# Introduction to Machine Learning

## Gradient and Regularization Methods

Andres Mendez-Vazquez

February 3, 2023

# Outline

# Outline

# Given that the Canonical Solution has problems

**We can develop a more robust algorithm**

- Using the Gradient Descent Idea

# Given that the Canonical Solution has problems

## We can develop a more robust algorithm

- Using the Gradient Descent Idea

## Basically, The Gradient Descent

- It uses the change in the surface of the cost function to obtain a direction of improvement.

# Gradient Descent

## The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}(1)$.

# Gradient Descent

**The basic procedure is as follow**

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.

# Gradient Descent

**The basic procedure is as follow**

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

# Gradient Descent

**The basic procedure is as follow**

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k)) \tag{1}$$

# Gradient Descent

**The basic procedure is as follow**

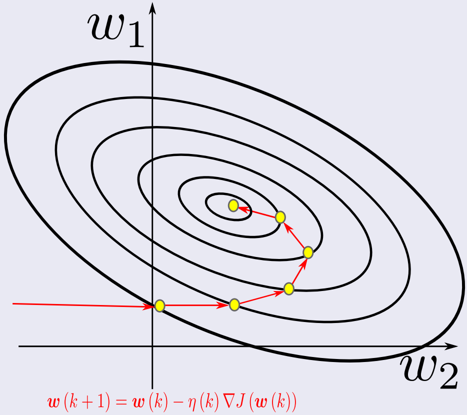
1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k)) \tag{1}$$

$\eta(k)$ is a positive scale factor or learning rate!!!

# Geometrically

## We have the following



# Outline

# For our full regularized equation

## We have

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d+1} w_j^2 \tag{2}$$

# For our full regularized equation

## We have

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d+1} w_j^2 \tag{2}$$

## Then, for each $w_j$

$$\frac{dJ(\boldsymbol{w})}{dw_j} = -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_j^i\right] + \lambda w_j \tag{3}$$

# For our full regularized equation

## We have

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d+1} w_j^2 \quad (2)$$

## Then, for each $w_j$

$$\frac{dJ(\boldsymbol{w})}{dw_j} = -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_j^i\right] + \lambda w_j \quad (3)$$

## Therefore

$$\nabla J(\boldsymbol{w}(k)) = \begin{pmatrix} -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_1^i\right] + \lambda w_1 \\ \vdots \\ -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_{d+1}^i\right] + \lambda w_{d+1} \end{pmatrix}$$

# Outline

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k)\nabla J(\boldsymbol{w}(k-1))$$

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\qquad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k)\nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k)\nabla J(\boldsymbol{w}(k)) < \theta$

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\qquad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\quad\quad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

## Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!

# Algorithm

## Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion $\theta$, $\eta(\cdot)$, $k=0$
2. do $k=k+1$
3. $\qquad \boldsymbol{w}(k)=\boldsymbol{w}(k-1)-\eta(k)\nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k)\nabla J(\boldsymbol{w}(k))<\theta$
5. return $\boldsymbol{w}$

## Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

# Outline

# Using the Taylor's second-order expansion around value $\boldsymbol{w}(\boldsymbol{k})$

## We do the following

$$J(\boldsymbol{w}) = J(\boldsymbol{w}(k)) + \nabla J^{T}(\boldsymbol{w} - \boldsymbol{w}(k)) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}(k))^{T} \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}(k)) \tag{4}$$

# Using the Taylor's second-order expansion around value $\boldsymbol{w}\left(\boldsymbol{k}\right)$

## We do the following

$$J\left(\boldsymbol{w}\right)=J\left(\boldsymbol{w}\left(k\right)\right)+\nabla J^{T}\left(\boldsymbol{w}-\boldsymbol{w}\left(k\right)\right)+\frac{1}{2}\left(\boldsymbol{w}-\boldsymbol{w}\left(k\right)\right)^{T}\boldsymbol{H}\left(\boldsymbol{w}-\boldsymbol{w}\left(k\right)\right) \quad (4)$$

Remark: This is know as Taylor's Second Order expansion!!!

# Using the Taylor's second-order expansion around value $\boldsymbol{w}(\boldsymbol{k})$

## We do the following

$$J(\boldsymbol{w}) = J(\boldsymbol{w}(k)) + \nabla J^T(\boldsymbol{w} - \boldsymbol{w}(k)) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}(k))^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}(k)) \quad (4)$$

Remark: This is know as Taylor's Second Order expansion!!!

## Here, we have

- $\nabla J$ is the vector of partial derivatives $\frac{\partial J}{\partial w_i}$ evaluated at $\boldsymbol{w}(k)$.

# Using the Taylor's second-order expansion around value $\boldsymbol{w}(\boldsymbol{k})$

## We do the following

$$J(\boldsymbol{w}) = J(\boldsymbol{w}(k)) + \nabla J^T (\boldsymbol{w} - \boldsymbol{w}(k)) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}(k))^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}(k)) \quad (4)$$

Remark: This is know as Taylor's Second Order expansion!!!

## Here, we have

- $\nabla J$ is the vector of partial derivatives $\frac{\partial J}{\partial w_i}$ evaluated at $\boldsymbol{w}(k)$.
- $\boldsymbol{H}$ is the Hessian matrix of second partial derivatives $\frac{\partial^2 J}{\partial w_i \partial w_j}$ evaluated at $\boldsymbol{w}(k)$.

# Then

**We substitute (Eq. 1) into (Eq. 4)**

$$\boldsymbol{w}(k+1) - \boldsymbol{w}(k) = \eta(k)\nabla J(\boldsymbol{w}(k)) \tag{5}$$

# Then

## We substitute (Eq. 1) into (Eq. 4)

$$\boldsymbol{w}(k+1) - \boldsymbol{w}(k) = \eta(k)\nabla J(\boldsymbol{w}(k)) \quad (5)$$

## We have then

$$\begin{aligned} J(\boldsymbol{w}(k+1)) \cong & J(\boldsymbol{w}(k)) + \nabla J^T(-\eta(k)\nabla J(\boldsymbol{w}(k))) + ... \\ & \frac{1}{2}(-\eta(k)\nabla J(\boldsymbol{w}(k)))^T \boldsymbol{H}(-\eta(k)\nabla J(\boldsymbol{w}(k))) \end{aligned}$$

# Then

**We substitute (Eq. 1) into (Eq. 4)**

$$\boldsymbol{w}(k+1) - \boldsymbol{w}(k) = \eta(k)\nabla J(\boldsymbol{w}(k)) \tag{5}$$

**We have then**

$$\begin{aligned} J(\boldsymbol{w}(k+1)) \cong & J(\boldsymbol{w}(k)) + \nabla J^T(-\eta(k)\nabla J(\boldsymbol{w}(k))) + ... \\ & \frac{1}{2}(-\eta(k)\nabla J(\boldsymbol{w}(k)))^T \boldsymbol{H}(-\eta(k)\nabla J(\boldsymbol{w}(k))) \end{aligned}$$

**Finally, we have**

$$J(\boldsymbol{w}(k+1)) \cong J(\boldsymbol{w}(k)) - \eta(k)\|\nabla J\|^2 + \frac{1}{2}\eta^2(k)\nabla J^T \boldsymbol{H}\nabla J \tag{6}$$

# Derive with respect to $\eta(k)$ and make the result equal to zero

## We have then

$$-\|\nabla J\|^2 + \eta(k)\nabla J^T \boldsymbol{H} \nabla J = 0 \tag{7}$$

## Derive with respect to $\eta(k)$ and make the result equal to zero

**We have then**

$$-\left\|\nabla J\right\|^2 + \eta(k)\,\nabla J^T \boldsymbol{H} \nabla J = 0 \tag{7}$$

**Finally**

$$\eta(k) = \frac{\left\|\nabla J\right\|^2}{\nabla J^T \boldsymbol{H} \nabla J} \tag{8}$$

Remark This is the optimal step size!!!

# Derive with respect to $\eta(k)$ and make the result equal to zero

**We have then**

$$-\|\nabla J\|^2 + \eta(k)\nabla J^T \boldsymbol{H} \nabla J = 0 \tag{7}$$

**Finally**

$$\eta(k) = \frac{\|\nabla J\|^2}{\nabla J^T \boldsymbol{H} \nabla J} \tag{8}$$

Remark This is the optimal step size!!!

**Problem!!!**

Calculating $\boldsymbol{H}$ can be quite expansive!!!

# We can have an adaptive linear search!!!

## We can use the idea of having everything fixed, but $\eta(k)$

Then, we can have the following function
$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$

# We can have an adaptive linear search!!!

## We can use the idea of having everything fixed, but $\eta(k)$

Then, we can have the following function

$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$

- We can optimized using linear search methods

# We can have an adaptive linear search!!!

## We can use the idea of having everything fixed, but $\eta(k)$

Then, we can have the following function
$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$

- We can optimized using linear search methods

## Linear Search Methods

- Backtracking linear search

# We can have an adaptive linear search!!!

## We can use the idea of having everything fixed, but $\eta(k)$

Then, we can have the following function

$$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$$

- We can optimized using linear search methods

## Linear Search Methods

- Backtracking linear search
- Bisection method

# We can have an adaptive linear search!!!

## We can use the idea of having everything fixed, but $\eta(k)$

Then, we can have the following function
$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$

- We can optimized using linear search methods

## Linear Search Methods

- Backtracking linear search
- Bisection method
- Golden ratio

# We can have an adaptive linear search!!!

**We can use the idea of having everything fixed, but $\eta(k)$**

Then, we can have the following function
$f(\eta(k)) = J(\boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)))$

- We can optimized using linear search methods

**Linear Search Methods**

- Backtracking linear search
- Bisection method
- Golden ratio
- Etc.

# Please Take a Look

**For more, please read the paper**

"SEQUENTIAL MINIMAX SEARCH FOR A MAXIMUM" by J. Kiefer

# There are better versions

**Take a look**

The papers at the repository.

# Outline

# Shrinkage Methods

### By retaining a subset of the predictors and discarding the rest

- Subset Selection produces a model that is interpretable,

# Shrinkage Methods

**By retaining a subset of the predictors and discarding the rest**

- Subset Selection produces a model that is interpretable,
- It possibly produces lower prediction error than the full model.

# Shrinkage Methods

**By retaining a subset of the predictors and discarding the rest**

- Subset Selection produces a model that is interpretable,
- It possibly produces lower prediction error than the full model.

**However given process**

- it often exhibits high variance,

# Shrinkage Methods

### By retaining a subset of the predictors and discarding the rest

- Subset Selection produces a model that is interpretable,
- It possibly produces lower prediction error than the full model.

### However given process

- it often exhibits high variance,
- It does not reduce the prediction error of the full model.

# Shrinkage Methods

**By retaining a subset of the predictors and discarding the rest**

- Subset Selection produces a model that is interpretable,
- It possibly produces lower prediction error than the full model.

**However given process**

- it often exhibits high variance,
- It does not reduce the prediction error of the full model.

**Therefore**

- Shrinkage methods are more continuous avoiding high variability.

# Outline

# The house example

## Imagine the following data set

# Now assume that we use LSE

## For the fitting

$$\frac{1}{2}\sum_{i=1}^{N}\left(h_{\boldsymbol{w}}\left(x_i\right)-y_i\right)^2$$

# Now assume that we use LSE

## For the fitting

$$\frac{1}{2}\sum_{i=1}^{N}\left(h_{\boldsymbol{w}}\left(x_i\right)-y_i\right)^2$$

## We can then run one of our machine to see what minimize better the previous equation

Question: Did you notice that I did not impose any structure to $h_{\boldsymbol{w}}\left(x\right)$?

# Then, First fitting

**What about using $h_1(x) = w_0 + w_1 x + w_2 x^2$?**

# Second fitting

**What about using $h_2(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$?**

# Therefore, we have a problem

**We get weird overfitting effects!!!**

What do we do? What about minimizing the influence of $w_3, w_4, w_5$?

# Therefore, we have a problem

**We get weird overfitting effects!!!**

What do we do? What about minimizing the influence of $w_3, w_4, w_5$?

**How do we do that?**

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2$$

What about integrating those values to the cost function? Ideas

# Outline

# We have

**Regularization intuition is as follow**

Small values for parameters $w_0, w_1, w_2, ..., w_n$

# We have

## Regularization intuition is as follow

Small values for parameters $w_0, w_1, w_2, ..., w_n$

## It implies

1. "Simpler" function
2. Less prone to overfitting

# We can do the previous idea for the other parameters

## We can do the same for the other parameters

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \sum_{i=1}^{d} \lambda_i w_i^2 \tag{9}$$

# We can do the previous idea for the other parameters

**We can do the same for the other parameters**

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \sum_{i=1}^{d} \lambda_i w_i^2 \quad (9)$$

**However handling such many parameters can be so difficult**

Combinatorial problem in reality!!!

# Better, we can

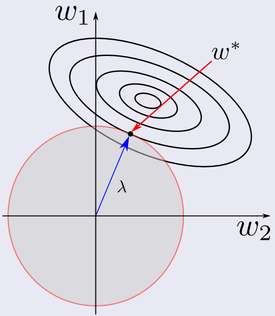
**We better use the following**

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \lambda \sum_{i=1}^{d} w_i^2 \tag{10}$$

# Graphically

## Geometrically Equivalent to

$$\sum_{i=1}^{N}\left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d+1} w_i^2$$



# Outline

# Ridge Regression

**Equation**

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\{ \sum_{i=1}^{N} \left( y_i - w_0 - \sum_{j-1}^{d} x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^{d} w_j^2 \right\}$$

# Ridge Regression

## Equation

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\{ \sum_{i=1}^{N} \left( y_i - w_0 - \sum_{j-1}^{d} x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^{d} w_j^2 \right\}$$

## Here

- $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage

# Therefore

**The Larger $\lambda \geq 0$**

- The coefficients are shrunk toward zero (and each other).

# Therefore

## The Larger $\lambda \geq 0$

- The coefficients are shrunk toward zero (and each other).

## This is also used in Neural Networks

- where it is known as weight decay

# This is also can be written

## Optimization Solution

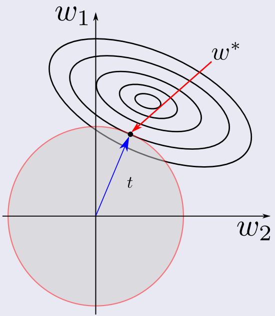
$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - w_0 - \sum_{j-1}^{d} x_{ij} w_j \right)^2$$

subject to $\sum_{j=1}^{d} w_j^2 < t$

# Graphically

## Geometrically Equivalent to

$$\arg\min \; \sum_{i=1}^{N} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2$$
$$\text{subject to } \sum_{i=1}^{d+1} w_i^2 < t$$



# Outline

# Least Absolute Shrinkage and Selection Operator (LASSO)

It was introduced by Robert Tibshirani in 1996 based on Leo Breiman's nonnegative garrote

$$\hat{\boldsymbol{w}}^{garrote} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} x_{ij} w_j \right)^2 + N\lambda \sum_{j=1}^{d} w_j$$

s.t. $w_j > 0 \; \forall j$

# Least Absolute Shrinkage and Selection Operator (LASSO)

## It was introduced by Robert Tibshirani in 1996 based on Leo Breiman's nonnegative garrote

$$\widehat{\boldsymbol{w}}^{garrote} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} x_{ij} w_j \right)^2 + N\lambda \sum_{j=1}^{d} w_j$$

$$\text{s.t. } w_j > 0 \; \forall j$$

## This is quite derivable

However, Tibshirani realized that you could get a more flexible model by using the absolute value at the constraint!!!

# Least Absolute Shrinkage and Selection Operator (LASSO)

## It was introduced by Robert Tibshirani in 1996 based on Leo Breiman's nonnegative garrote

$$\widehat{\boldsymbol{w}}^{garrote} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} x_{ij} w_j \right)^2 + N\lambda \sum_{j=1}^{d} w_j$$

s.t. $w_j > 0 \; \forall j$

## This is quite derivable

However, Tibshirani realized that you could get a more flexible model by using the absolute value at the constraint!!!

## Robert Tibshirani proposed the use of the $L_1$ norm

$$\|\boldsymbol{w}\|_1 = \sum_{i=1}^{d} |w_i|$$

# The Final Optimization Problem

## LASSO

$$\widehat{\boldsymbol{w}}^{LASSO} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} x_{ij} w_j \right)^2$$

$$\text{s.t. } \sum_{i=1}^{d} |w_i| \leq t$$

# The Final Optimization Problem

## LASSO

$$\widehat{\boldsymbol{w}}^{LASSO} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{d} x_{ij} w_j \right)^2$$

$$\text{s.t.} \sum_{i=1}^{d} |w_i| \leq t$$

## This is not derivable

- More advanced methods are necessary to solve this problem!!!

# Outline

# Lagrange Multipliers

## The method of Lagrange multipliers

- It gives a set of necessary conditions to identify optimal points of equality constrained optimization problems.

# Lagrange Multipliers

## The method of Lagrange multipliers

- It gives a set of necessary conditions to identify optimal points of equality constrained optimization problems.

## This is done by converting a constrained problem to an equivalent unconstrained problem

- with the help of certain unspecified parameters known as Lagrange multipliers.

# Lagrange Multipliers

## The classical problem formulation

$$\min\ f\left(x_1,...,x_n\right)$$
$$s.t\ h_1\left(x_1,...,x_n\right)=0$$

# Lagrange Multipliers

## The classical problem formulation

$$\begin{aligned} &\min \ f(x_1, ..., x_n) \\ &s.t \ h_1(x_1, ..., x_n) = 0 \end{aligned}$$

## It can be converted into

$$\min \ L(x_1, ..., x_n, \lambda) = \min \{f(x_1, ..., x_n) - \lambda h_1(x_1, ..., x_n)\}$$

# Lagrange Multipliers

## The classical problem formulation

$$\min \; f(x_1, ..., x_n)$$
$$s.t \; h_1(x_1, ..., x_n) = 0$$

## It can be converted into

$$\min \;\; L(x_1, ..., x_n, \lambda) = \min \{f(x_1, ..., x_n) - \lambda h_1(x_1, ..., x_n)\}$$

## where

- $L(\mathbf{x}, \lambda)$ is the Lagrangian function.
- $\lambda$ is an unspecified positive or negative constant called the **Lagrange Multiplier.**

# Finding an Optimum using Lagrange Multipliers

**New problem**

$$\min \; L(x_1, ..., x_n, \lambda) = \min \{ f(x_1, ..., x_n) - \lambda h_1(x_1, ..., x_n) \}$$

# Finding an Optimum using Lagrange Multipliers

**New problem**

$$\min \; L(x_1, ..., x_n, \lambda) = \min \{ f(x_1, ..., x_n) - \lambda h_1(x_1, ..., x_n) \}$$

**We want a $\lambda = \lambda^*$ optimal**

If the minimum of $L(x_1, ..., x_n, \lambda^*)$ occurs at

$$(x_1, x_2, ..., x_n)^T = (x_1, x_2, ..., x_n)^{T*}$$

# Therefore

$(x_1, ..., x_n)^{T*}$ satisfies $h_1(x_1, ..., x_n) = 0$, then $(x_1, ..., x_n)^{T*}$ minimizes

$$\min\ f(x_1, ..., x_n)$$
$$s.t\ h_1(x_1, ..., x_n) = 0$$

# Therefore

**$(x_1, ..., x_n)^{T*}$ satisfies $h_1(x_1, ..., x_n) = 0$, then $(x_1, ..., x_n)^{T*}$ minimizes**

$$\begin{aligned} &\min \ f(x_1, ..., x_n) \\ &s.t \ h_1(x_1, ..., x_n) = 0 \end{aligned}$$

**Trick**

- It is to find appropriate value for Lagrangian multiplier $\lambda$.

# Remember

**Think about this**

Remember First Law of Newton!!!

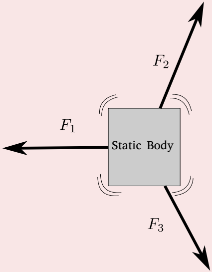
# Remember

## Think about this

Remember First Law of Newton!!!

## Yes!!!

**A system in equilibrium does not move**



# Lagrange Multipliers

**Definition**

Gives a set of necessary conditions to identify optimal points of equality constrained optimization problem

# Lagrange was a Physicists

## He was thinking in the following formula

A system in equilibrium has the following equation:

$$F_1 + F_2 + ... + F_K = 0 \tag{11}$$

# Lagrange was a Physicists

## He was thinking in the following formula

A system in equilibrium has the following equation:

$$F_1 + F_2 + ... + F_K = 0 \tag{11}$$

## But functions do not have forces?

Are you sure?

# Lagrange was a Physicists

## He was thinking in the following formula

A system in equilibrium has the following equation:

$$F_1 + F_2 + ... + F_K = 0 \tag{11}$$

## But functions do not have forces?

Are you sure?

## Think about the following

The Gradient of a surface.

# Gradient to a Surface

**After all a gradient is a measure of the maximal change**

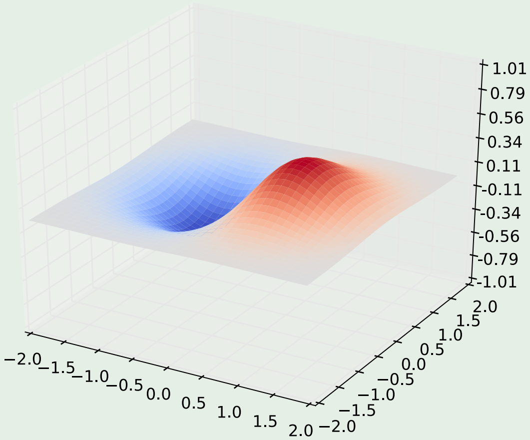
For example the gradient of a function of three variables:

$$\nabla f(\boldsymbol{x}) = i\frac{\partial f(\boldsymbol{x})}{\partial x} + j\frac{\partial f(\boldsymbol{x})}{\partial y} + k\frac{\partial f(\boldsymbol{x})}{\partial z} \tag{12}$$

where $i$, $j$ and $k$ are unitary vectors in the directions $x$, $y$ and $z$.

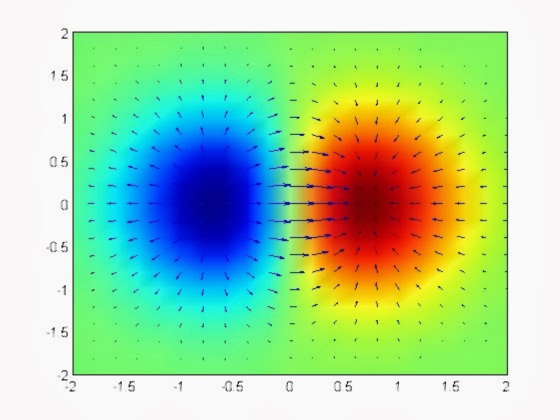
# Example

We have $f(x, y) = x \exp\{-x^2 - y^2\}$



# Example

## With Gradient at the the contours when projecting in the 2D plane




100

# Now, Think about this

**Yes, we can use the gradient**

However, we need to do some scaling of the forces by using parameters $\lambda$

# Now, Think about this

**Yes, we can use the gradient**

However, we need to do some scaling of the forces by using parameters $\lambda$

**Thus, we have**

$$F_0 + \lambda_1 F_1 + ... + \lambda_K F_K = 0 \tag{13}$$

where $F_0$ is the gradient of the principal cost function and $F_i$ for $i = 1, 2, .., K$.

## Thus

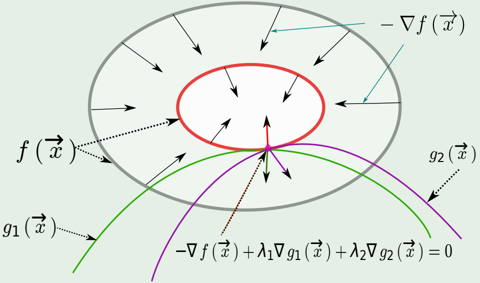
If we have the following optimization:

$$
\begin{aligned}
\min \;\; & f(\boldsymbol{x}) \\
s.t g_1(\boldsymbol{x}) &= 0 \\
g_2(\boldsymbol{x}) &= 0
\end{aligned}
$$

# Geometric interpretation in the case of minimization

## What is wrong? Gradients are going in the other direction, we can fix by simple multiplying by -1

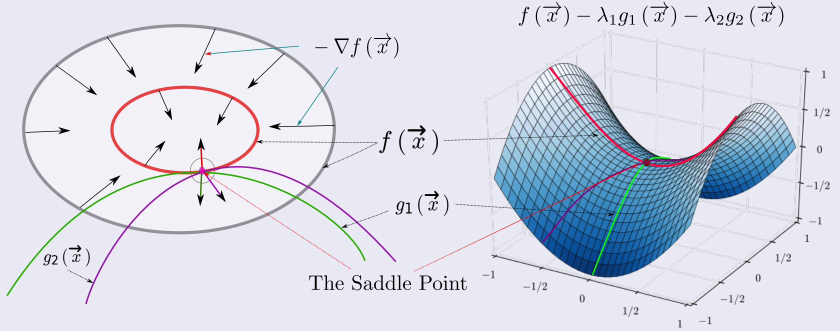
Here the cost function is $f(x, y) = x \exp\{-x^2 - y^2\}$ we want to minimize




Nevertheless: it is equivalent to $\nabla f(\vec{x}) - \lambda_1 \nabla g_1(\vec{x}) - \lambda_2 \nabla g_2(\vec{x}) = 0$

# Basically, we convert the problem into a one looking for a **Saddle Point**

## At the left the original problem, at the right the Lagrangian!!!



# Yes!!!

## Basically

- We convert the minimization or maximization of a convex or concave section of a function living in a constrained environment!!!

# Outline

# Method

## Steps

1. Original problem is rewritten as:
    1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$

# Method

## Steps

1. Original problem is rewritten as:
   1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$
2. Take derivatives of $L(\boldsymbol{x}, \lambda)$ with respect to $x_i$ and set them equal to zero.

# Method

## Steps

1. Original problem is rewritten as:
    1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$
2. Take derivatives of $L(\boldsymbol{x}, \lambda)$ with respect to $x_i$ and set them equal to zero.
$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d} |w_i| \tag{14}$$
    1. Here you need to use a soft version of the absolute value
3. Express all $x_i$ in terms of Lagrangian multiplier $\lambda$.

# Method

## Steps

1. Original problem is rewritten as:
   1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$
2. Take derivatives of $L(\boldsymbol{x}, \lambda)$ with respect to $x_i$ and set them equal to zero.

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d} |w_i| \tag{14}$$

   1. Here you need to use a soft version of the absolute value
3. Express all $x_i$ in terms of Lagrangian multiplier $\lambda$.
4. Plug $\boldsymbol{x}$ in terms of $\lambda$ in constraint $h_1(\boldsymbol{x}) = 0$ and solve $\lambda$.

# Method

## Steps

1. Original problem is rewritten as:
    1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$
2. Take derivatives of $L(\boldsymbol{x}, \lambda)$ with respect to $x_i$ and set them equal to zero.

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d} |w_i| \tag{14}$$

    1. Here you need to use a soft version of the absolute value
3. Express all $x_i$ in terms of Lagrangian multiplier $\lambda$.
4. Plug $\boldsymbol{x}$ in terms of $\lambda$ in constraint $h_1(\boldsymbol{x}) = 0$ and solve $\lambda$.
5. Calculate $\boldsymbol{x}$ by using the just found value for $\lambda$.

# Method

## Steps

1. Original problem is rewritten as:
   1. minimize $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda h_1(\boldsymbol{x})$
2. Take derivatives of $L(\boldsymbol{x}, \lambda)$ with respect to $x_i$ and set them equal to zero.

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d} |w_i| \tag{14}$$

   1. Here you need to use a soft version of the absolute value
3. Express all $x_i$ in terms of Lagrangian multiplier $\lambda$.
4. Plug $\boldsymbol{x}$ in terms of $\lambda$ in constraint $h_1(\boldsymbol{x}) = 0$ and solve $\lambda$.
5. Calculate $\boldsymbol{x}$ by using the just found value for $\lambda$.

## From the step 2

If there are $n$ variables (i.e., $x_1, \cdots, x_n$) then you will get $n$ equations with $n+1$ unknowns (i.e., $n$ variables $x_i$ and one Lagrangian multiplier $\lambda$).

# Example

**We can apply that to the following problem**

$$
\begin{aligned}
\min \;\; & f(x,y) = x^2 - 8x + y^2 - 12y + 48 \\
s.t \;\; & x + y = 8
\end{aligned}
$$

# Outline

# The Lagrangian Version

**The Lagrangian**

$$\widehat{\boldsymbol{w}}^{LASSO} = \arg\min_{\boldsymbol{w}} \left\{ \sum_{i=1}^{N} \left( y_i - \boldsymbol{x}^T \boldsymbol{w} \right)^2 + \lambda \sum_{i=1}^{d} |w_i| \right\}$$

# The Lagrangian Version

## The Lagrangian

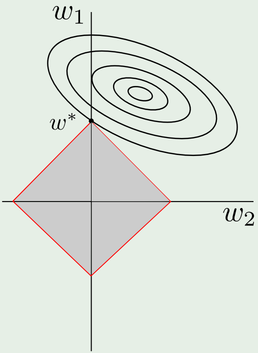
$$\widehat{\boldsymbol{w}}^{LASSO} = \arg\min_{\boldsymbol{w}} \left\{ \sum_{i=1}^{N} \left( y_i - \boldsymbol{x}^T \boldsymbol{w} \right)^2 + \lambda \sum_{i=1}^{d} |w_i| \right\}$$

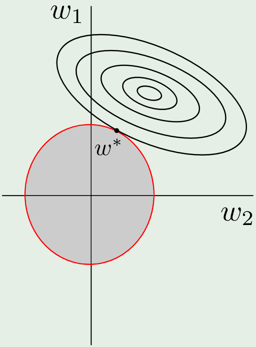
## However

- You have other regularizations as $\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$

# Graphically

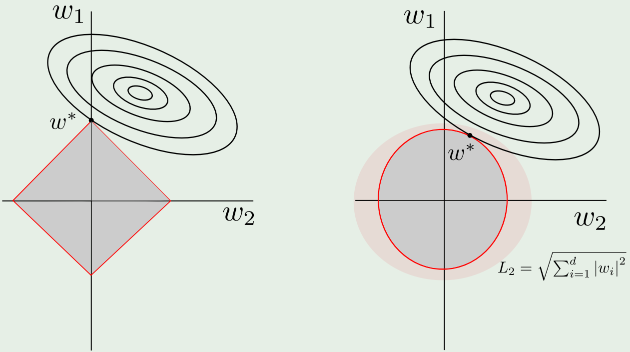
**The first area correspond to the $L_1$ regularization and the second one?**



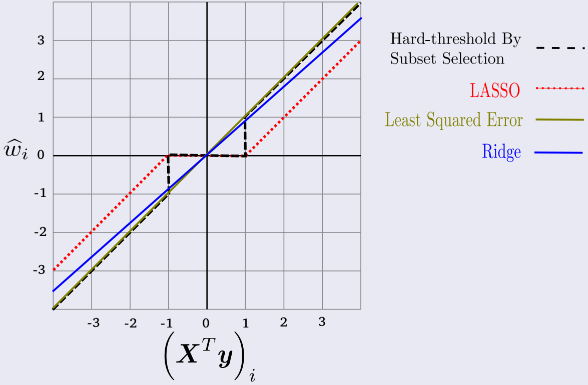
# Graphically

**Yes the circle defined as $\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$**



# For Example

## In the Case of $\boldsymbol{X}$ is a Orthogonal Matrix



# The seminal paper by Robert Tibshirani

**An initial study of this regularization can be seen in**

"Regression Shrinkage and Selection via the LASSO" by Robert Tibshirani - 1996

# This out the scope of this class

**However, it is worth noticing that the most efficient method for solving LASSO problems is**

"Pathwise Coordinate Optimization" By Jerome Friedman, Trevor Hastie, Holger Ho and Robert Tibshirani

# This out the scope of this class

**However, it is worth noticing that the most efficient method for solving LASSO problems is**

"Pathwise Coordinate Optimization" By Jerome Friedman, Trevor Hastie, Holger Ho and Robert Tibshirani

**Nevertheless**

It will be a great seminar paper!!!

# Outline

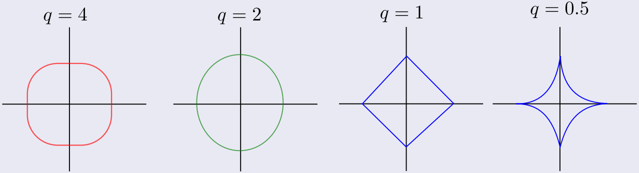
# Furthermore

**We can generalize ridge regression and the lasso, and view them as Bayes estimates**

$$\hat{\boldsymbol{w}}^{LASSO} = \arg\min_{\boldsymbol{w}} \left\{ \sum_{i=1}^{N} \left( y_i - \boldsymbol{x}^T \boldsymbol{w} \right)^2 + \lambda \sum_{i=1}^{d} |w_i|^q \right\} \text{ with } q \geq 0$$

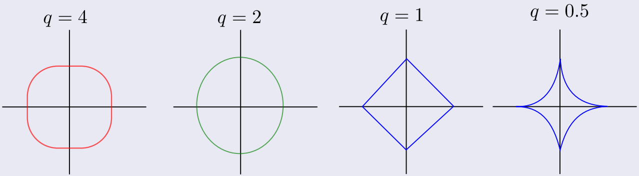
# For Example

**We have when $d = 2$**



# For Example

## We have when $d = 2$




## Here, when $q > 1$

- You are having a derivable Lagrangian, but you lose the LASSO properties

# Therefore

**Zou and Hastie (2005) introduced the elastic- net penalty**

$$\lambda \sum_{i=1}^{d} \left\{ \alpha w_i^2 + (1-\alpha)\left| w_i \right| \right\}$$

# Therefore

## Zou and Hastie (2005) introduced the elastic- net penalty

$$\lambda \sum_{i=1}^{d} \left\{ \alpha w_i^2 + (1-\alpha)\left| w_i \right| \right\}$$

## This is Basically

- A Compromise Between the Ridge and LASSO.