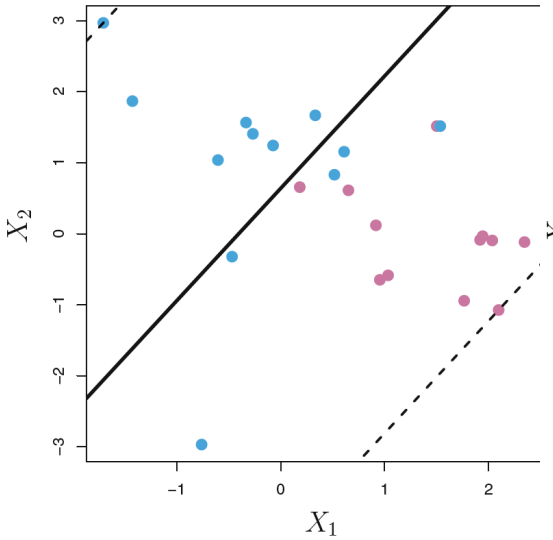
$$y^{(i)}(w^T x^{(i)}) \geq M(1-\epsilon_i), \forall\, i = 1 \ldots m$$

$$\epsilon_i \geq 0, \quad \|\epsilon\| = \sum_{i=1}^{m} \epsilon_i^2 \leq C,$$

slack variables

Hyper parameter ≥ 0

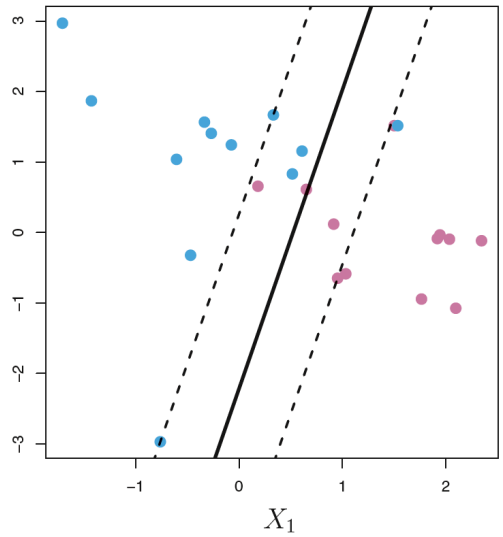
C is large



High tolerance for examples being on the wrong side of the margin ($\epsilon_i > 0$)

Underfitting:
(high bias, low variance)

C is small



Low tolerance for examples being on the wrong side of the margin ($\epsilon_i > 0$)

Overfitting:
(low bias, high variance)

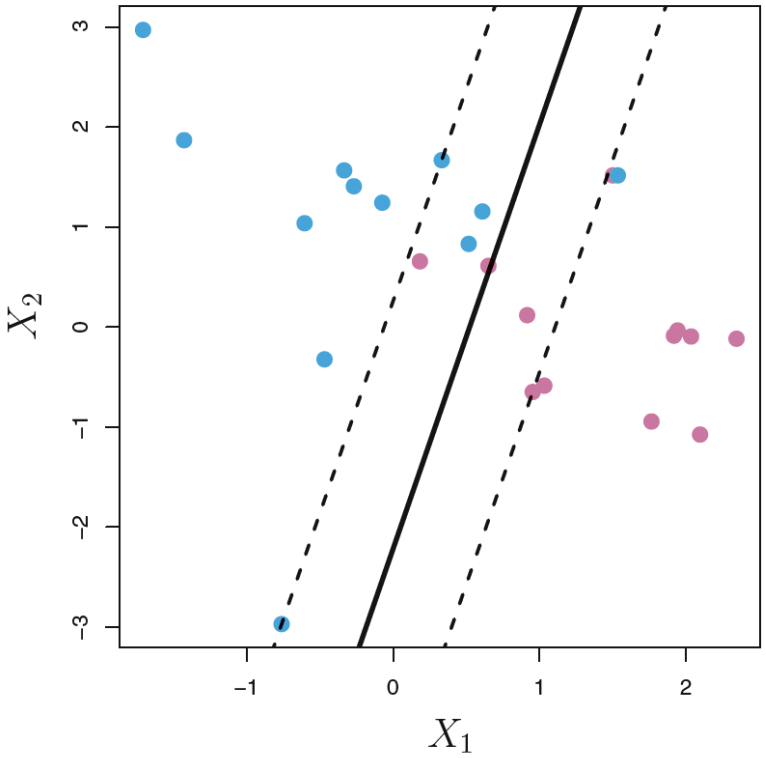
Soft Margin Classifier

- It turns out that, using quadratic programming, the solution is

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

and $w_0 = y^{(k)} - w^T x^{(k)}$ for any k where $C > \alpha_i > 0$

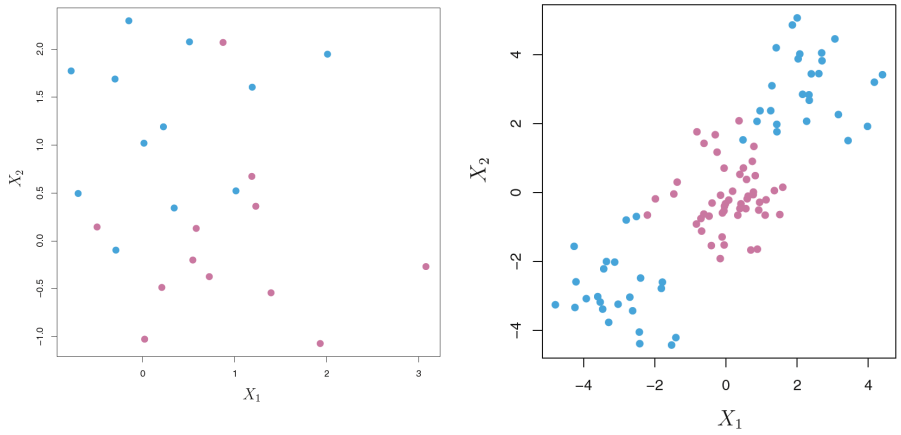
- α_i are Lagrange multipliers!
- Then, for a new $x^{(i)}$, $\hat{y}^{(i)} = sign(w^T x^{(i)})$
- $x^{(i)}$ where $\alpha_i > 0$ are called Support Vectors
- They are examples that lie directly on the margin, or on the wrong side of the margin for their class.
- Only those examples can affect the hyperplane, and hence the support vector classifier f.



Support Vector Machines

- Highly non-linearly separable case
- Use feature mapping $\varphi(x)$ to address this non-linearity.
- Example: high order polynomials

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} -> \varphi(x) = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}$$



N is the number of features in the new space

Support Vector Machines

The optimization problem

$$\underset{w,\epsilon}{\text{maximize}}\, M$$

Subject to: $\|w\| = \sum_{j=1}^{N} w_j^2 = 1,$

$$y^{(i)}(w^T \varphi(x^{(i)})) \geq M(1-\epsilon_i),$$
$$\forall\, i = 1 \ldots m$$

$$\epsilon_i \geq 0, \quad \|\epsilon\| = \sum_{i=1}^{m} \epsilon_i^2 \leq C,$$

If S is the set of support vectors, then:

$$f(x) = w_0 + \sum_{i \epsilon S} \alpha_i \varphi(x)^T \varphi(x^{(i)})$$

N could be very large → the computations would become unmanageable!

→ Use Kernel Trick

Support Vector Machines

- Non linearly separable data become separable in higher space!
- So, first go to higher feature space $x -> \varphi(x)$
- To solve SVM, you have to compute the Kernel $K(u, v) = \varphi(u)^T \varphi(v)$
  - But: very costly !!!
- Kernel Trick: If you chose φ carefully, you end up getting K, without calculating the very costly dot product $\varphi(u)^T \varphi(v)$
- The solution: $w = \sum_{i=1}^{m} \alpha_i y^{(i)} \varphi\left(x^{(i)}\right)$

and $w_0 = y^{(k)} - w^T \varphi\left(x^{(k)}\right)$ for any k where $C > \alpha_k > 0$

- Instead, compute: $w\ \varphi(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} K\left(x, x^{(i)}\right)$

Support Vector Machines

- Exemple
  - Assume each example $x = [x_1, x_2]^T$ is mapped to the quadratic feature space $\varphi(x) = \left[x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right]^T$
  - We can then show that $K(x, x') = \varphi(x)^T\varphi(x') = (1 + x^Tx')^2$
  - In this way, the computation in the higher dimensional space is performed implicitly in the original input space !

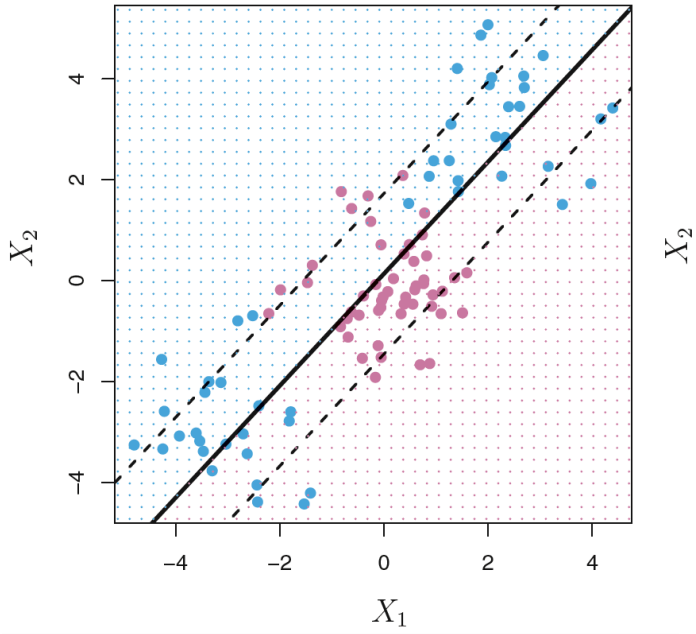
Support Vector Machines

- Kernel Examples
  - Linear Kernel $K(u, v) = u^T v$,
  - Polynomial Kernel: $K(u, v) = (c + u^T v)^d$,
  - Radial Basis Function (RBF) Kernel (Gaussian Kernel) :

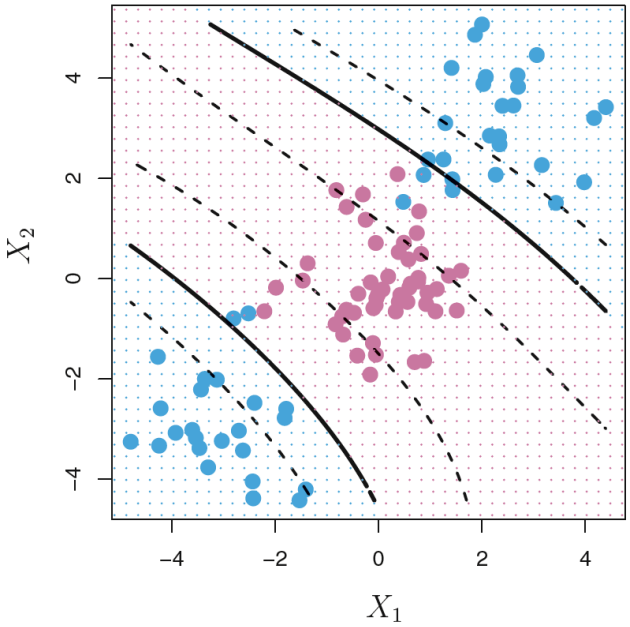
    $$K(u, v) = \exp(-\gamma \|u - v\|^2),$$ (infinite feature space!)
  - And many others: Sigmoid Kernel, String kernel, chi-square kernel, histogram intersection kernel, etc.

    (d, c and γ are hyper-parameters)
- Kernels need to satisfy technical conditions called "Mercer's conditions"

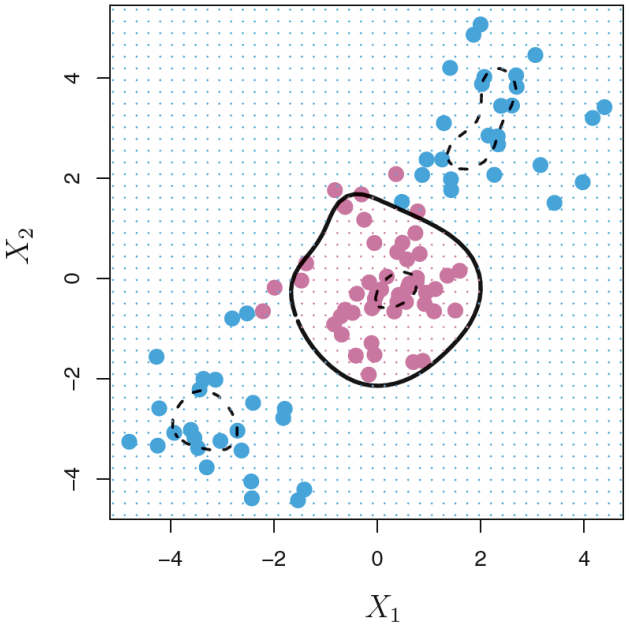
Support Vector Machines



Linear Kernel



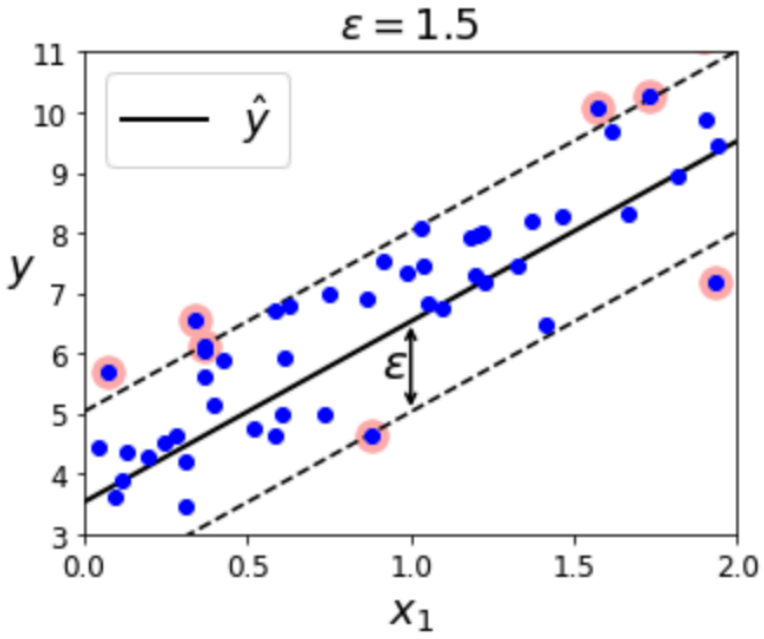
Polynomial Kernel
d=3



Radial Kernel
γ =0.1

Support Vector Machines

- Regression
  - Fit as many points as possible on the street while limiting margin violations.
  - The width of the street is controlled by a hyper-parameter ε



Support Vector Machines

- Hyper-Parameters Tuning
  - C, d: polynomial Kernel
  - γ: RBF kernel
  - ε: for regression
  - Etc.