A gentle introduction to Supervised Learning.

Oriol Pujol

Introduction to Machine Learning

2013

Acknowledgements

This work is inspired in the courses of T. Jaakkola, M. Collins, L. Kaelbling, and T. Poggio at MIT, Andrew Ng at Stanford, Y. Abu-Mostafa at CalTech, E. Xing at CMU, P. Domingos at Georgia and all my mentors and people who made me realize Machine Learning is one of my passions.

Learning goals.

- Get used to matrix notation and linear algebra.
- Understand what a supervised learning problem is.
- Identify the elements of the supervised machine learning pipeline.
- Know the difference between the regression and classification.
- Build the intuition on the hypotheses space and the parameter space.
- Have an example solving a complete supervised problem from scratch.
- Understand the difference among training, validation and test data sets.

Outline

- What is supervised learning?
- An introductory example. The linear regression model.
- Traveling across the parameter space. Descent optimization methods.
- A linear classification example. Introduction to the feature space.
- Our first classifiers.
- Applying the machine learning model.

What is machine learning?

Improve the performance of a software system, based on previous experience

What is machine learning?

Improve the performance of a software system, based on previous experience

- prediction: phone and credit card fraud, medical diagnosis, document retrieval,
- understanding: market basks, discovering astronomical features
- control: flying helicopters, playing backgammon

Key elements for machine learning.

Machine learning is used when

- There is a pattern
- We can not pin it down mathematically
- We have data on it

Which is the most important of the three?

Key elements for machine learning.

Machine learning is used when

- There is a pattern
- We can not pin it down mathematically
- We have data on it

Which is the most important of the three? That's why we also call it Learning from data.

What is machine learning?

Improve the performance of a software system, based on previous experience:

- prediction: **supervised learning** : given (x, y) pairs, find a mapping from a new x to a new y, e.g. regression, classification,
- understanding: **unsupervised learning** : given a set of x. find something interesting or useful about their structure, e.g. density estimation, clustering, dimensionality reduction,
- control:**reinforcement learning** : given an external system upon which you can exert control action a and receive percepts p, a reward signal r indicating good performance, find a mapping from $P \rightarrow A$ that maximizes some long-term measure of r

Generalization in supervised learning

We want to find a model that will perform the best on future examples

- We don't know what the future data will be.
- We have some past data
- We **hope** that future data will remember past data in a way that while let us use the past data to construct a model that will perform well in the future.

Framing the problem

Humans have to do a lot of work, up front, to set up a machine learning problem

- What do you want to predict?
- What kind of data can you get?
- What is the relative costs of different types of errors?
- How does the available data relate to future data?

An example.

Suppose you want to know which grade I may obtain at the end this course.

If you know information (students record) about people who passed this course (data $\mathbf{x}$), and their performance (label y), then, given your own record you may ask the system to predict your grade based on the previous experience and the power of the learning algorithm.

An example.

Av. math grade	Grade in ML
5.0	4.2
6.2	5.9
7.4	8.1
⋮	⋮

Supervised learning:
Given the "right" answer for each example in de data.

Regression problem:
Predict a real-value output.

Notation

Input (features): $\mathbf{x} \in \mathbf{R}^d$ (student's record)
Output (labels): $y \in \mathbf{R}$ (actual grade in ML)
Data: Examples of inputs and output pairs: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

Dataset jargon

We will usually use column-wise notation for each sample.

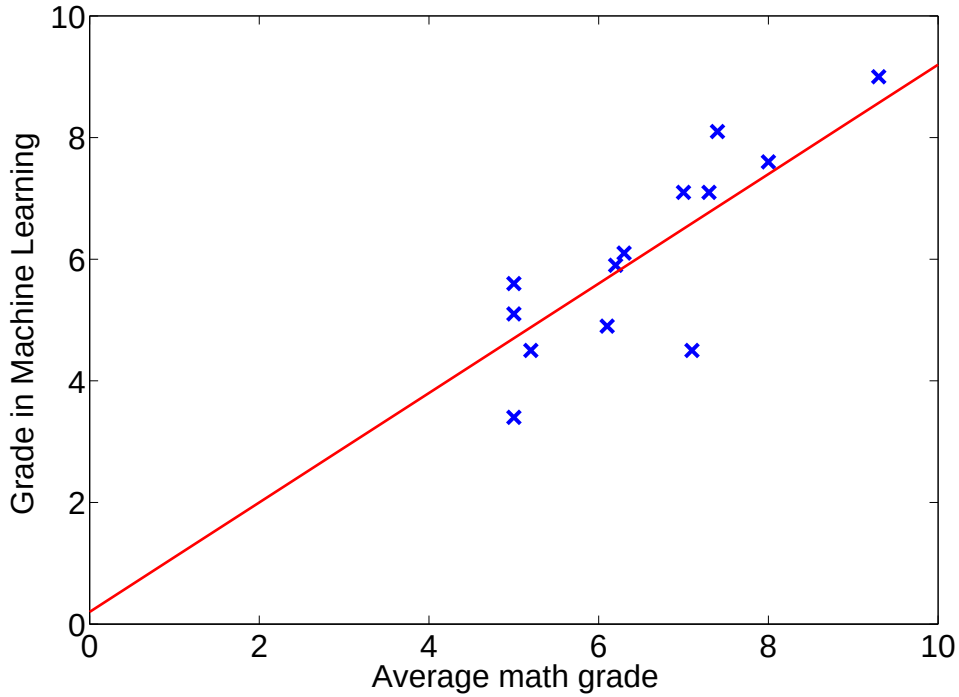
Common jargon:

Rows: features/ attributes/ dimensions.
Columns: instances/ examples/ samples.
The feature to be predicted: target/ outcome/ response/ label/ dependent variable.
The other features: independent variables/ covariates/ predictors/ regressors.

According to the type of data, we can talk about:

- iid (independent identically distributed) vectors
- Time series (dependent vectors)
- Images (matrices)
- Variable-size non-vector data (e.g. strings, trees, graphs, text)
- Objects (e.g. within a relational schema)

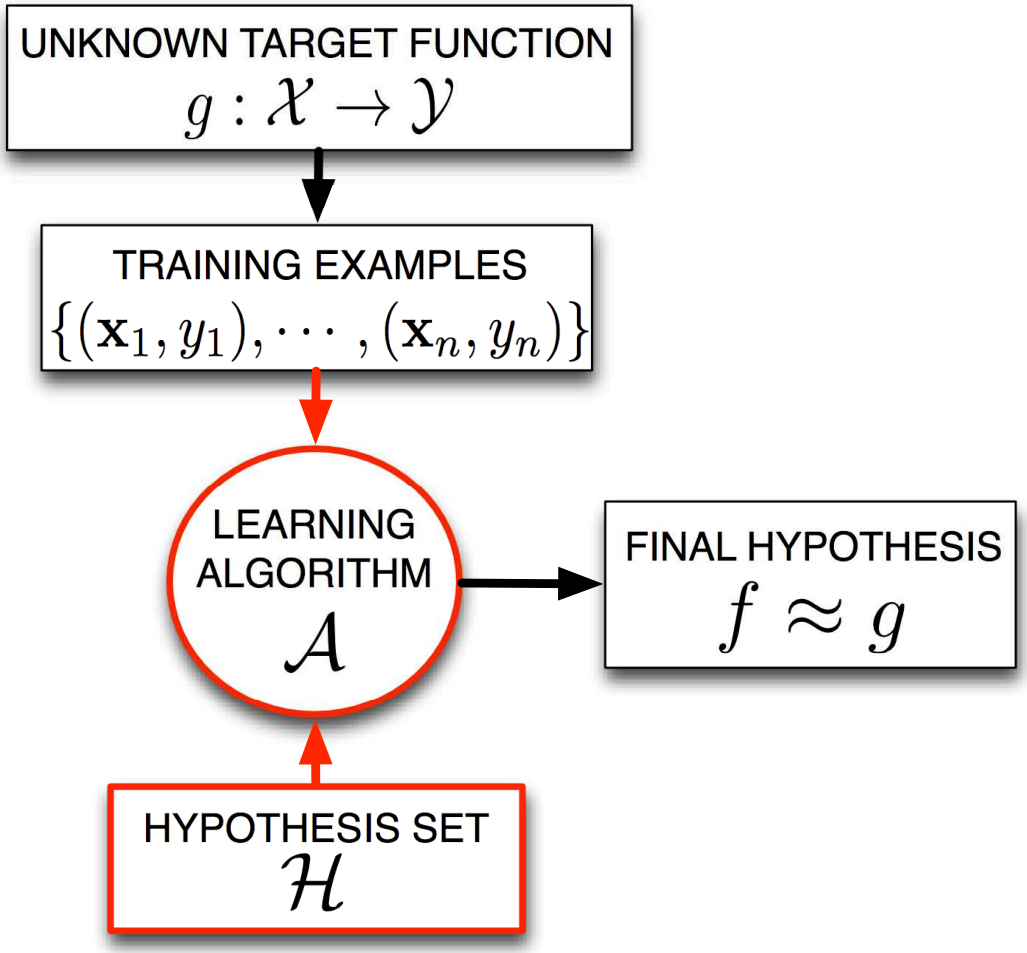
An example.



Hypothesis: $f : \mathbf{R}^d \rightarrow \mathbf{R}$ A model that maps from the input data to the output label, e.g. $f(x) = 0.9x + 0.5$

Learning algorithm: The process for selecting the most appropriate hypothesis from a hypothesis set $\mathcal{H}$, e.g. the set of linear models $\mathcal{H}(w_0, w_1) \equiv w_1 x + w_0$, where (w_0, w_1) are called parameters.

The supervised classification problem.



Elements of the supervised problem.

Data:
Training: $\{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N)\}; \quad (\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$
where
x is the feature data set.
y is the target output / label.
Model:

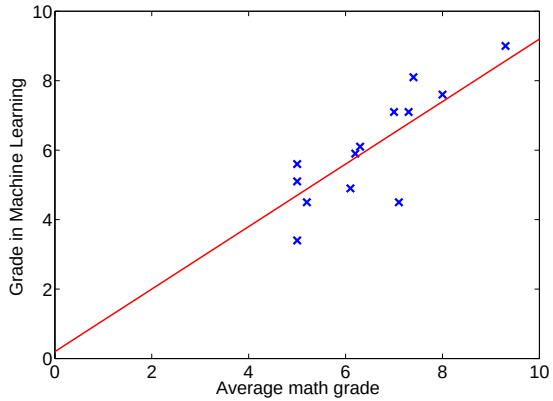
- Hypothesis set: The set of possible / candidate models, $\mathcal{H}$.
- Learning algorithm: The method for selecting the most appropriate model candidate (f) with respect to some quality function (e.g. model accuracy)

Output: The selected model, $f \in \mathcal{H}$.

Our first supervised learning model: Linear regression.

Univariate linear regression hypotheses set

$$\mathcal{H}(w_0, w_1) \equiv w_1 x + w_0.$$



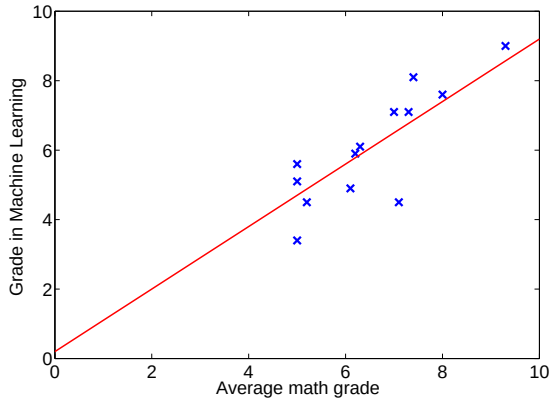
Learning process idea:

We want to find the model defined by the parameters (w_0, w_1) so that our prediction $f(x_i; w_0, w_1)$ on sample x_i is as close as possible to y_i. This is, the "distance" between $f(x_i; w_0, w_1)$ and y_i is minimum for all elements in the training set.

Our first supervised learning model: Linear regression.

Univariate linear regression hypotheses set

$$\mathcal{H}(w_0, w_1) \equiv w_1 x + w_0.$$



Learning process idea:

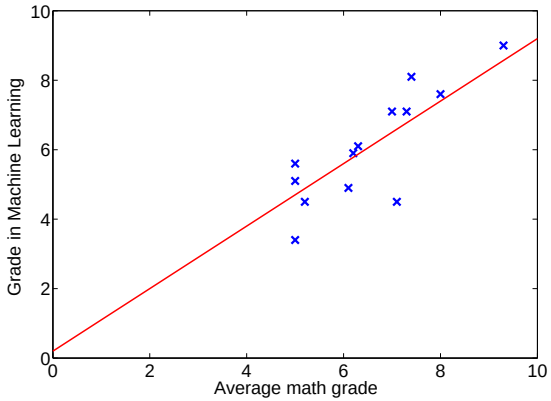
We want to find the model defined by the parameters (w_0, w_1) so that our prediction $f(x_i; w_0, w_1)$ on sample x_i is as close as possible to y_i. This is, the "distance" between $f(x_i; w_0, w_1)$ and y_i is minimum for all elements in the training set.

$$\underset{w_0, w_1}{\text{minimize}} \frac{1}{N} \sum_{i=1}^{N} (f(x_i; w_0, w_1) - y_i)^2$$

Our first supervised learning model: Linear regression.

Univariate linear regression hypotheses set

$$\mathcal{H}(w_0, w_1) \equiv w_1 x + w_0.$$



The last ingredient:

The cost function/ loss function/ error measure:

$$\underset{w_0, w_1}{\text{minimize}}\ J(w_0, w_1)$$

Putting all the elements together.

Hypothesis:

$$f(x; w) = w_1 x + w_0.$$

Parameters:

$$\mathbf{w} = (w_0, w_1)^T$$

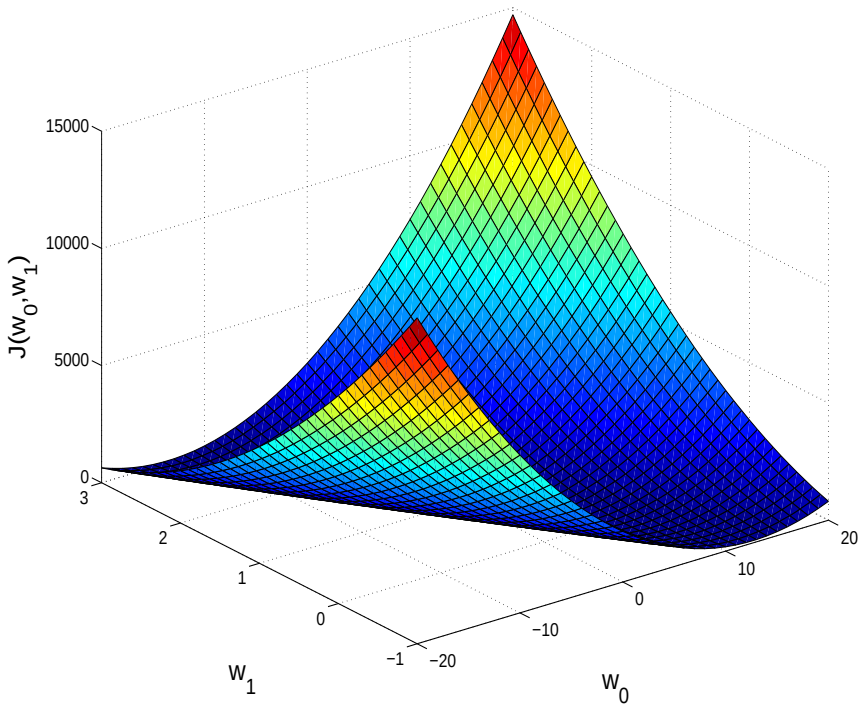
Cost function :

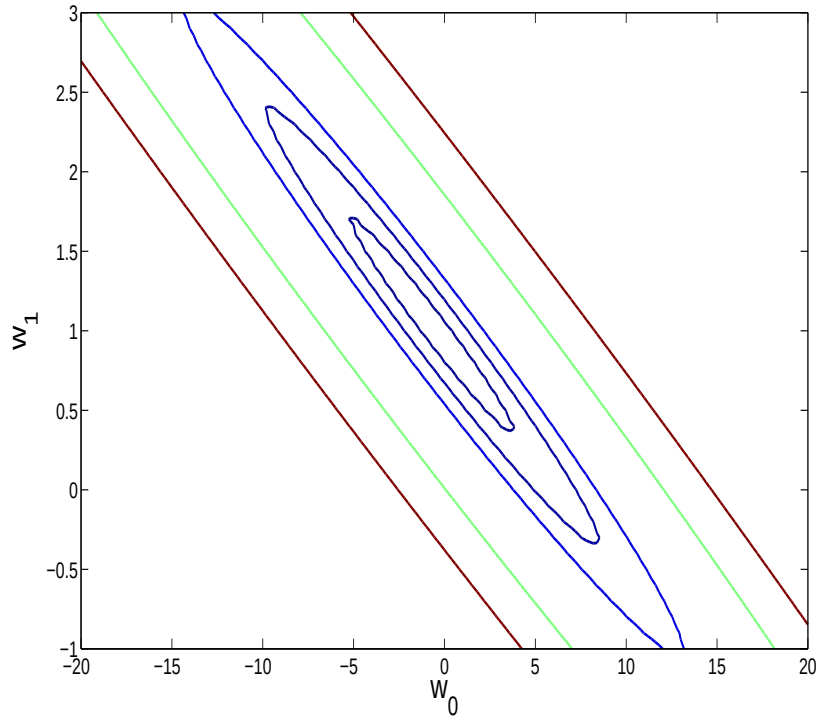
$$J(w_0, w_1) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i; w_0, w_1) - y_i)^2$$

Goal:

$$\underset{w_0, w_1}{\text{minimize}}\ J(w_0, w_1)$$

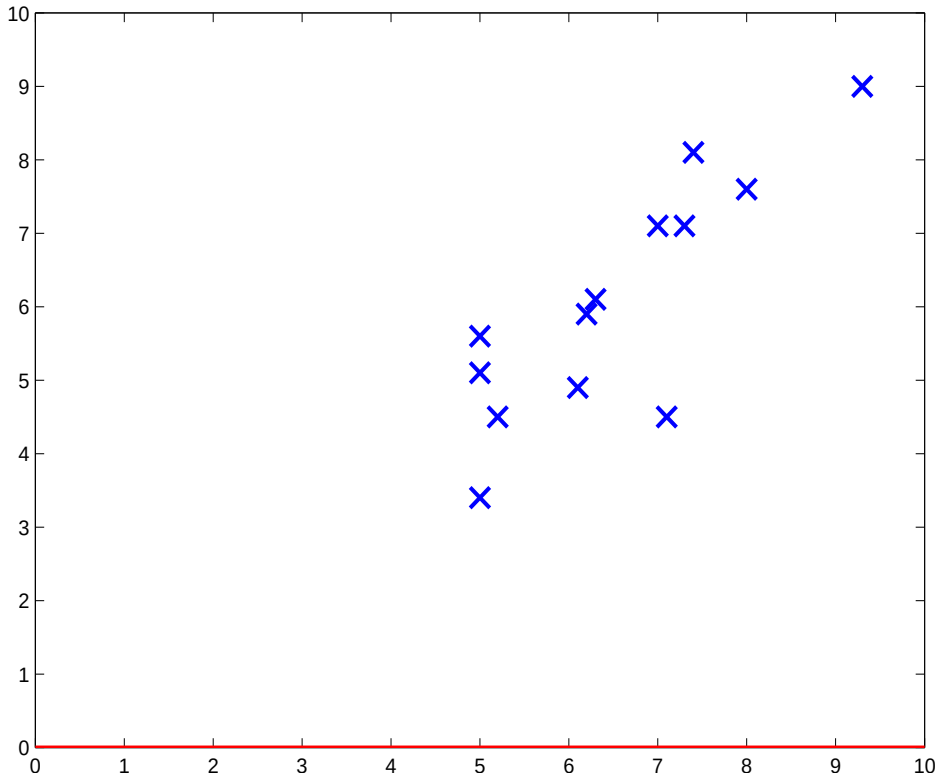
Visualizing the parameter space.



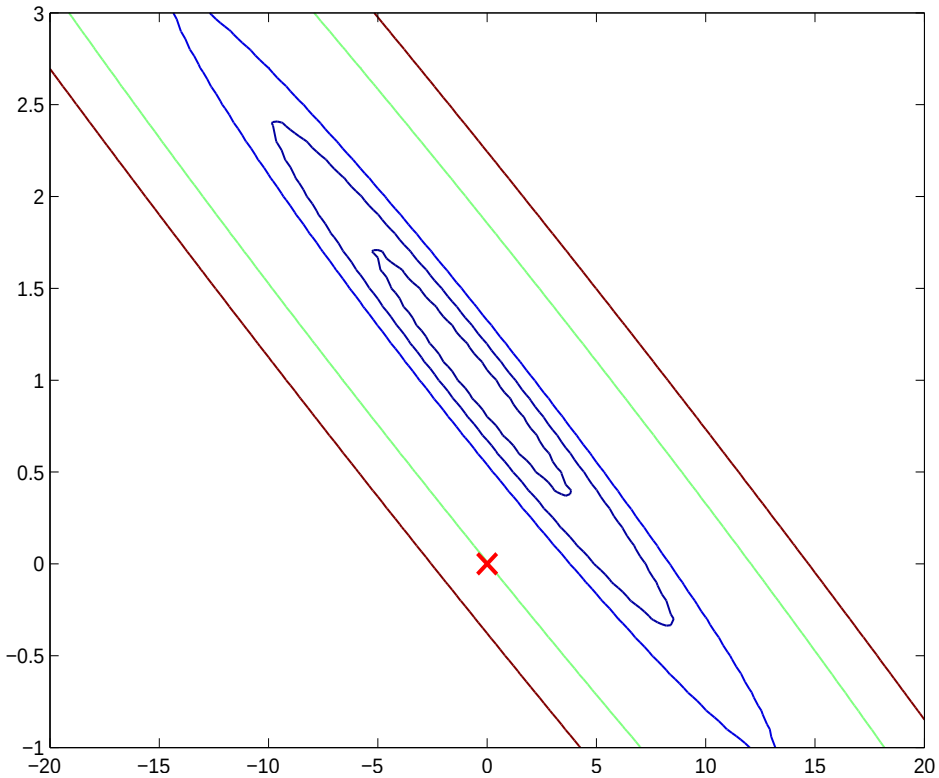


Visualizing the parameter space.

$f_{\mathbf{w}}(x)$



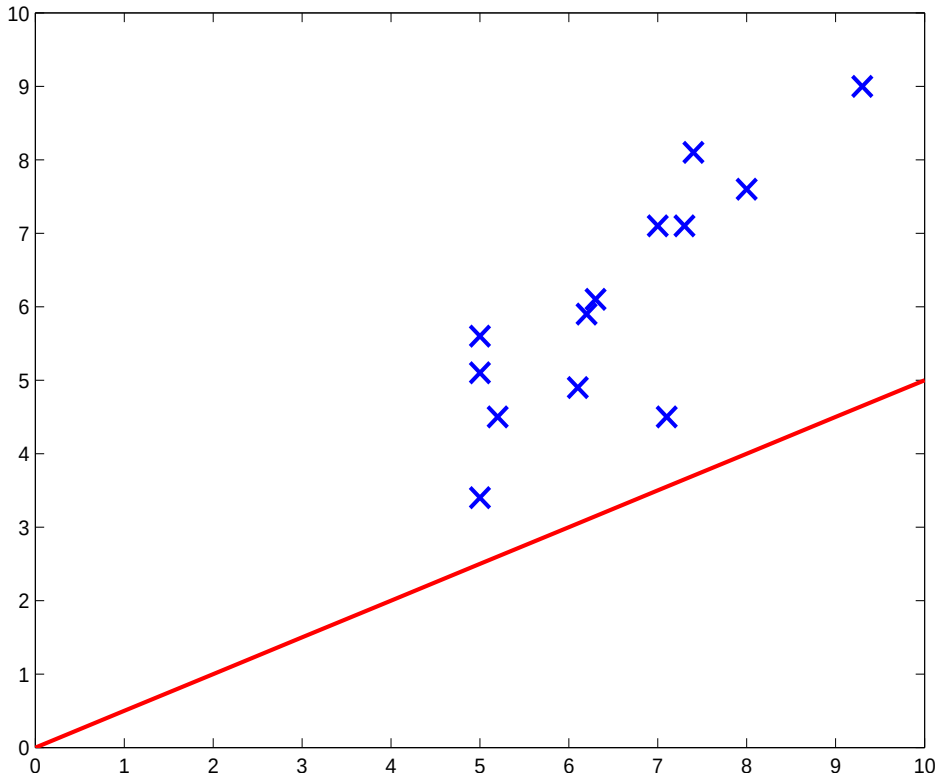
$J(\mathbf{w})$



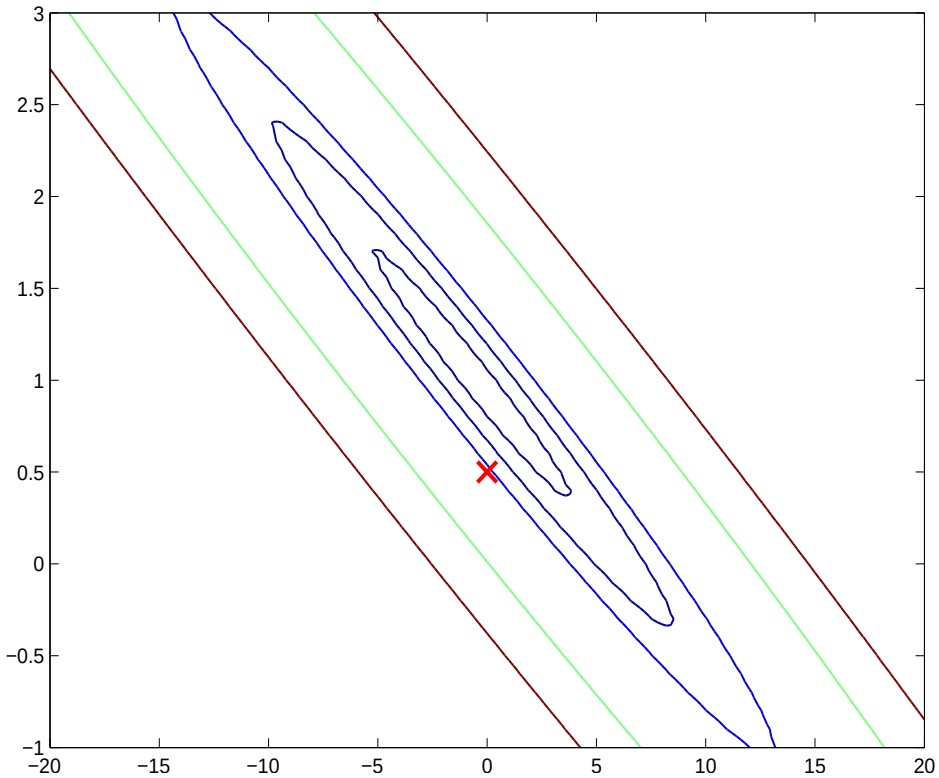
$w_0 = 0,\ w_1 = 0$

Visualizing the parameter space.

$f_{\mathbf{w}}(x)$



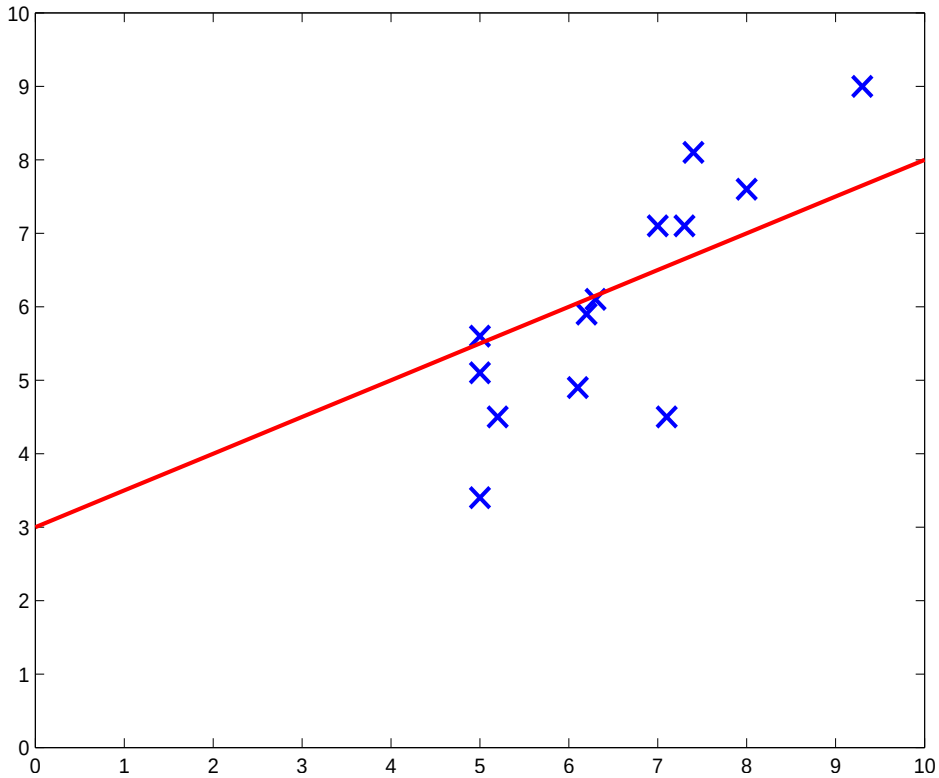
$J(\mathbf{w})$



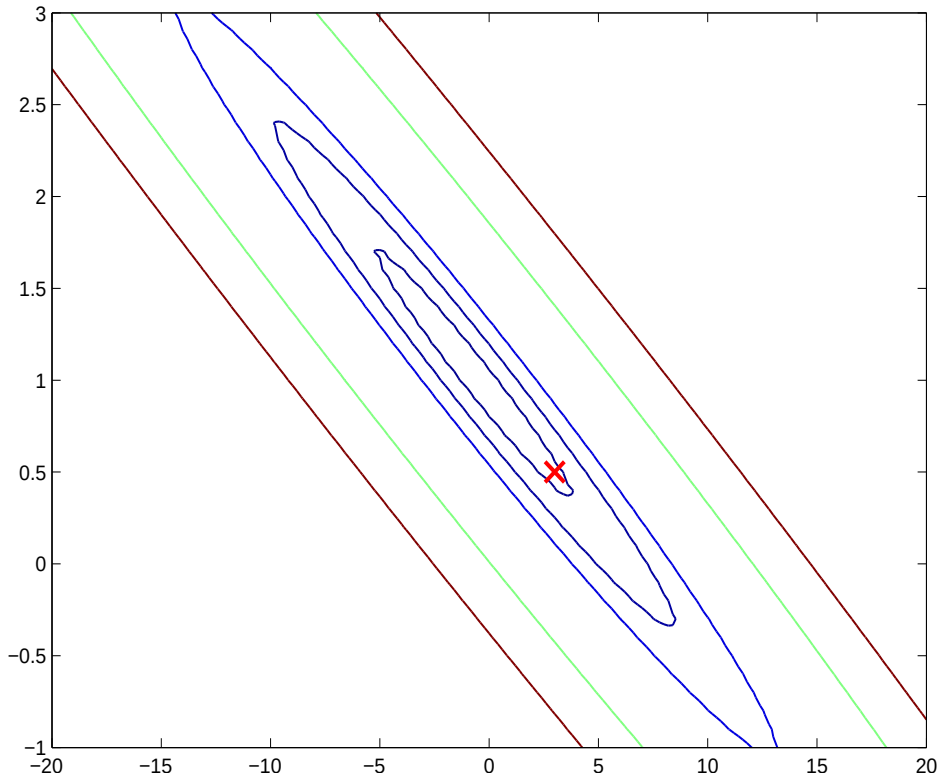
$w_0 = 0,\ w_1 = 0.5$

Visualizing the parameter space.

$f_{\mathbf{w}}(x)$



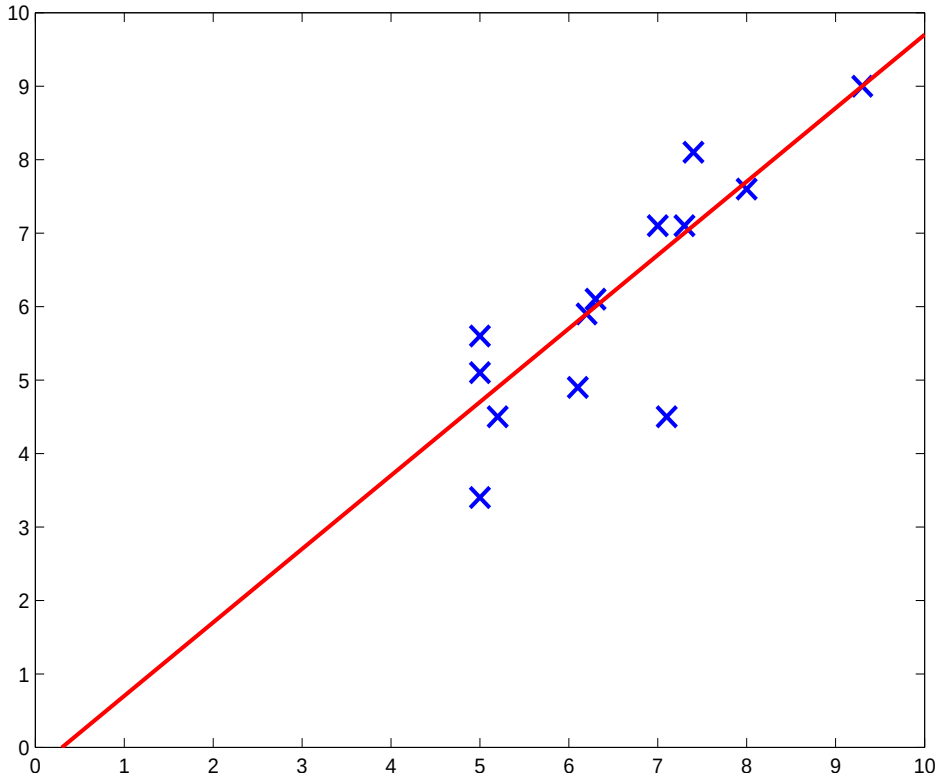
$J(\mathbf{w})$



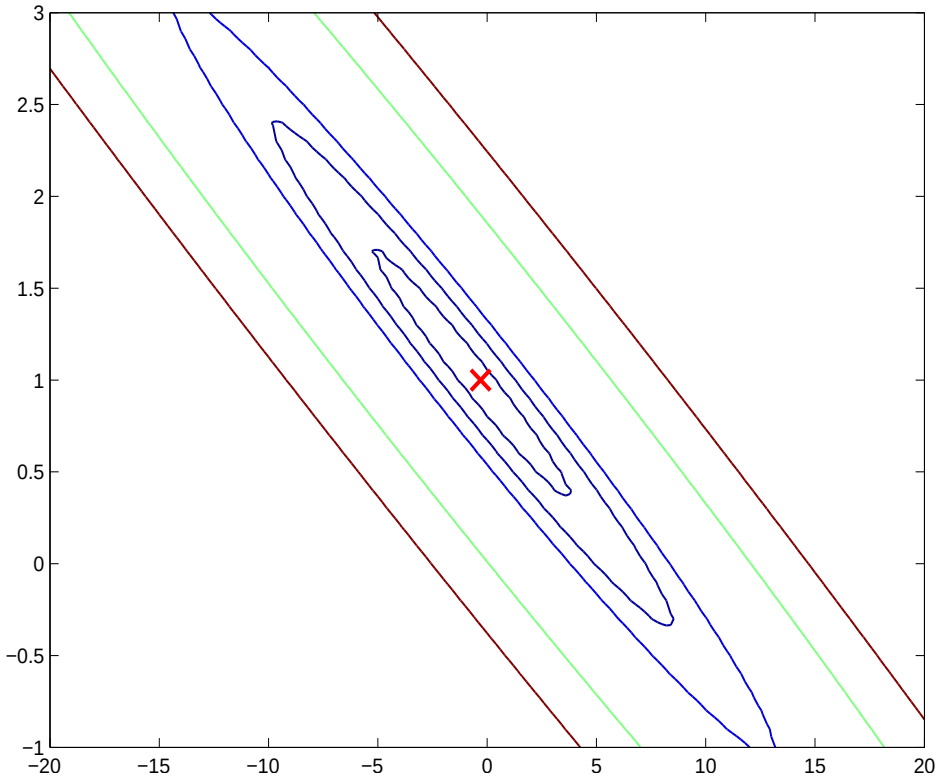
$w_0 = 3,\ w_1 = 0.5$

Visualizing the parameter space.

$f_{\mathbf{w}}(x)$



$J(\mathbf{w})$



$$w_0 = -0.3, \; w_1 = 1$$

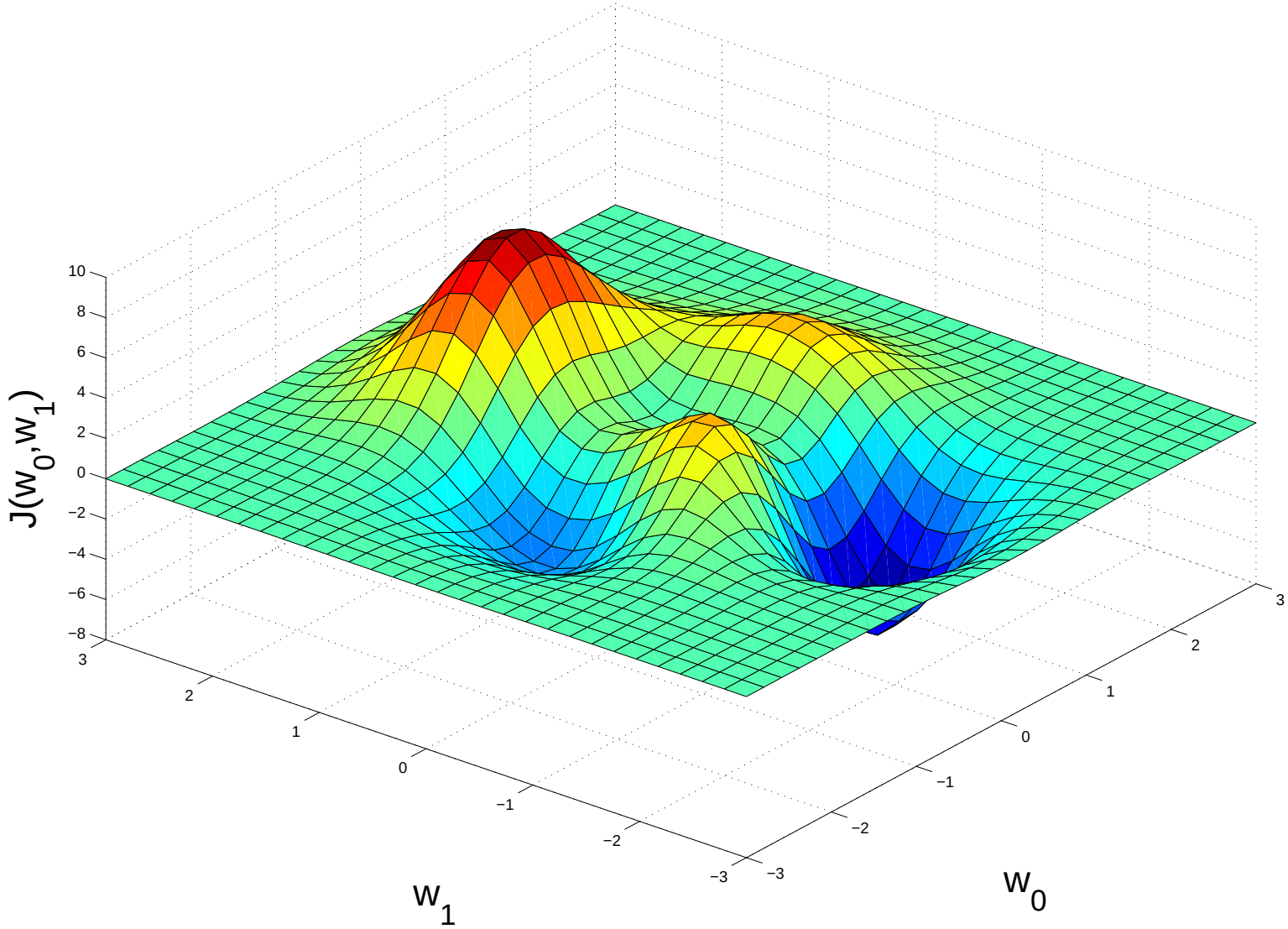
Traveling across the parameter space.

Question: How do we find the minimum in an automated way?

A basic technique:

1. Start with some starting point (guess) $(w_0^{(0)}, w_1^{(0)})$.
2. Change the parameters so that we reduce the cost function $J(w_0^{(k+1)}, w_1^{(k+1)}) < J(w_0^{(k)}, w_1^{(k)})$
3. Repeat (2) until the cost function is not reduced anymore.

Traveling across the parameter space. Intuition.



Traveling across the parameter space with descent methods.

Input: given a starting point $x \in$ **dom** f
Output:
while *stopping criterion is not satisfied:* **do**
    (a) Determine a descent direction Δx;
    [(b) *Line search*. Choose a step size $t > 0$];
    (c) *Update*. $x := x + t\Delta x$;
end

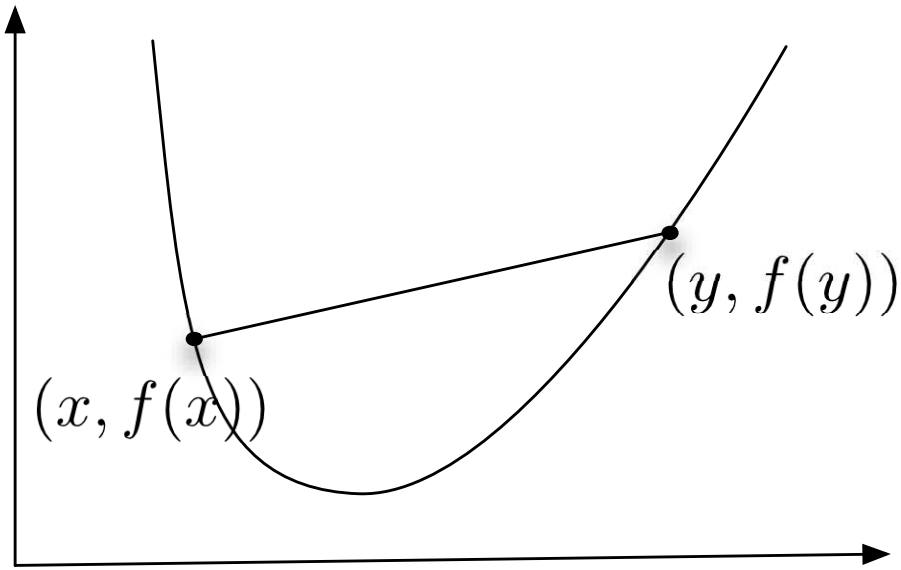
Algorithm 1: General descent method

Traveling across the parameter space with descent methods. Previous concepts.

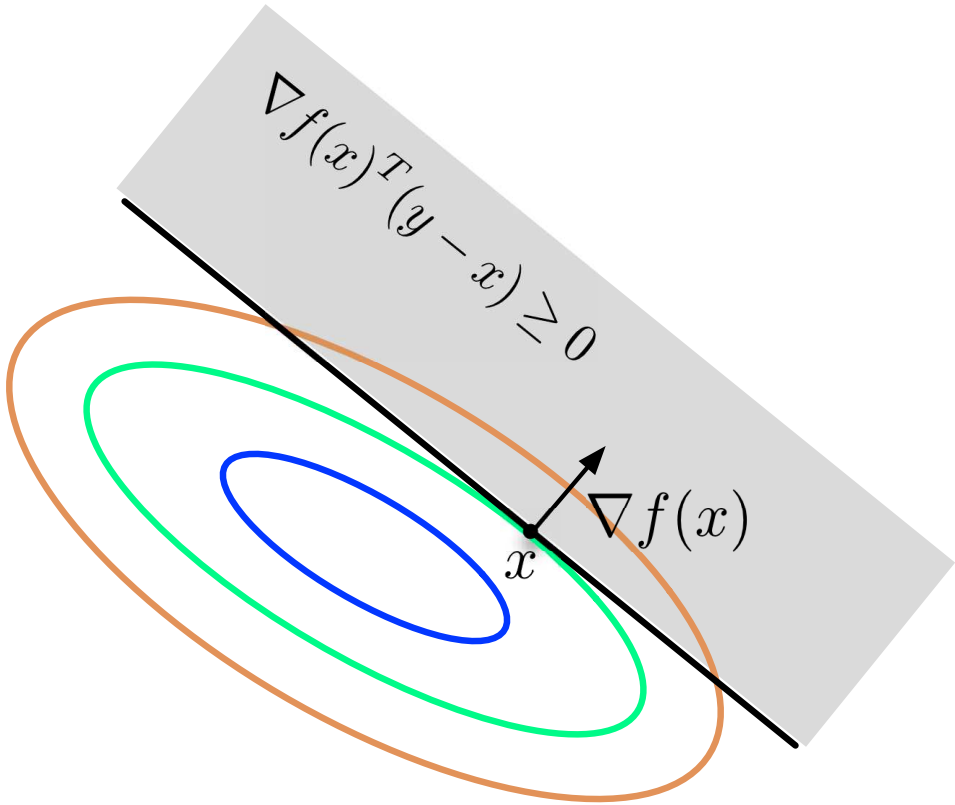
Previous concepts

Definition A function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if **dom** f is a convex set and if for all $x, y \in$ **dom** f, and θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y).$$



Traveling across the parameter space with descent methods.

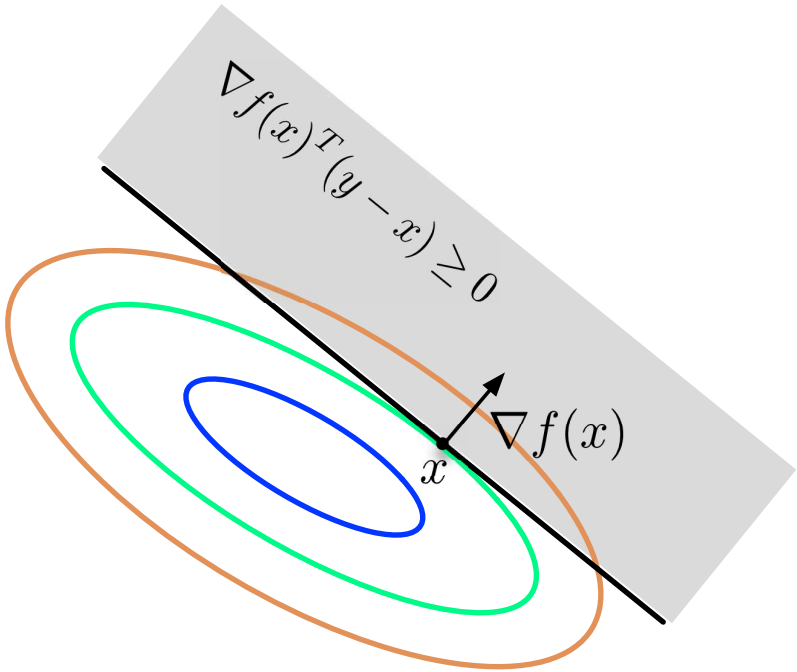


Previous concepts

A differentiable function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if and only if **dom** f is convex and

$$f(y) \geqslant f(x) + \nabla f(x)^T(y-x), \quad \forall x, y \in \textbf{dom } f.$$

Traveling across the parameter space with descent methods.



A descent direction must satisfy (necessary but not sufficient):

$$\nabla f(x)^T \Delta x < 0.$$

A natural choice is to select the steepest descent direction given by

$$\Delta x = -\nabla f(x).$$

Gradient/Steepest descent.

Input: given a starting point $x \in$ **dom** f
Output:
while *stopping criterion is not satisfied:* **do**
    (a) Determine a descent direction $\Delta x = -\nabla f(x)$;
    [(b) *Line search*. Choose a step size $t > 0$];
    (c) *Update*. $x := x + t\Delta x$;
end

Algorithm 2: Gradient descent method

Steepest descent implementation

Hypothesis:

$$f(x; w) = w_0 + w_1 x$$

Cost function :

$$J(w_0, w_1) = \frac{1}{2N} \sum_{i=1}^{N} (f(x_i; w_0, w_1) - y_i)^2$$

Gradient :

$$\nabla J(\mathbf{w}) = \left(\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}\right)^T$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{N} \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i) 1$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{N} \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i) x_i$$

Steepest descent implementation

Input: given a starting point $\mathbf{w} = (-10, -10),\ t = 0.001, \tau = 1000$
Output: model $\mathbf{w}$
for $k = 1 : \tau$ **do**

$$w_0^{(k+1)} := w_0^{(k)} - t\frac{1}{N}\sum_{i=1}^{N}(w_0^{(k)} + w_1^{(k)}x_i - y_i)1;$$

$$w_1^{(k+1)} := w_1^{(k)} - t\frac{1}{N}\sum_{i=1}^{N}(w_0^{(k)} + w_1^{(k)}x_i - y_i)x_i;$$

end

Matrix notation

Hypothesis:

$$f(x; w) = w_0 + w_1 x = \begin{pmatrix} 1 & x \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} 1 & x \end{pmatrix} \mathbf{w} = \tilde{\mathbf{x}}^T \mathbf{w}$$

Cost function :

$$J(w_0, w_1) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i; w_0, w_1) - y_i)^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} (\tilde{\mathbf{x}}_i^T \mathbf{w} - y_i)^2$$

$$= \frac{1}{N} (\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})^T (\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})$$

where

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \cdots & \tilde{\mathbf{x}}_N \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{pmatrix}$$

Steepest descent implementation in matrix notation

Hypothesis:

$$f(x; w) = \tilde{\mathbf{x}}^T \mathbf{w}$$

Cost function :

$$J(\mathbf{w}) = \frac{1}{2N}(\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})^T (\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})$$

Gradient :

$$\nabla J(\mathbf{w}) = \frac{1}{N}\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})$$

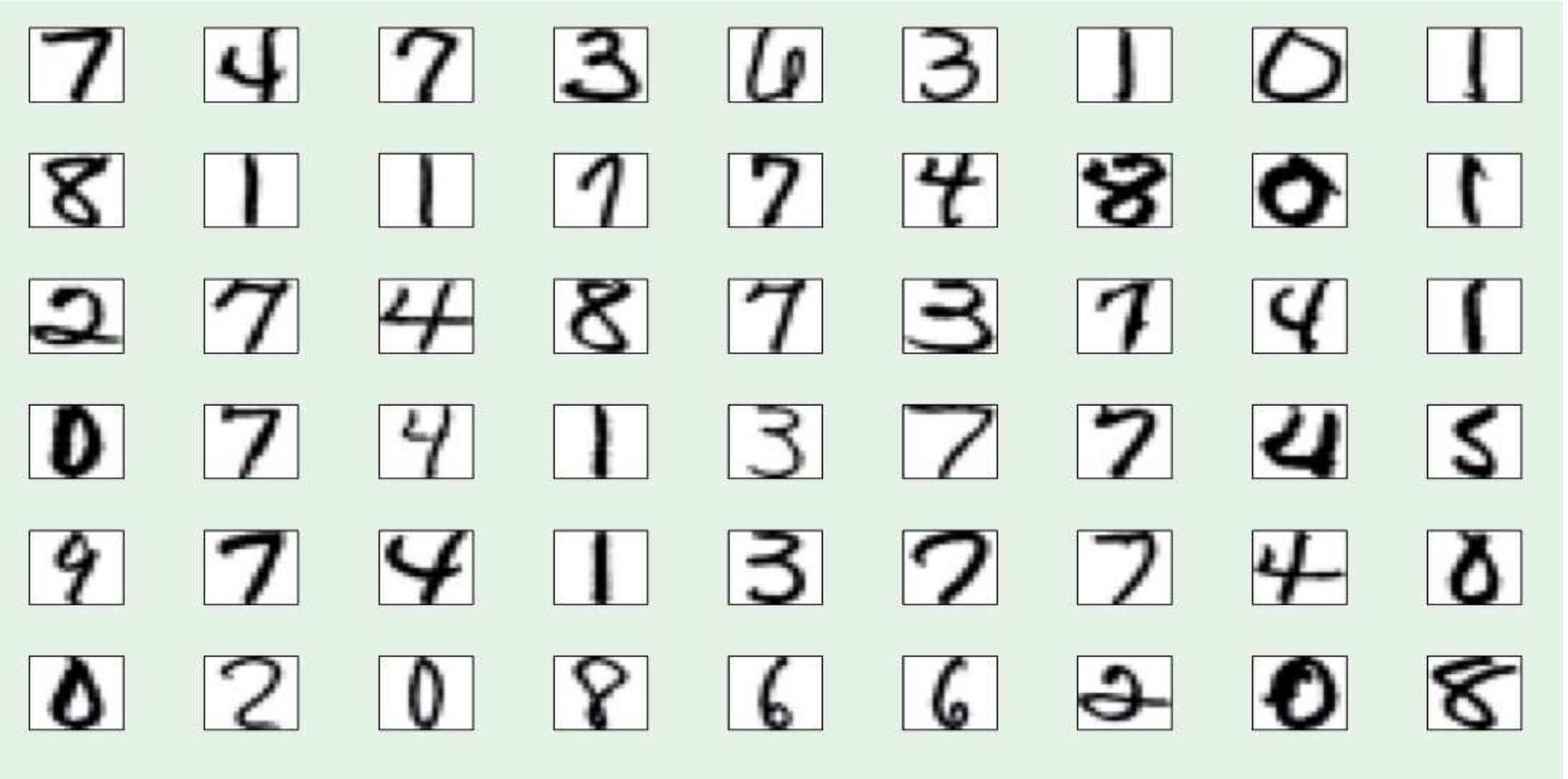
An classification example.

Problem introduction

Instead of the grade, suppose you want to simply know if you will pass the course. If you know information (students record) about people who passed this course (data $\mathbf{x}$), and their discrete performance (label $y = 0$ if she fails and $y = 1$ if she pass), then, given your own record you may ask the system to predict whether you will pass or not based on the previous experience and the power of the learning algorithm.

A real dataset

Alternatively, there are problems with labels defined in the discrete domain, e.g. the MNIST dataset of handwritten digits.



The input space: data representation

The input to the supervised learning training system are the data pairs $(\mathbf{x}_i, y_i)$ corresponding to the i-th observation or sample.

Raw data.

- Pros: No domain specific knowledge is needed.
- Cons: Highly redundant $\Longrightarrow$ high dimensionality.
- Cons: Unknown discriminability.

Example: An 8×8 image is represented as $\mathbf{x} = (x_1, \ldots, x_{256})$ where x_i corresponds to image intensity level.

The input space: data representation

The input to the supervised learning training system are the data pairs $(\mathbf{x}_i, y_i)$ corresponding to the i-th observation or sample.

Dimensionality reduction and manifold learning. Embed data that originally lies in a high dimensional space in a lower dimensional space, while preserving characteristic properties.

- Pros: No domain specific knowledge is needed.
- Pros: Lower dimensionality $\Longrightarrow$ less complexity.
- Cons: Information can be lost.
- Cons: Unknown discriminability.

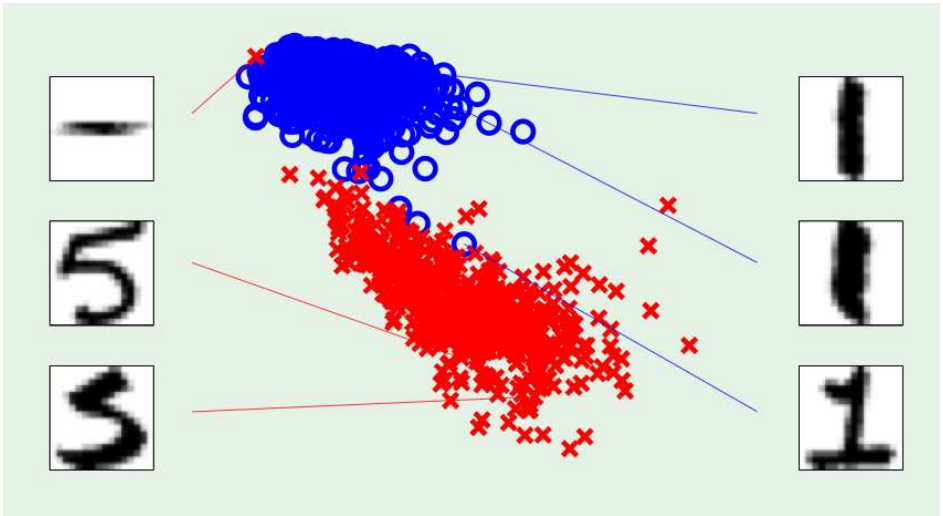
Example: Project the raw data vector using Principal Component Analysis into an space that preserves 90% of the "energy".

The input space: data representation

The input to the supervised learning training system are the data pairs $(\mathbf{x}_i, y_i)$ corresponding to the i-th observation or sample.

Feature extraction.

- Pros: Attempt to capture discriminant data.
- Pros: Lower dimensionality $\Longrightarrow$ less complexity.
- Cons: Domain specific knowledge is needed.



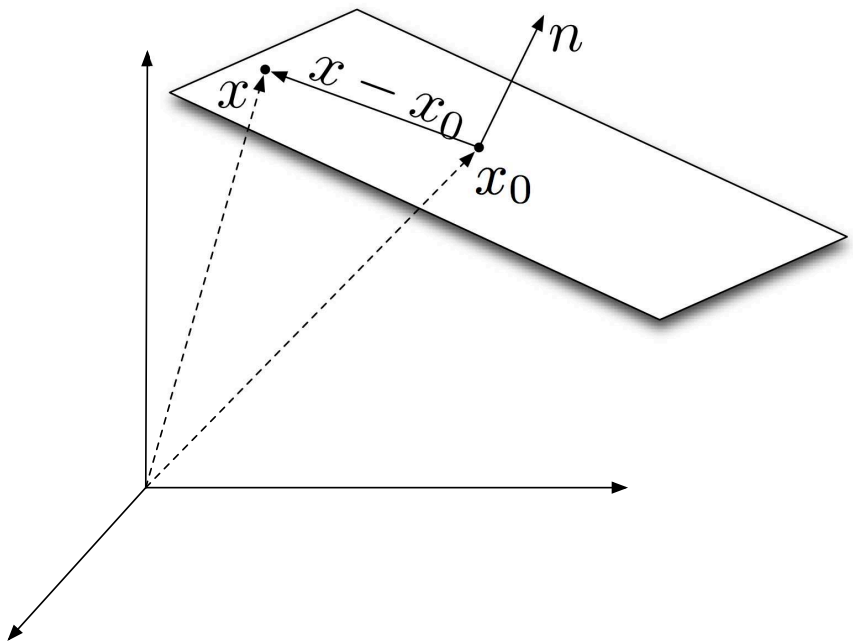
Example: Use high level descriptors, such as symmetry or average image intensity.

The linear model and basic geometry.

Given $x \in \mathbf{R}^{\mathbf{N}}$ we define a hyperplane as the set of points that fulfills the following relation

$$\{x | a^T x = b\}$$

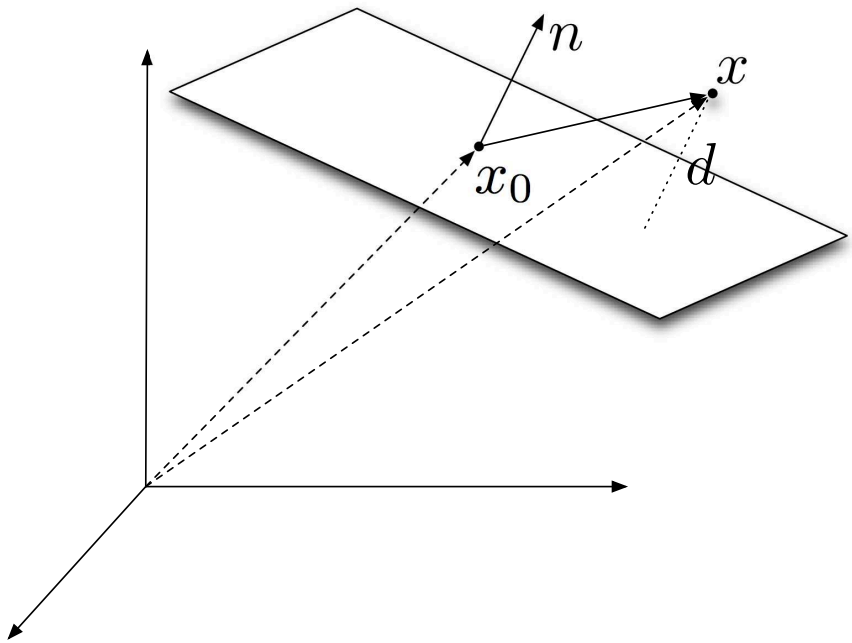
It can be decomposed as $\{x | n^T (x - x_0) = 0\}$ where x_0 is the *base point* and n is the *normal* vector to the hyperplane.



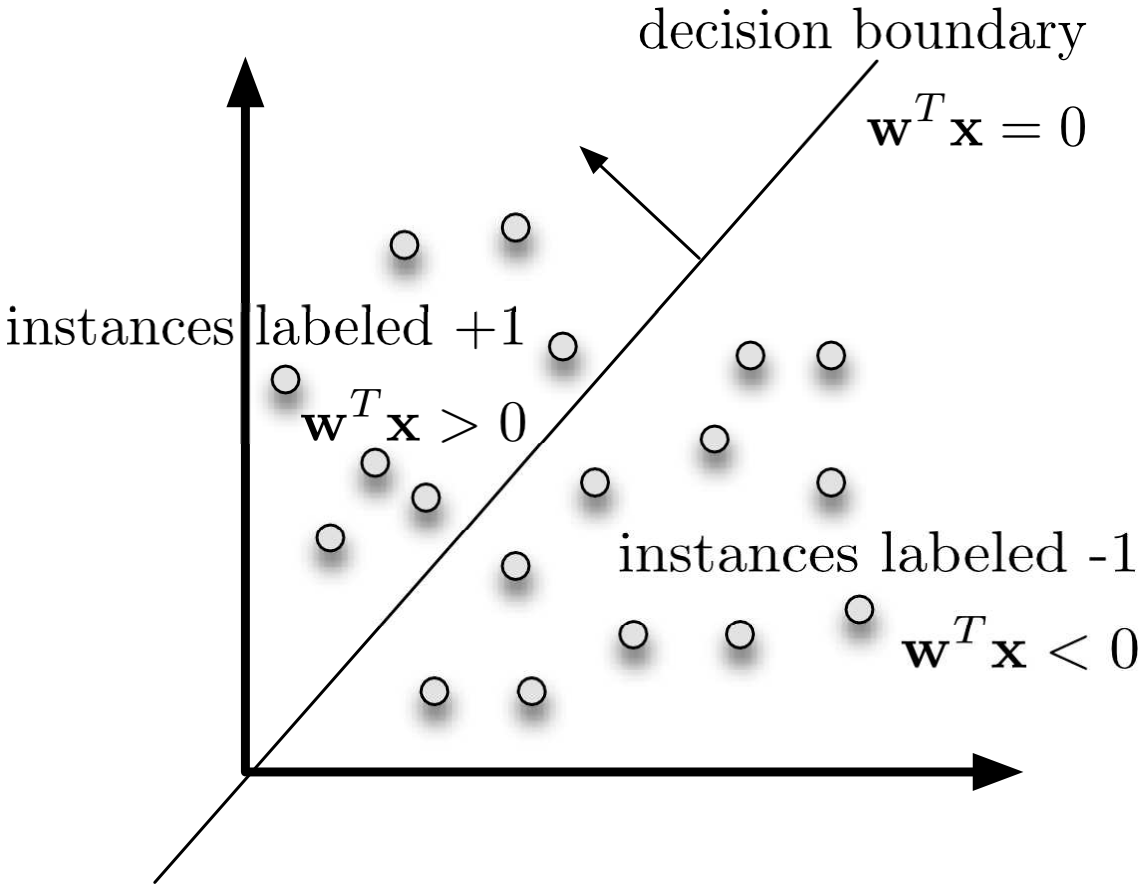
The linear model and basic geometry.

Observe that if $x \notin \pi(x)$ then;

- $d(x, \pi) = |n^T(x - x_0)|/\|n\|_2$ is the distance from x to the hyperplane.
- and the sign$(n^T(x - x_0))$ identifies the half-space where the point lies.



Our first classifier



Our first classifier

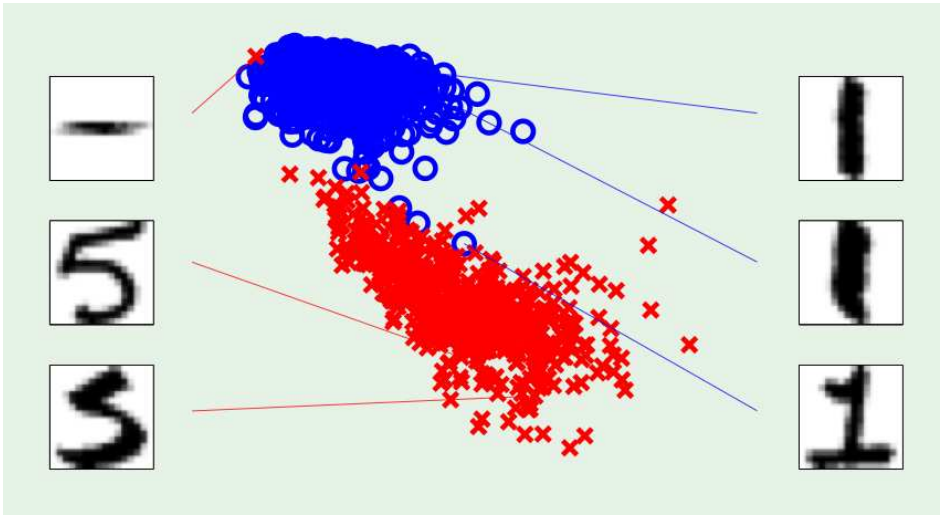
For input $\mathbf{x} = (x_1, \ldots, x_d)$ attributes/features.

- $\mathbf{x}$ belongs to class '5' if:

$$\sum_{i=1}^{d} w_i x_i > \text{threshold}$$

- $\mathbf{x}$ belongs to class '1' if:

$$\sum_{i=1}^{d} w_i x_i \leqslant \text{threshold}$$

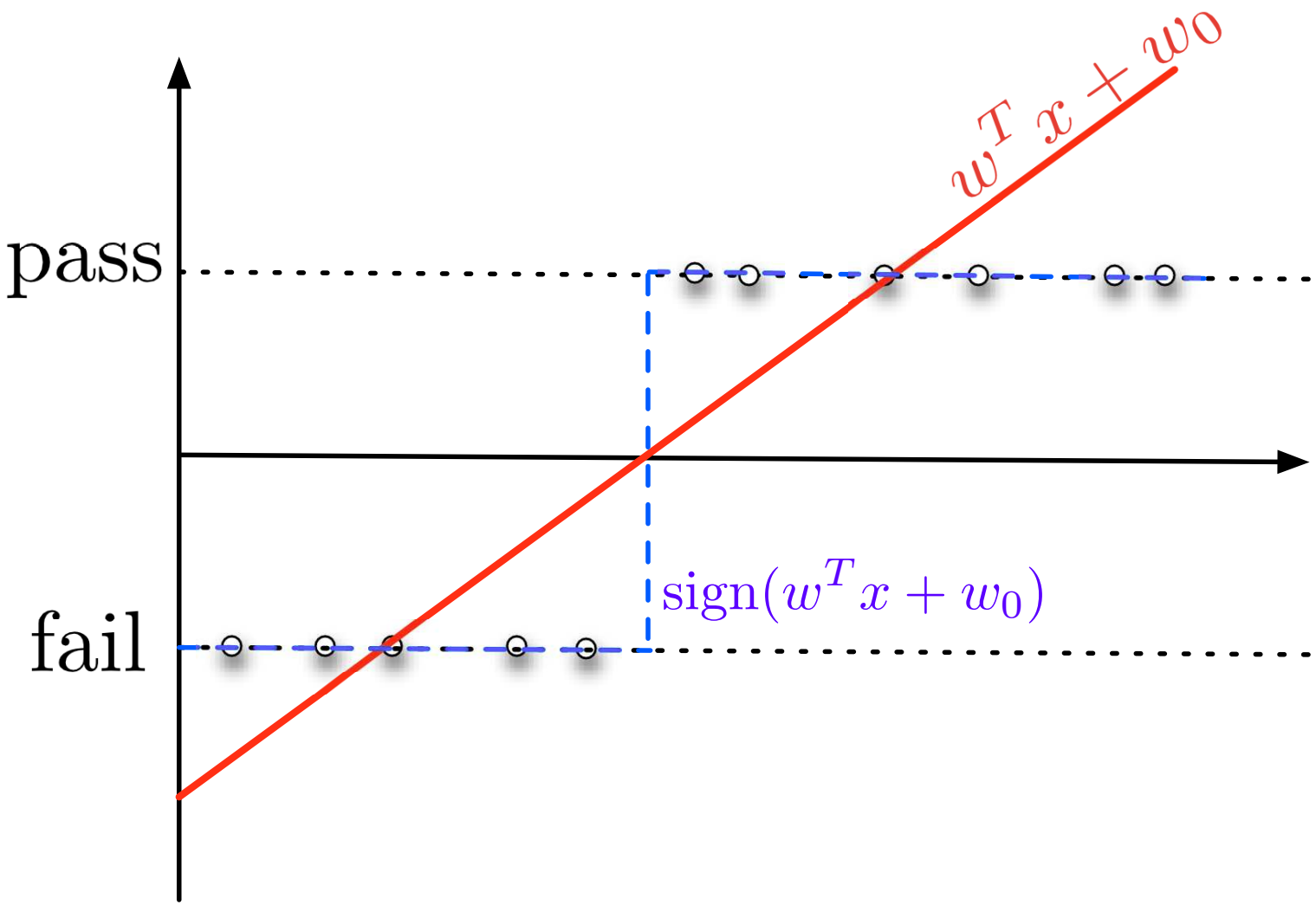


This can be written as $h \in \mathcal{H}$

$$h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + \underbrace{w_0}_{-\text{threshold}}\Big) = \text{sign}(\mathbf{w}^T \tilde{\mathbf{x}})$$

Thresholded linear regression as a classifier.

In our fail or pass problem, the classifier looks like:



The elements in the classification context.

Exactly the same elements and models as in linear regression learning.

Hypothesis:

$$f(x; w) = \tilde{\mathbf{x}}^T \mathbf{w}$$

Cost function :

$$J(\mathbf{w}) = \frac{1}{N}(\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})^T (\tilde{\mathbf{X}}^T \mathbf{w} - \mathbf{y})$$

Differences

- Define y,e.g. $y \in \{-1, +1\}$.
- **Final classification model:**

$$h(x) = \text{sign}(f(x; w))$$

Learning linear regression (part II). Analytic solution.

We have seen that in order to find the solution we can use iterative methods. However, this particular problem has a closed-form analytic solution.

$$w^* = \arg\min_w \frac{1}{N}(\tilde{\mathbf{X}}^T w - \mathbf{y})^T(\tilde{\mathbf{X}}^T w - \mathbf{y})$$

$$\frac{\partial J}{\partial w} = \frac{2}{N}\tilde{\mathbf{x}}(\tilde{\mathbf{X}}^T w - \mathbf{y}) = 0$$

Solving for w,

$$\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T w^* - \tilde{\mathbf{X}} y = 0 \implies (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T) w^* = \tilde{\mathbf{X}} y$$

$$w^* = (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T)^{-1}\tilde{\mathbf{X}} y \implies w^* = \tilde{\mathbf{X}}^\dagger y$$

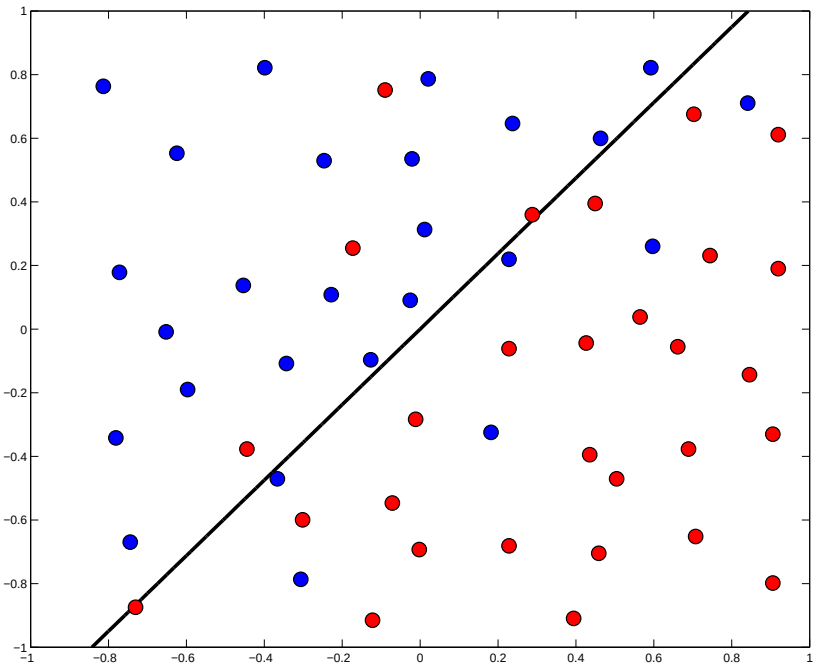
where $\tilde{\mathbf{X}}^\dagger = (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T)^{-1}\tilde{\mathbf{X}}$ is called the **pseudo-inverse** of $\mathbf{X}$.

Iterative vs analytic learning.

Iterative methods	Analytic solution
• Need to choose the learning rate t. • May be slow. • Usually, each iteration is computationally cheap. • Can be used with large number of data.	• Looks nicer. • No iterations are required. • Large computational cost. In the simplest case where matrix inversion is needed it takes $\mathcal{O}(n^3)$.

Thresholded linear regression in action

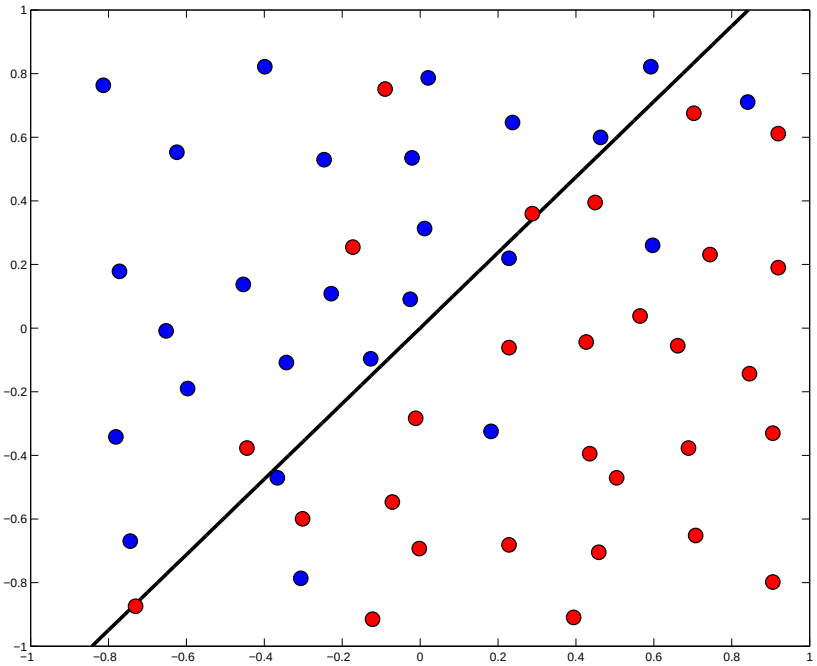
Classification result



Quite good result!

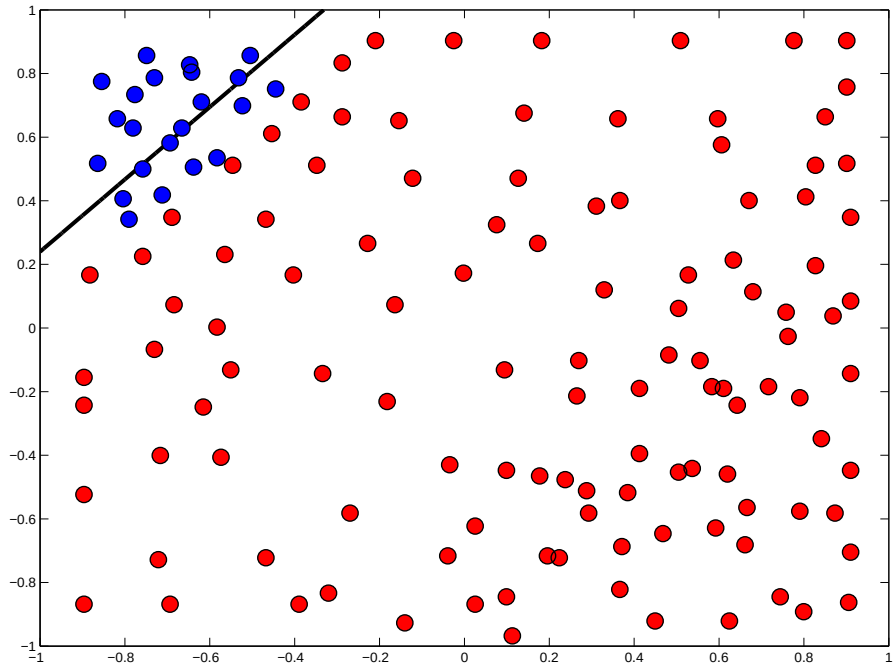
Thresholded linear regression in action

Classification result



Quite good result!

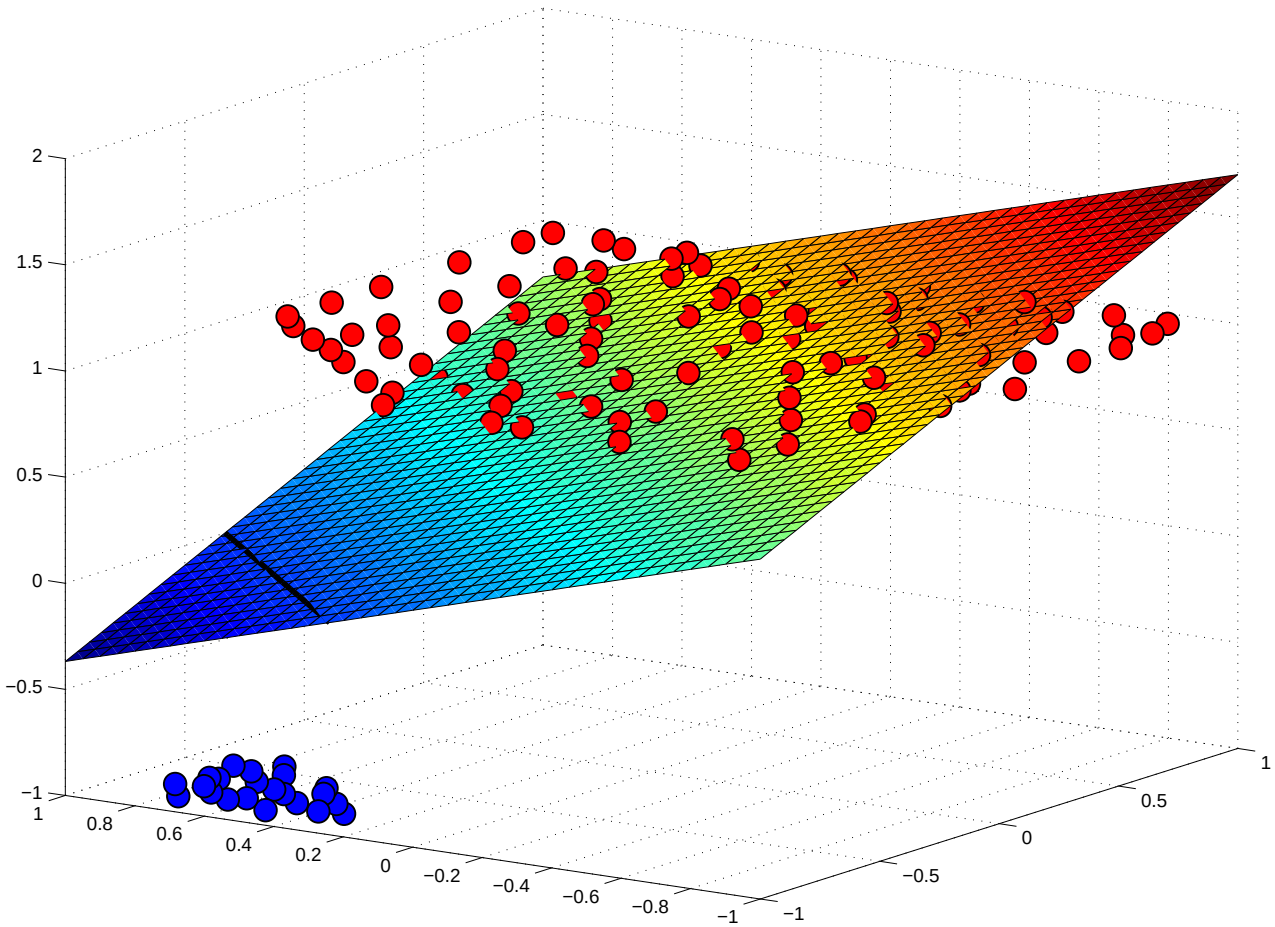
Classification result



What is happening?

Linear regression in action

Let us look at a 3D visualization of the problem and the model we obtain



Application of the learning model.

Training

We are given a dataset $\mathcal{D}$ which we use to learn our model parameters, e.g. the parameters of the linear regressor.

Exploitation

Apply the learned model to new data and **hope** it predicts correctly.

A first evaluation of the learning model.

But... we want to know approximately how well it will perform during the exploitation step!!. What to do?

A first evaluation of the learning model.

But... we want to know approximately how well it will perform during the exploitation step!!. What to do?

The simplest evaluation strategy: Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in two sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$$

A first evaluation of the learning model.

But... we want to know approximately how well it will perform during the exploitation step!!. What to do?

The simplest evaluation strategy: Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in two sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$$

Training

Use $\mathcal{D}_{train}$ to learn the model.

Evaluation

Use $\mathcal{D}_{test}$ to compute the performance of the method.

Exploitation

Apply the model to new data and **hope** it predicts correctly.

A first evaluation of the learning model.

Suppose that we have two different models. We want to select the one that has a better performance during the exploitation step. But... we don't have exploitation data to evaluate on. What to do?

A first evaluation of the learning model.

Suppose that we have two different models. We want to select the one that has a better performance during the exploitation step. But... we don't have exploitation data to evaluate on. What to do?

Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in two sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{validation}$$

A first evaluation of the learning model.

Suppose that we have two different models. We want to select the one that has a better performance during the exploitation step. But... we don't have exploitation data to evaluate on. What to do?

Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in two sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{validation}$$

Training

Use $\mathcal{D}_{train}$ to learn both models.

Evaluation

Use $\mathcal{D}_{validation}$ to decide which of the two models better adapts to our problem.

Apply the selected model to new data and **hope** it predicts correctly.

A first evaluation of the learning model.

But... I want to know the expected performance of the best method!!!
What to do?

A first evaluation of the learning model.

But... I want to know the expected performance of the best method!!!
What to do?

Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in three sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{validation} \cup \mathcal{D}_{test}$$

A first evaluation of the learning model.

But... I want to know the expected performance of the best method!!!
What to do?

Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in three sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{validation} \cup \mathcal{D}_{test}$$

Training and model selection

Use $\mathcal{D}_{train}$ to learn both models. Use $\mathcal{D}_{validation}$ to decide which of the two models better adapts to our problem.

Evaluation

Use $\mathcal{D}_{test}$ to compute the performance of the selected method.

Apply the selected model to new data and **hope** it predicts correctly.

Quiz