CSCI5654 (Linear Programming, Fall 2013)

Lectures 10-12

Today's Lecture

1. Introduction to norms: L_1, L_2, L_∞.
2. Casting absolute value and max operators.
3. Norm minimization problems.
4. Applications: fitting, classification, denoising etc..

Norm

Basic idea: Notion of length/distance between vector.

Definition (Norm): Let $\ell : X \mapsto \mathcal{R}^{\geq 0}$ be a mapping from a vector space to non-negative reals. The function ℓ is a norm iff

1. $\ell(\vec{x}) = 0$ iff 0,
2. $\ell(a\vec{x}) = |a|\ell(\vec{x})$
3. $\ell(\vec{x} + \vec{y}) \leq \ell(\vec{x}) + \ell(\vec{y})$ (**Triangle Inequality**).

"Length" of $\vec{x}$: $\ell(\vec{x})$.

"Distance" between $\vec{x}, \vec{y}$ is $\ell(\vec{y} - \vec{x})(= \ell(\vec{x} - \vec{y}))$.

L_2 (Euclidean) norm

Distance between (x_1, y_1) to (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
In general, $||\vec{x}||_2 = \sqrt{\vec{x}^{\mathrm{T}}.\vec{x}}$.
Distance between $\vec{x}, \vec{y}$ is given by $||\vec{x} - \vec{y}||_2$.

L_p norm

For $p \geq 1$, we define L_p norm as

$$||\vec{x}||_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}} .$$

The L_p norm distance between two vectors is: $||\vec{x} - \vec{y}||_p$
L_1 **norm:** $||\vec{x}||_1 = |x_1| + |x_2| + \ldots + |x_n|$.
Class exercise: verify that L_1 norm distance is indeed a norm.
Q: Why is $p \geq 1$ necessary?

L_∞ norm

Obtained as the limit $lim_{p \to \infty} \; ||\vec{x}||_p$.
Definition: For vector $\vec{x}$, $||\vec{x}||_\infty$ is defined as

$$||\vec{x}||_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|) \, .$$

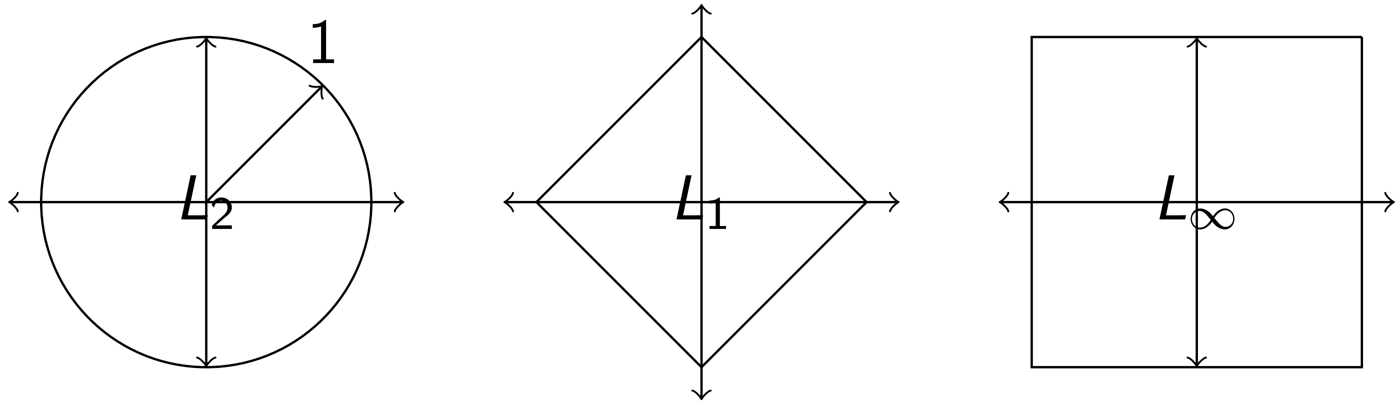
Ex-1: Can we verify that L_∞ is really a norm?
Ex-2 (challenge): Prove that the limit definition coincides with the explicit form of the L_∞ distance.

Norm: Exercise

Take vectors $\vec{x} = (0, 1, -1), \vec{y} = (2, 1, 1)$.
Write down L_1, L_2, L_∞ norm distances between $\vec{x}, \vec{y}$.

Norm Spheres

ϵ-**Sphere:** $\{\vec{x} \mid d(\vec{x}, 0) \leq \epsilon\}$ for norm d. The 1−spheres corresponding to the norms L_1, L_∞, L_2.



Note: Norm spheres are convex sets.

Norm Minimization problems

General form:

$$\begin{array}{rcl} \text{minimize} & ||x||_p & \\ & Ax & \leq & b \end{array}$$

Note-1: We always minimize the norm functions (in this course).
Note-2: <u>Maximization</u> of norm is generally a hard (non-convex) problem.

Unconstrained Norm Minimization

Lets first study the following problem:

$$\min. \ ||Ax - b||_p \ .$$

1. For $p = 2$, this problem is called (unconstrained) least squares.
   We will solve this using calculus.
2. For $p = 1, \infty$, this problem is called $L_1 (L_\infty)$ least squares.
   We will reduce this problem to LP.
3. Applications: solving (noisy) linear systems, regularization, denoising, max. likelihood estimation, regression and so on.

Unconstrained Least Squares

Decision Variables: $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n$.
Objective function: min.$||A\vec{x} - \vec{b}||_2$.
Constraints: No explicit constraint.

Least Squares: Example

Example: $\min. ||(2x + 3y - 1, y)||_2$.
Solution: We will equivalently minimize $(2x + 3y - 1)^2 + y^2$. Using fundamental theorem of calculus, we obtain the conditions:

$$\begin{aligned} \frac{\partial (2x+3y-1)^2 + y^2}{\partial x} &= 0 \\ \frac{\partial (2x+3y-1)^2 + y^2}{\partial y} &= 0 \end{aligned}$$

In other words, we obtain:

$$\begin{aligned} 4(2x + 3y - 1) &= 0 \\ 6(2x + 3y - 1) + 2y &= 0 \end{aligned}$$

The optima lies at is $y = 0, x = \frac{1}{2}$.
Verify minima by computing checking second order derivative (Hessian matrix).

Unconstrained Least Squares

Problem: minimize $||A\vec{x} - \vec{b}||_2$.
Solution: We will use calculus minimum finding using partial derivatives.
Recall:

$$\nabla f(x_1, \ldots, x_n) = (\partial_{x_1} f, \partial_{x_2} f, \ldots, \partial_{x_n} f)\,.$$

Criterion for optimal point for $f(\vec{x})$ is that $\nabla f = (0, \ldots, 0)$.

$$\begin{aligned}
\nabla(||A\vec{x} - \vec{b}||_2^2) &= \nabla((A\vec{x} - \vec{b})^{\mathrm{T}}(A\vec{x} - \vec{b})) \\
&= \nabla(\vec{x}^{\mathrm{T}} A^{\mathrm{T}} A\vec{x} - 2\vec{x}^{\mathrm{T}} A^{\mathrm{T}} \vec{b} + \vec{b}^{\mathrm{T}} \vec{b}) \\
&= 2A^{\mathrm{T}} A\vec{x} - 2A^{\mathrm{T}} \vec{b} \\
&= 0
\end{aligned}$$

Minima will occur at $\vec{x}^* = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \vec{b}$ (assume A is "full rank").
Similar solution for L_p norm for even number p.

L_1 norm minimization

Decision Variables: $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n$.
Objective function: min.$||A\vec{x} - \vec{b}||_1$.
Constraints: No explicit constraint.
<u>We will reduce this to a linear program</u>.

Example

Example: min.$||(2x+3y-1, y)||_1 = |2x+3y-1| + |y|$.
Trick: Let $t_1 \geq 0, t_2 \geq 0$ be s.t.

$$\begin{array}{rcl} |2x+3y-1| & \leq & t_1 \\ |y| & \leq & t_2 \end{array}$$

Consider the following problem:

$$\begin{array}{llcl} \text{min.} & t_1 + t_2 & & \\ & |2x+3y-1| & \leq & t_1 \\ & |y| & \leq & t_2 \\ & t_1, t_2 & \geq & 0 \end{array}$$

Goal: convince class that this problem is equivalent to original.
Note: $|y| \leq t_2$ can be rewritten as $y \leq t_2 \ \wedge \ -y \leq t_2$.

Example: LP formulation

$$
\begin{array}{lrcl}
\text{min.} & t_1 + t_2 & & \\
& 2x + 3y - 1 & \leq & t_1 \\
& -2x - 3y + 1 & \leq & t_1 \\
& y & \leq & t_2 \\
& -y & \leq & t_2 \\
& t_1, t_2 & \geq & 0
\end{array}
$$

Solution: $x = \frac{1}{2}, y = 0$.

Q: Can we repeat the same trick if $|y| \geq t_2$?

N: No. A disjunction of inequalities will be needed.

L_1 norm minimization

Problem: min. $||Ax - b||_1$.

Solution: Let A be $m \times n$ matrix. Add variables $t_1, \ldots, t_m$ correspoding to rows of m.

$$\left[\begin{array}{llll} \text{min.} & \sum_{i=1}^{m} t_i & & \\ & |A_i\vec{x} - \vec{b}| & \leq & t_i \\ & t_1, \ldots, t_m & \geq & 0 \end{array}\right] \Rightarrow \left[\begin{array}{llll} \text{min.} & \sum_{i=1}^{m} t_i & & \\ & A\vec{x} - \vec{b} & \leq & \vec{t} \\ & -A\vec{x} + \vec{b} & \leq & \vec{t} \\ & t_1, \ldots, t_m & \geq & 0 \end{array}\right]$$

We write it in the general form:

$$\begin{array}{llll} \text{max.} & -\vec{1}^{\mathrm{T}}\vec{t} & & \\ & A\vec{x} - \vec{t} & \leq & \vec{b} \\ & -A\vec{x} - \vec{t} & \leq & -\vec{b} \\ & t_1, \ldots, t_m & \geq & 0 \end{array}$$

L_∞ norm minimization

Decision Variables: $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n$.
Objective function: min.$||A\vec{x} - \vec{b}||_\infty$.
Constraints: No explicit constraint.
<u>We will reduce this to a linear program.</u>

Example

Example: min.$||(2x+3y-1, y)||_\infty =$ min max.$(|2x+3y-1|, |y|)$.
Trick: Let $t \geq 0$ be s.t.

$$\begin{array}{rcl} |2x+3y-1| & \leq & t \\ |y| & \leq & t \end{array}$$

Consider the following problem:

$$\begin{array}{llcl} \text{min.} & t & & \\ & |2x+3y-1| & \leq & t \\ & |y| & \leq & t \\ & t & \geq & 0 \end{array}$$

Goal: convince class that this problem is equivalent to original.
Note: $|y| \leq t$ can be rewritten as $-t \leq y \leq t$.

Example:LP formulation

$$
\begin{array}{lrcl}
\text{min.} & t & & \\
& 2x + 3y - 1 & \leq & t \\
& -2x - 3y + 1 & \leq & t \\
& y & \leq & t \\
& -y & \leq & t \\
& t & \geq & 0
\end{array}
$$

Solution: $x = \frac{1}{2}, y = 0$.

L_∞ norm minimization

Problem: min.$||Ax - b||_\infty$.
Solution: Let A be $m \times n$ matrix. Add variables $t_1, \ldots, t_m$ correspoding to rows of m.

$$\left[\begin{array}{lccc} \text{min.} & t & & \\ & |A_i\vec{x} - \vec{b}| & \leq & t \\ & t & \geq & 0 \end{array}\right] \Rightarrow \left[\begin{array}{lccc} \text{min.} & t & & \\ & A\vec{x} - \vec{b} & \leq & t.\vec{1} \\ & -A\vec{x} + \vec{b} & \leq & t.\vec{1} \\ & t & \geq & 0 \end{array}\right]$$

We write it in the general form:

$$\begin{array}{lccc} \text{max.} & t & & \\ & A\vec{x} - t.\vec{1} & \leq & \vec{b} \\ & -A\vec{x} - t.\vec{1} & \leq & -\vec{b} \\ & t & \geq & 0 \end{array}$$

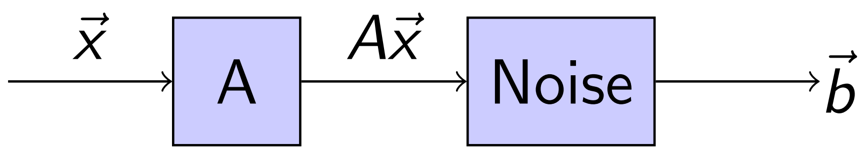
Application-1: Estimation of Solutions

Application-1: Solution Estimation

Problem: We are trying to solve the linear equation

$$A\vec{x} = \vec{b}$$

1. The entries in $A, \vec{b}$ could have random errors (noisy measurements).
2. There are many more equations than variables.



Application: Estimation

Example Problem: We are trying to measure a quantity x using noisy measurements:

$$\begin{aligned} 2x &= 5.1 \\ 3x &= 7.6 \\ 4x &= 9.91 \\ 5x &= 12.45 \\ 6x &= 15.1 \end{aligned}$$

Q-1: Is there a value of x that explains the measurement?
Q-2: What do we do in this case?

Application: Estimation

Example Problem: We are trying to measure a quantity x using noisy measurements:

$$\begin{aligned} 2x &= 5.1 \\ 3x &= 7.6 \\ 4x &= 9.91 \\ 5x &= 12.45 \\ 6x &= 15.1 \end{aligned}$$

Find x that nearly fits all the measurements., i,e,

$$\text{min.} ||(2x - 5.1, 3x - 7.6, 4x - 9.91, 5x - 12.45, 6x - 15.1)||_p \,.$$

We can choose $p = 1, 2, \infty$ and see what answer we get.

Application:Estimation

L_2 norm: Apply least squares for

$$A = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad \vec{b} = \begin{pmatrix} -5.1 \\ -7.6 \\ -9.91 \\ -12.45 \\ -15.1 \end{pmatrix}$$

Solving for $x = \mathrm{argmin}||Ax - \vec{b}||_2$, yields $x \sim 2.505$.
Residual error:

$$(0.09, 0.08, -.11, -0.08, -.06)\,.$$

Application:Estimation

L_1 norm: Reduce to linear program using

$$A = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}, \ \vec{b} = \begin{pmatrix} -5.1 \\ -7.6 \\ -9.91 \\ -12.45 \\ -15.1 \end{pmatrix}, .$$

Solution: $x = 2.49$.
Residual Error:

$$(-.12, -.13, -0.05, 0.0, .16) .$$

L_∞ norm: Reduce to linear program.
Solution: $x = 2.501$
Residual Error:

$$(-.1, -.1, .1, .05, -.1)$$

Experiment with Larger Matrices (matlab)

- L_1 norm: large number of entries with small residuals, spread is larger.
- L_2 norm: residuals look like a gaussian distribution.
- L_∞ norm: residuals look more uniform, smaller spread.

Matlab code is available, run your own experiments!

Regularized Approximation

Ordinary least squares can have poor performance (ill conditioning problem).

Tikhonov Regularized Approximation:

$$\|A\vec{x} - b\|_2 + \lambda \|\vec{x}\|_2 \ .$$

$\lambda \geq 0$ is a tuning parameter.

Note-1: This is the same as the following norm minimization:

$$\left\| \begin{bmatrix} A \\ \lambda^{\frac{1}{2}} I \end{bmatrix} \vec{x} - \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix} \right\|_2^2 .$$

Note-2: In general, we can also have regularized approximation using L_1, L_∞ and other norms.

Penalty Function Approximation

Problem: Solve

$$\text{minimize.}\phi(A\vec{x} - \vec{b}) \,.$$

where ϕ is a penalty function.

If $\phi = L_1, L_2, L_\infty$, this is exactly the same as norm minimization.

Note-1: In general, ϕ need not be a norm.
Note-2: ϕ is sometimes called a loss function.

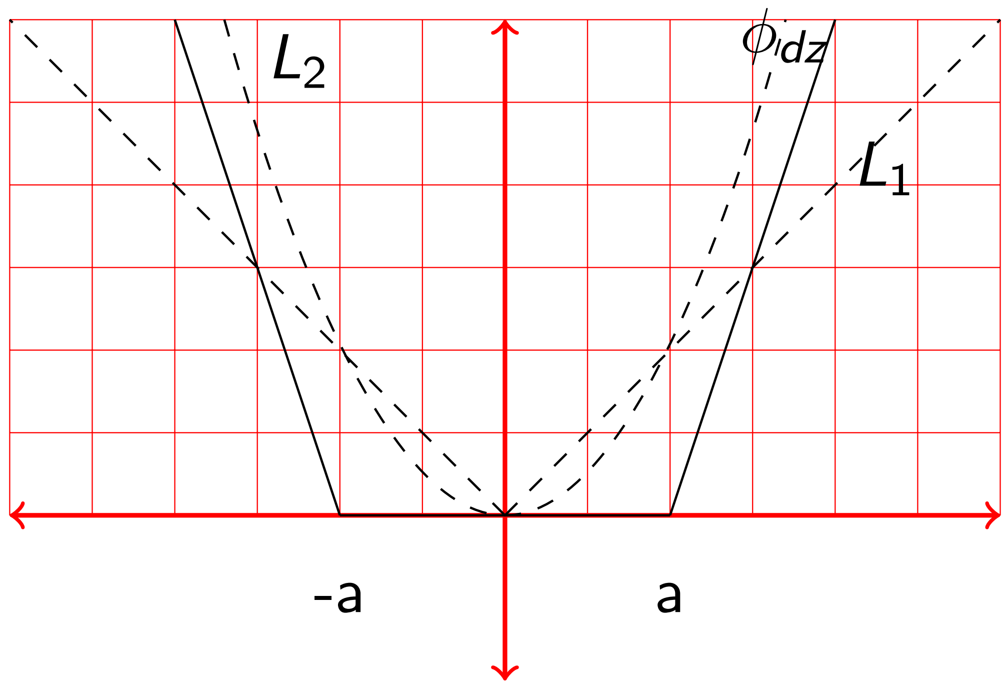
Penalty Functions: Example

Deadzone Linear Penalty: Let $\vec{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$. The deadzone linear penalty is sum of penalties on each individual component $x_1, \ldots, x_n$:

$$\phi_{dz}(\vec{x}) :\ \phi_{dz}(x_1) + \phi_{dz}(x_2) + \cdots + \phi_{dz}(x_n)\,,$$

wherein

$$\phi_{dz}(x) = \begin{cases} 0 & |u| \leq a \\ b(|u| - a) & |u| > a \end{cases}$$



Deadzone Linear Penalty (cont)

Deadzone vs. L_2:

- L_2 norm imposes large penalty when residual is high.
  Therefore, outliers in data can affect performance drastically.

- Deadzone penalty function is generally less sensitive to outliers.

Q: How do we solve the deadzone penalty approximation problem?
A: Apply tricks for L_1, L_∞ (upcoming assignment).

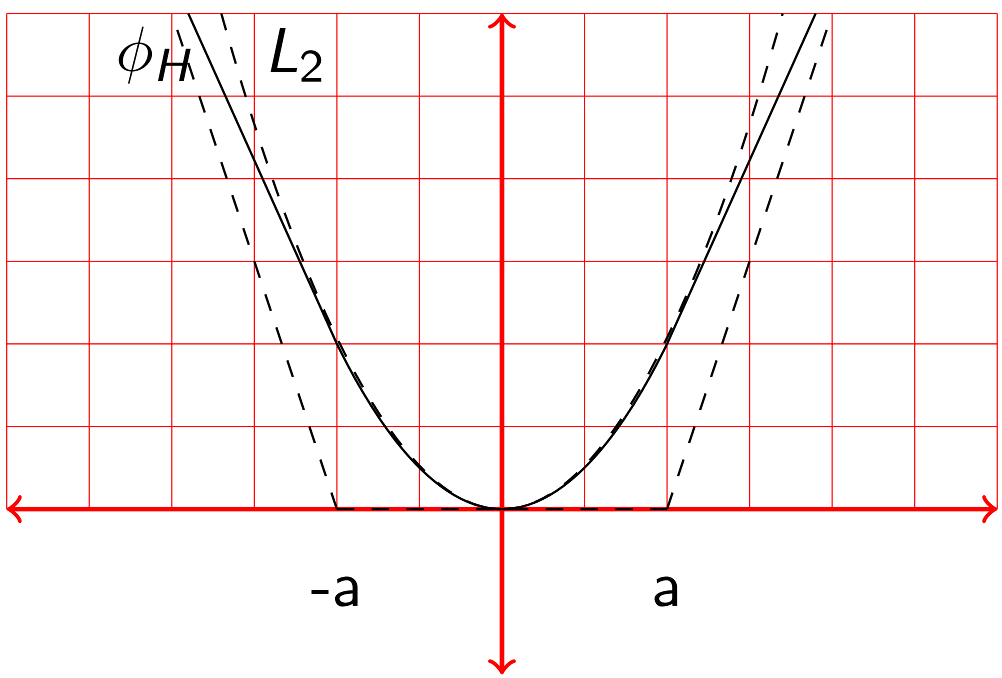
Other penalty functions

Huber Penalty: The Huber penalty is sum of functions on individual components $x_1, \ldots, x_n$:

$$\phi_H(x_1) + \phi_H(x_2) + \cdots + \phi_H(x_n)\,,$$

wherein

$$\phi_H(x) = \begin{cases} u^2 & |u| \leq a \\ a(2|u| - a) & |u| > a \end{cases}$$



Penalty Function Approximation: Summary

General observations:

- L_2: Natural, easy to solve (using pseudo inverse). However, sensitive to outliers.
- L_1: Sparse: ensures that most residuals are very small.
  Insensitive to outliers.
  However, spread of error is larger.
- L_∞ : Non-sparse but minimizes spread. Very sensitive to outliers.
- Other loss functions: provide various tradeoffs.

Details: Cf. Boyd's book.

Application-2: Signal Denoising

Application-2: Signal Denoising

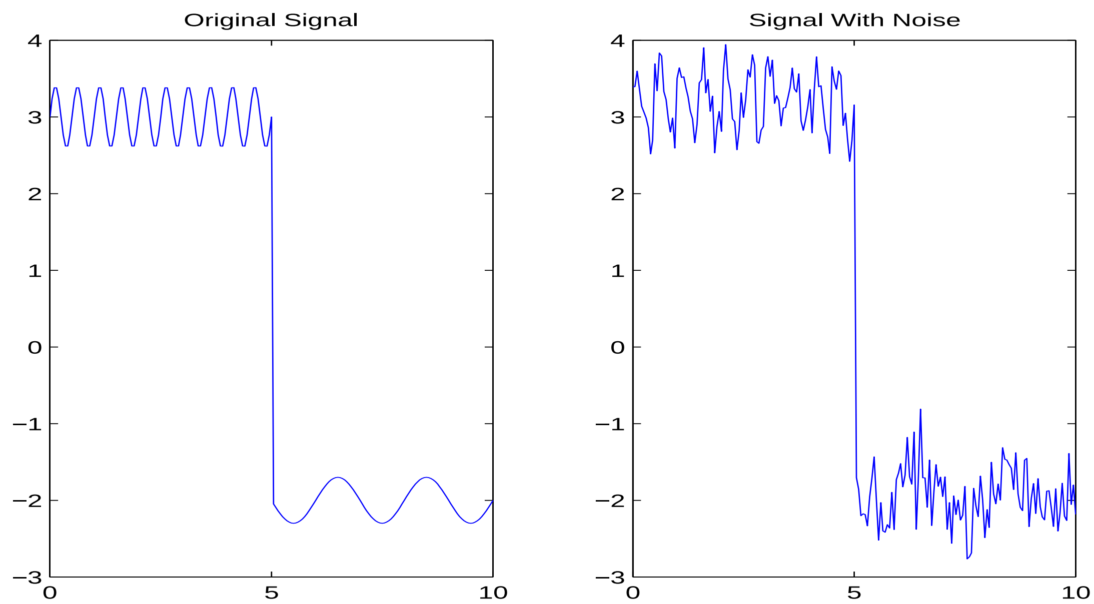
Input: Given samples of a noisy signal:

$$\vec{y} : (\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_N)^{\mathrm{T}} .$$

Original signal is unknown (very little data available).
Problem: Find new signal $\vec{y}^*$ by minimizing

$$\text{mininize. } ||\vec{y}^* - \vec{y}||_p^p + \phi_s(\vec{y}^*) .$$

ϕ_s is a <u>smoothing</u> function.



Smoothing Criteria

- Quadratic Smoothing: L_2 norm on successive differences.

$$\phi_q(\vec{y}) = \sum_{i=2}^{N} (y_i - y_{i-1})^2 \,.$$

- Total Variation Smoothing: L_1 norm on successive differences.

$$\phi_{tv}(\vec{y}) = \sum_{i=2}^{N} |y_i - y_{i-1}| \,.$$

- Maximum Variation Smoothing: L_∞ norm on successive differences.

$$\phi_{max}(\vec{y}) = \max_{i=2}^{N} (|y_{i+1} - y_i|) \,.$$

Quadratic Smoothing

Problem: minimize $\|\vec{y} - \vec{y}_n\|_2^2 + \phi_q(\vec{y})$.
This can be viewed as a least squares problem:

$$\text{minimize} \quad \left\| \begin{bmatrix} I \\ \Delta \end{bmatrix} \vec{y}^* - \begin{bmatrix} \vec{y}_n \\ 0 \end{bmatrix} \right\|_2^2 .$$

Where Δ is the matrix:

$$\Delta = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

Total/Maximum Variance Smoothing

Total variance smoothing: L_1 least squares problem.

$$\text{minimize} \quad \left\| \begin{bmatrix} I \\ \Delta \end{bmatrix} \vec{y}^{*} - \begin{bmatrix} \vec{y}_n \\ 0 \end{bmatrix} \right\|_1 .$$

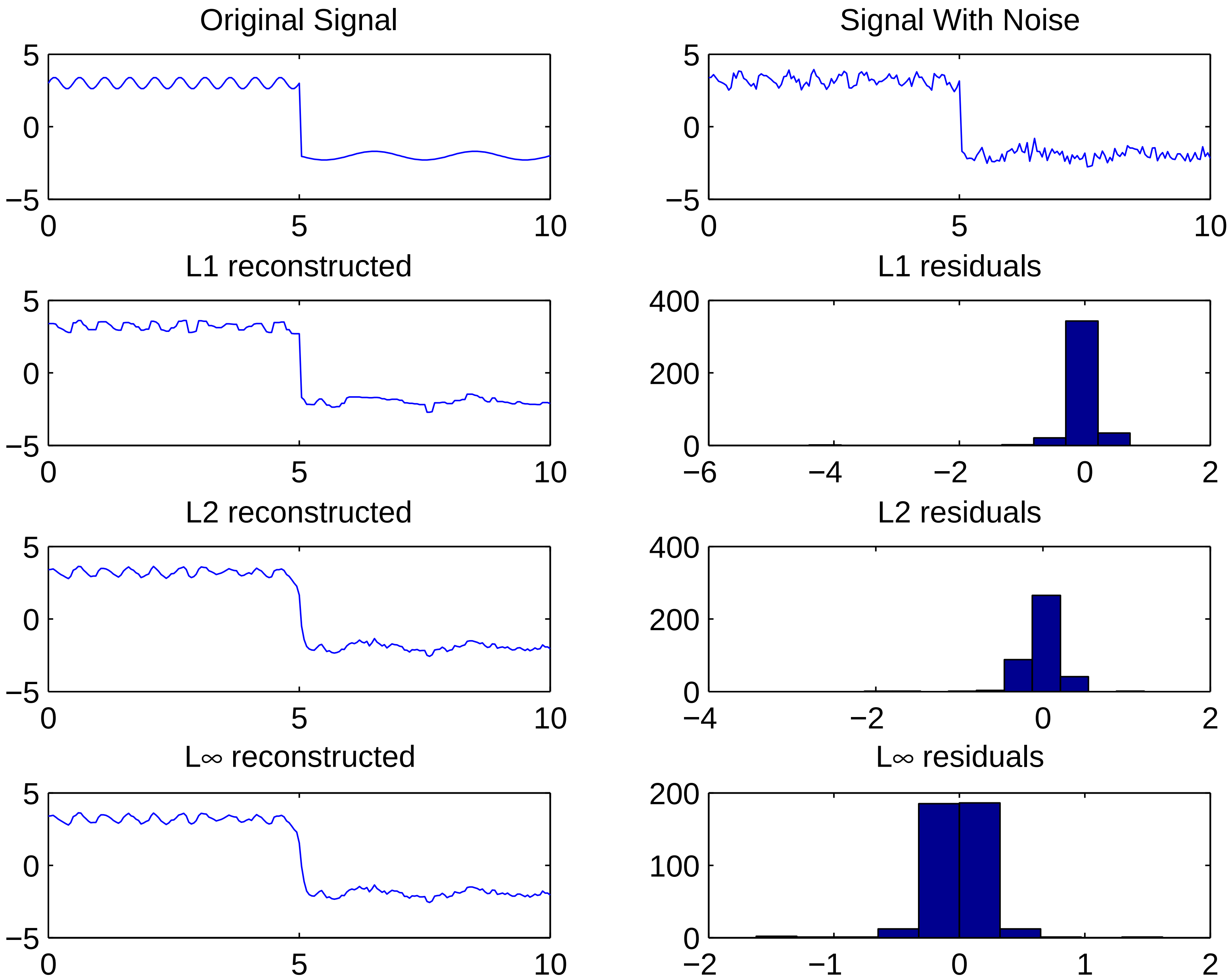
Max. variance smoothing: L_1 least squares problem.

$$\text{minimize} \quad \left\| \begin{bmatrix} I \\ \Delta \end{bmatrix} \vec{y}^{*} - \begin{bmatrix} \vec{y}_n \\ 0 \end{bmatrix} \right\|_\infty .$$

Where Δ is the matrix:

$$\Delta = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

Experiment



Original Signal
Signal With Noise
L1 reconstructed
L1 residuals
L2 reconstructed
L2 residuals
L∞ reconstructed
L∞ residuals

Other Smoothing Functions

Consider three successive data points instead of two:

$$\phi_T : \sum_{i=2}^{m-1} |2y_i - (y_{i-1} + y_{i+1})| \,.$$

Modify matrix Δ as follows:

$$\Delta = \begin{bmatrix} -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 & -1 \end{bmatrix}$$

Δ is called a <u>Topelitz Matrix</u>.

Application-3: Regression

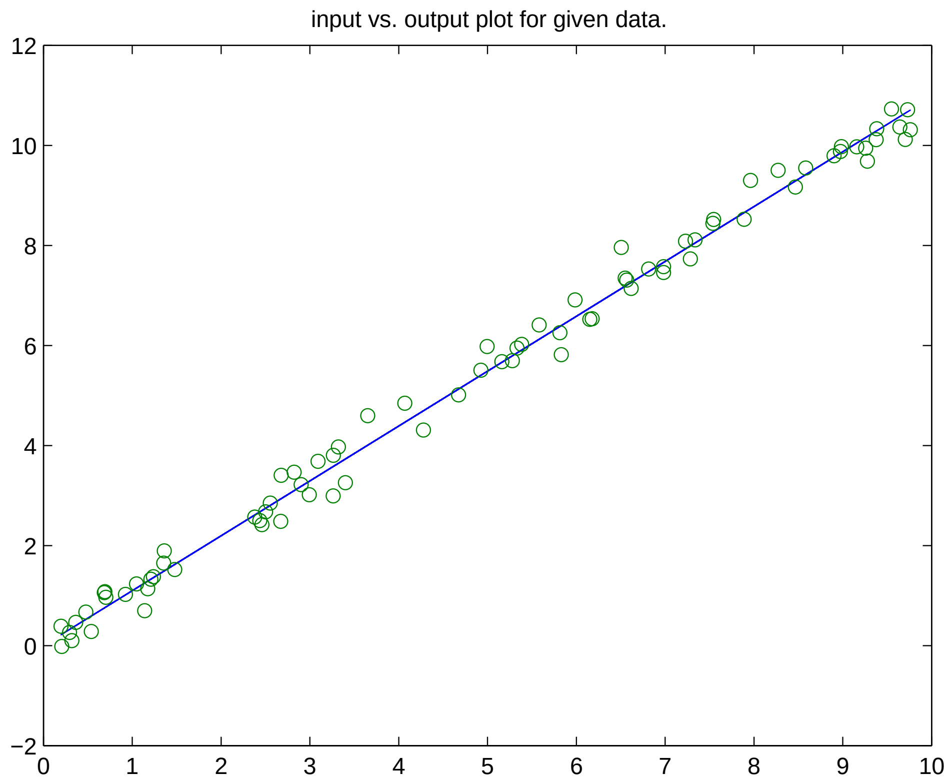
Linear Regression

Inputs: Data points $\vec{x}_i : (x_{i1}, \ldots, x_{in})$ and outcome y_i.

Goal: Find a function

$$f(\vec{x}) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + c_0$$

that best fits the data.



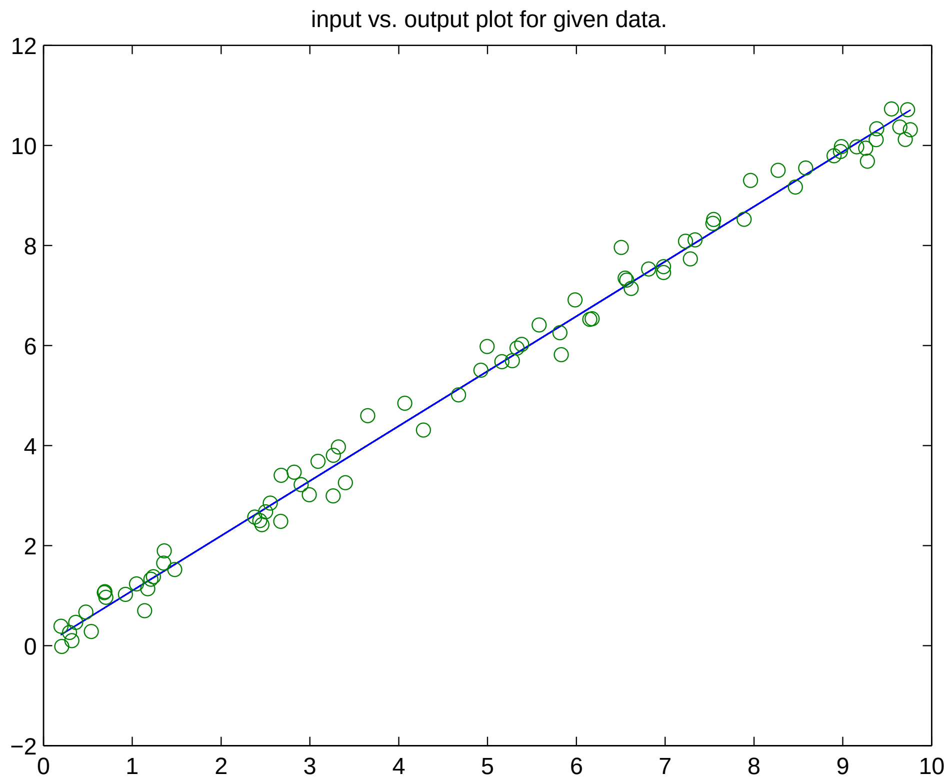
Linear Regression

Inputs: Data points $\vec{x}_i : (x_{i1}, \ldots, x_{in})$ and outcome y_i.

Goal: Find a function

$$f(\vec{x}) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + c_0$$

that best fits the data.



Linear Regression

Let X be the data matrix and $\vec{y}$ be corresponding outcome matrix.

Note: The i^{th} row X_i represents data point i
y_i is the corresponding outcome for X_i

If $\vec{c}^{\mathrm{T}}\vec{x} + c_0$ is the best line fit, the error for i^{th} data point is:

$$\epsilon_i : \ (X_i \cdot \vec{c}) + c_0 - y_i \,.$$

We can write the overall error as:

$$\left\| [X \quad \vec{1}]\vec{c} - \vec{y} \right\|_p \,.$$

Regression

Inputs: Data X, outcome $\vec{y}$.
Decision Vars: $c_0, c_1, \ldots, c_n$ the coefficients of the straight line.
Solution:minimize $\left\| [X \quad \vec{1}]\vec{c} - \vec{y} \right\|_p$.

Note: Instead of the norm, we could use any penalty function.

Ordinary Regression: min $\left\| [X \quad \vec{1}]\vec{c} - \vec{y} \right\|_p^p$.

Tikhanov Regression: min $\left\| [X \quad \vec{1}]\vec{c} - \vec{y} \right\|_2^2 + \lambda \cdot ||\vec{c}||_2^2$.

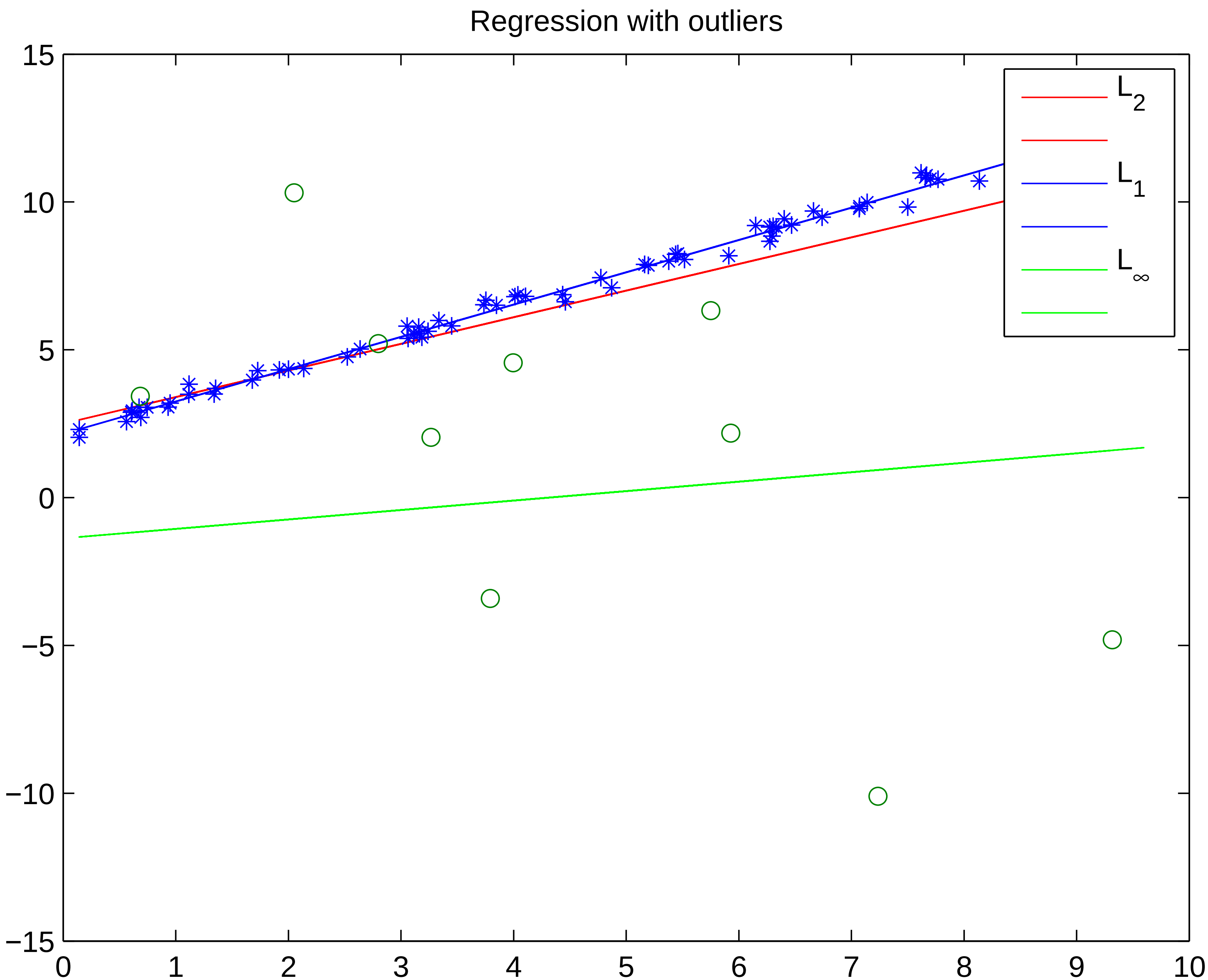
Experiment

Experiment: Create noisy data set with outliers.

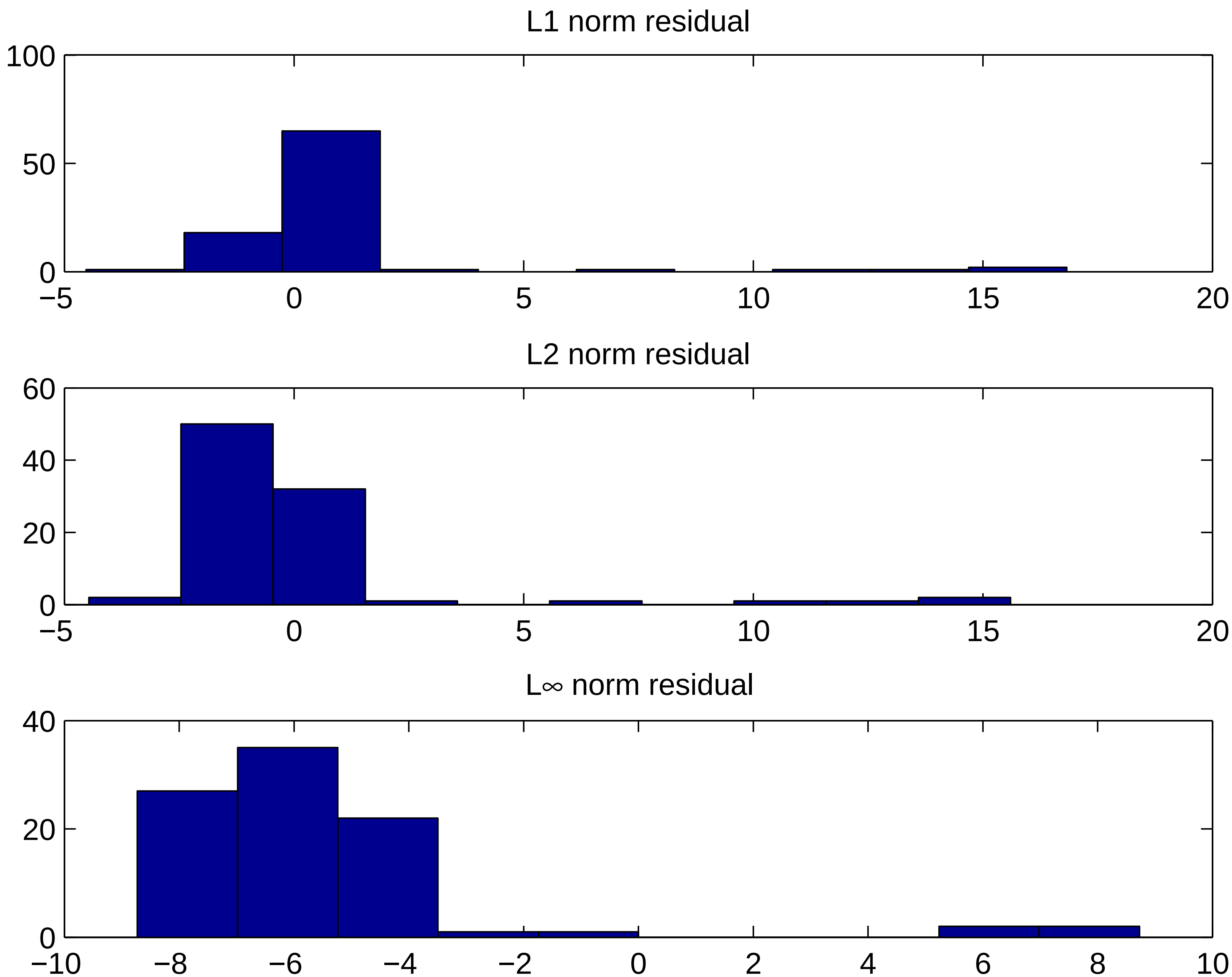
80 regular data points with low noise.
5 outlier datapoints with large noise.

Goal: Compare effect of L_1, L_2, L_∞ norm regressions.

Regression with Outliers



Regression with Outliers (Error Histogram)



Linear Regression With Non-Linear Models

Input: Data X, outcome $\vec{y}$, functions $g_1, \ldots, g_k$.
Goal: Fit a model of the form

$$y_i = c_0 + c_1 g_1(\vec{x}) + c_2 g_2(\vec{x}) + \ldots + c_k g_k(\vec{x}) \, .$$

Solution: Transform the data matrix as follows:

$$X' = \left[\begin{array}{ccc} g_1(X_1) & \ldots & g_k(X_1) \\ g_1(X_2) & \ldots & g_k(X_2) \\ & \ddots & \\ g_1(X_n) & \ldots & g_k(X_n) \end{array} \right] \, .$$

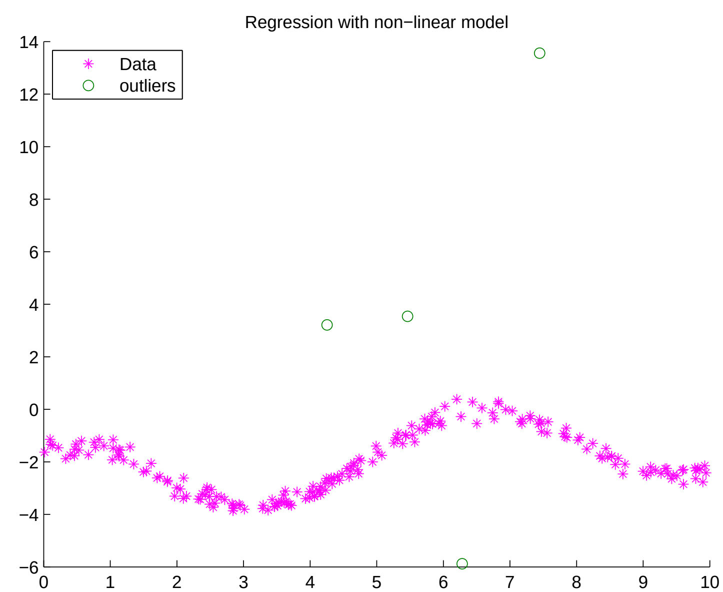
Apply linear regression on $(X', \vec{y})$.

Experiment

Experiment: Create noisy non-linear data set with outliers.

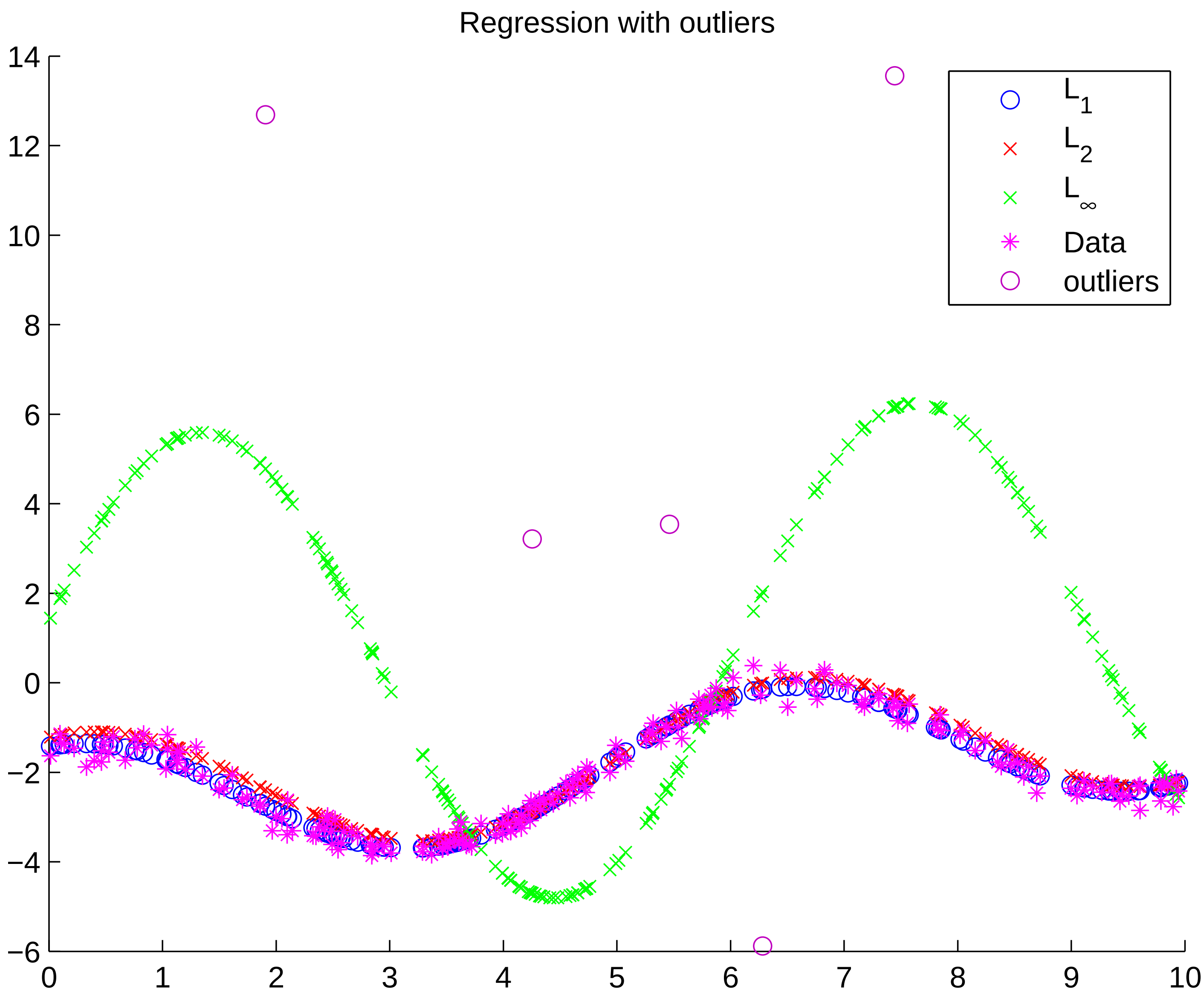
$$\vec{y} = 0.2sin(x) + 1.5 * cos(x) + 2 * cos(1.5 * x) + .2 * x - 3 + \text{Noise}\,.$$

80 regular data points with low noise.
5 outlier datapoints with large noise.

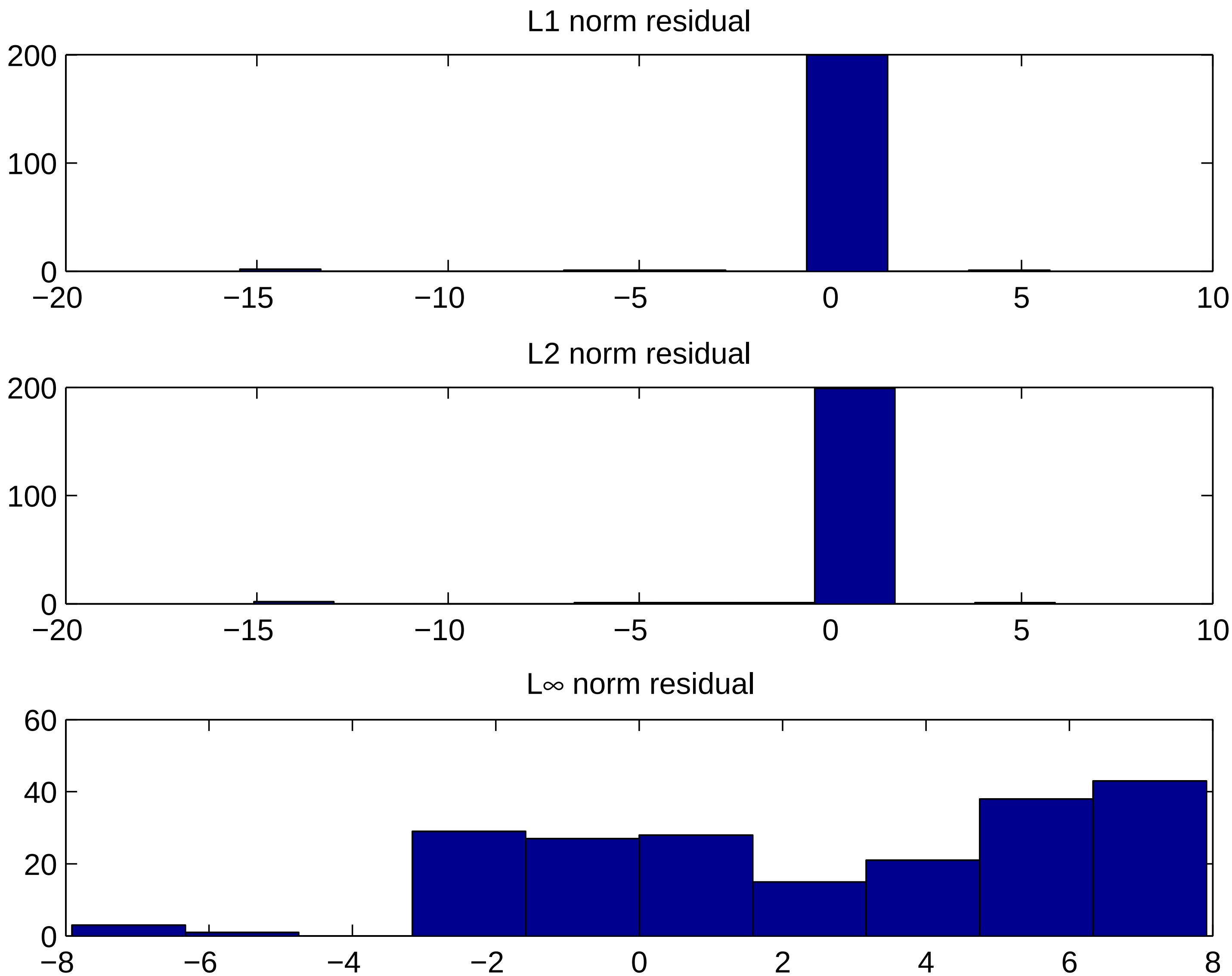


Goal: Compare effect of L_1, L_2, L_∞ norm regressions.

Experiment: linear regression non-linear model

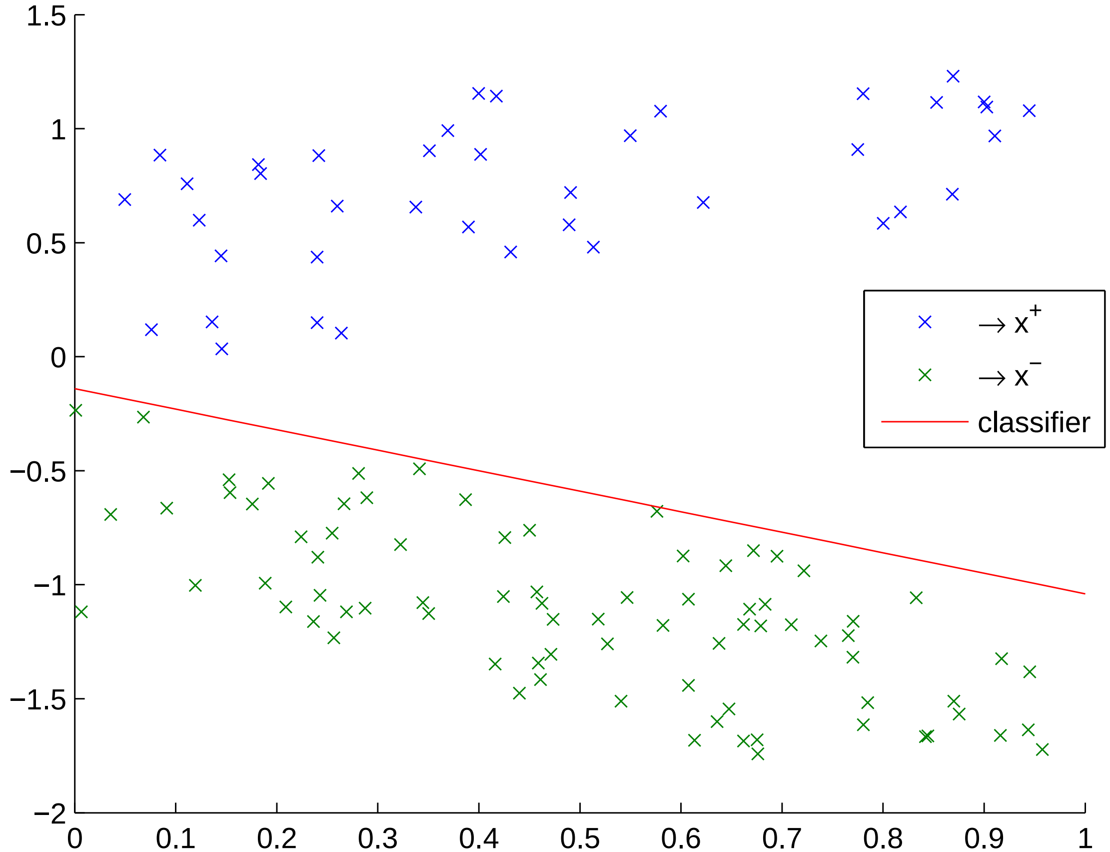


Experiment: residuals



Classification

Problem: Given two classes: $\vec{x}_1^+, \ldots, \vec{x}_N^+$ and $\vec{x}_1^-, \ldots, \vec{x}_M^-$ of data points.
Goal: Find separating hyperplane between the classes.



Application-4: Classification

Linear Classification

Problem: Given two classes: $\vec{x}_1^+, \ldots, \vec{x}_N^+$ and $\vec{x}_1^-, \ldots, \vec{x}_M^-$ of data points.
Goal: Find separating hyperplane between the classes.

Q-1: Will such an hyperplane always exist?

A: Not always! Only if data is <u>linearly separable</u>.

1. Assume that such a hyperplane exists.
2. Deal with cases where a hyperplane does not exist.

Linear Classification (cont)

Solution: Use linear programming.
Decision Variables: $w_0, w_1, w_2, \ldots, w_n$, representing the classfier:

$$f_w(\vec{x}) : w_0 + w_1 x_1 + \cdots + w_n x_n \,.$$

Linear Programming:

$$\begin{array}{rcl} \text{max.} & ??? & \\ X^{+}\vec{w} & \geq & 0 \\ X^{-}\vec{w} & \leq & 0 \end{array}$$

$$X^{+} : \left[\begin{array}{cc} \vec{x}_1^{+} & 1 \\ \vec{x}_2^{+} & 1 \\ \vec{x}_3^{+} & 1 \\ \vdots & 1 \\ \vec{x}_N^{+} & 1 \end{array}\right] \quad X^{-} : \left[\begin{array}{cc} \vec{x}_1^{-} & 1 \\ \vec{x}_2^{-} & 1 \\ \vdots & 1 \\ \vec{x}_M^{-} & 1 \end{array}\right]$$

Classification Using Linear Programs

Objective: Lets try $\sum_i w_i = 1^{\mathrm{T}} \vec{w}$.

Note: We write $\left\langle \vec{a}, \vec{b} \right\rangle$ for dot product $\vec{a}^{\mathrm{T}} \vec{b}$.

Simplified Data representation: Represent two classes $+1, -1$ for data.

$$\textbf{Data:}(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N) \Rightarrow (X, \vec{y})\,,$$

where $y_i = +1$ for class I and $y_i = -1$ for class II data.

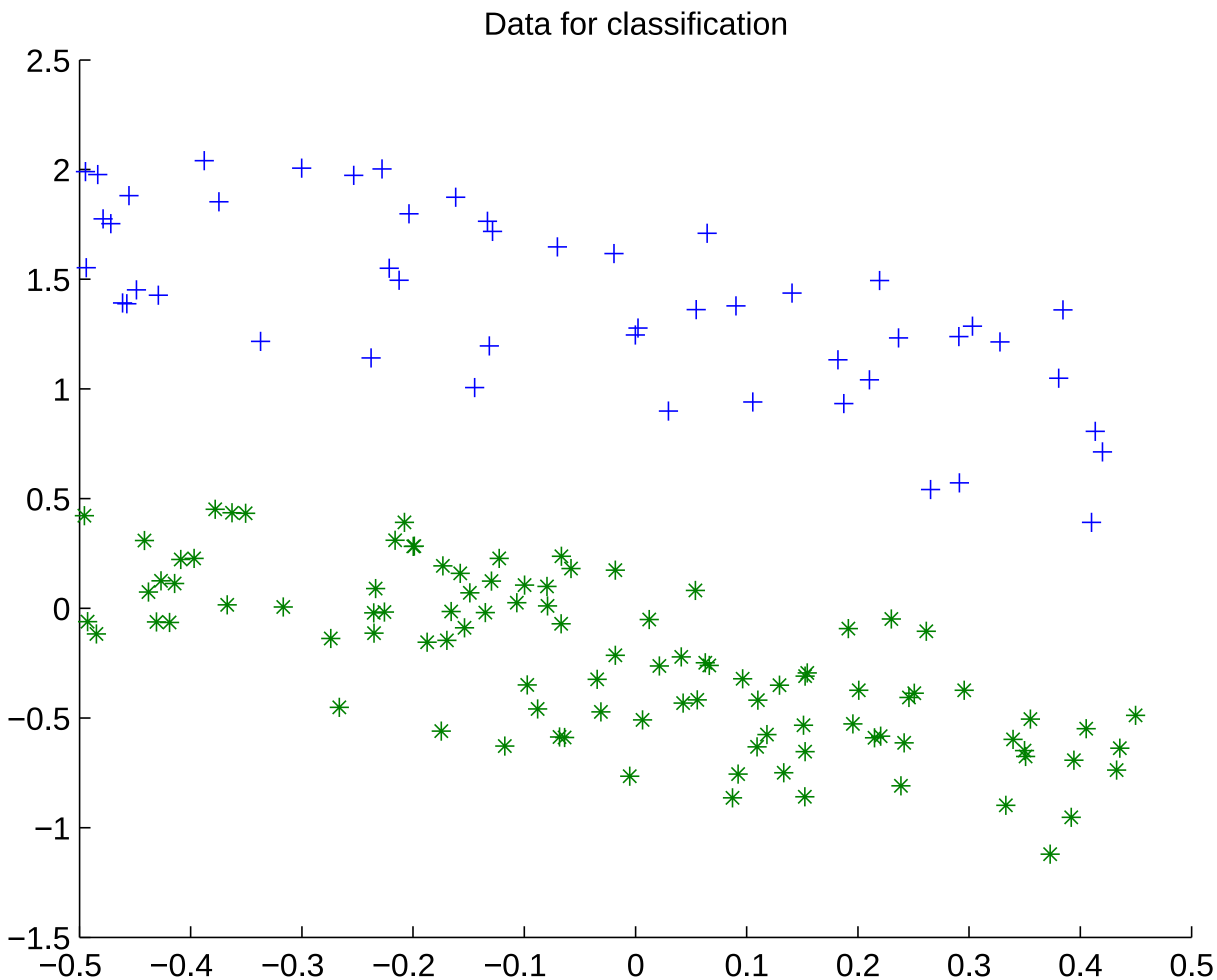
Linear Program:

$$\begin{array}{lrcl} \text{min.} & \left\langle \vec{1}, \vec{w} \right\rangle + w_0 & & \\ & y_i \left(\langle \vec{x}_i, \vec{w} \rangle + w_0 \right) & \leq & 0 \end{array}$$

Verify that this formulation works.

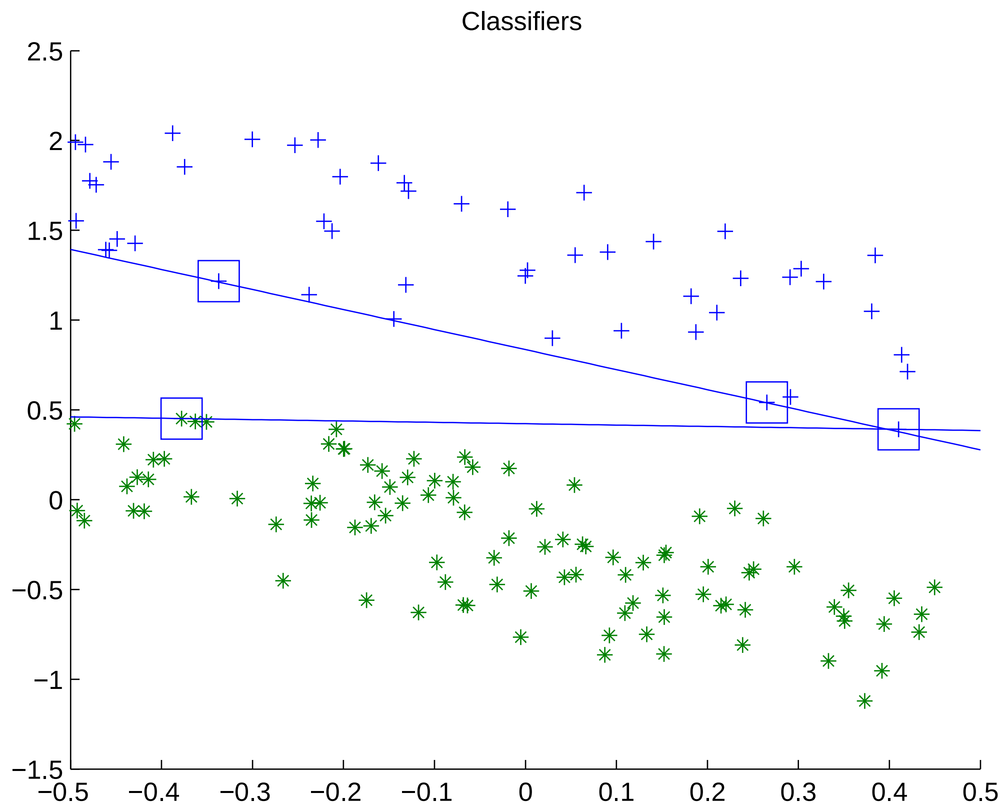
Experiment: Classification

Experiment: Find a classifier for the following points.



Experiment: Classification

Experiment: Outcome.



Note: Many different classifiers are possible.

Note-2: What is the deal with points that lie on the classifier line?

Dual, Complementary Slackness

Linear Program:

$$\begin{aligned} \text{min.} \quad & \left\langle \vec{1}, \vec{w} \right\rangle + w_0 \\ y_i \left(\langle \vec{x}_i, \vec{w} \rangle + w_0 \right) & \leq \quad 0 \end{aligned}$$

Complementary Slackness: Let $\alpha_1, \ldots, \alpha_N$ be the dual variables.

$$\alpha_i y_i \left(\langle \vec{x}_i, \vec{w} \rangle + w_0 \right) = 0, \;\; \alpha_i \geq 0 \, .$$

Claim: There are at least $n + 1$ points $\vec{x}_{j_1}, \ldots, \vec{x}_{j_{n+1}}$ s.t.

$$\langle \vec{x}_{j_i}, \vec{w} \rangle + w_0 = 0 \, .$$

Note: $n + 1$ pts. fully determine the hyper-plane $\vec{w}; w_0$.

Classifying Points

Problem: Given a new point $\vec{x}$, let us find which class it belongs to.
Solution-1: Just find out the sign of $\langle \vec{w}, \vec{x} \rangle + w_0$, we know $\vec{w}; w_0$ from our LP solution.

Solution-2: More complicated. Express $\vec{x}$ as a linear combination of support vectors:

$$\vec{x} = \lambda_1 \vec{x}_{j_1} + \lambda_2 \vec{x}_{j_2} + \cdots + \lambda_{n+1} \vec{x}_{n+1}$$

We can write $\langle \vec{w}, \vec{x} \rangle$ as

$$\langle \vec{w}, \vec{x} \rangle + w_0 = \sum_i \lambda_i \left(\underbrace{w_0 + \langle \vec{w}, \vec{x}_{j_i} \rangle}_{0} \right) + \left(1 - \sum_i \lambda_i \right) w_0 = \left(1 - \sum_i \lambda_i \right) w_0 \,.$$

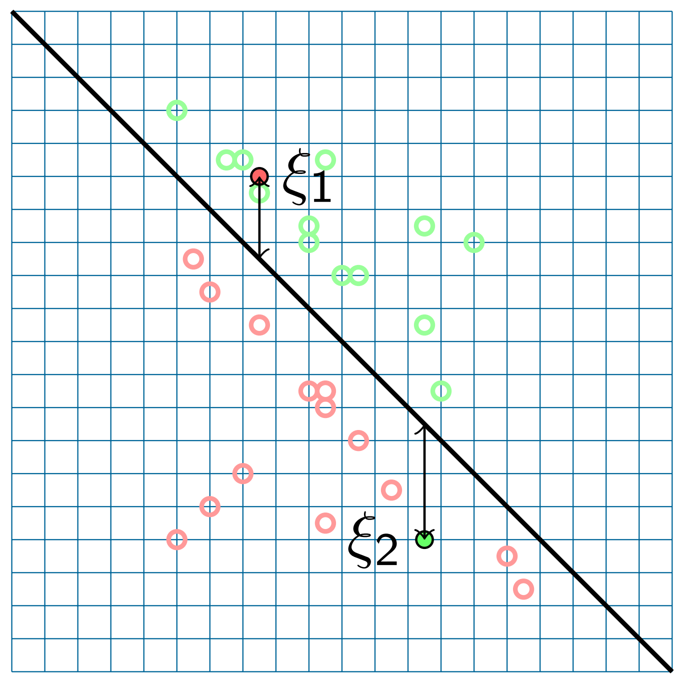
This is impractical for linear programming formulations.

Non-Separable Data

Q: What if data is not linear separable?
The linear program will be primal infeasible

In practice, linear separation may hold without noise but not with noise.

Solution: Relax the conditions on half-space w.



Non-Separable Data: Linear Programming

Classifier: (old) $w_0 + \langle \vec{w}, \vec{x} \rangle$.

(new) $\langle \vec{w}, \vec{x} \rangle + 1$.

Note: Classifier $\sum_i w_i x_i + w_0$ is equivalent to $\sum_i \frac{w_i}{w_0} x_i + 1$ if $w_0 \neq 0$.

LP Formulation:

$$
\begin{array}{lrcl}
\text{minimize} & \left\| \vec{\xi} \right\|_1 + \cdots & & \\
& \left\langle \vec{x}_i^{+}, \vec{w} \right\rangle + 1 & \leq & \xi_i^{+} \\
& \left\langle \vec{x}_j^{-}, \vec{w} \right\rangle + 1 & \geq & -\xi_j^{-} \\
& \xi_i^{+}, \xi_j^{-} & \geq & 0
\end{array}
$$

The variable ξ encodes tolerance. We minimize ξ as part of objective.

Exercise: Verify that the support vector interpretation of dual works.

Non Linear Model Classification

Linear Classifier: $\langle \vec{w}, \vec{x} \rangle + 1$.

Non-linear functions: $\vec{\phi} : \mathcal{R}^n \mapsto \mathcal{R}$.

$$\langle \vec{w}, (\phi_1(\vec{x}), \phi_2(\vec{x}), \ldots, \phi_m(\vec{x})) \rangle \ .$$

This is exactly same idea as linear regression with non-linear functions.