Chapter 1

Application overview (1)

Contents (class version)

1.0 Introduction

(Read)

This chapter gives an overview of some of the signal/image processing problems that will be addressed later. This chapter does not describe any algorithms; just the cost functions are shown.

We will especially focus on **unconstrained optimization** problems of the form

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \tag{1.1}$$

where $\Psi : \mathbb{F}^N \mapsto \mathbb{R}$ denotes a **cost function** (or **loss function**) whose minimizer $\hat{\boldsymbol{x}}$ we seek to estimate a **latent variable** $\boldsymbol{x}$.

The **field** $\mathbb{F}$ will be either $\mathbb{R}$ or $\mathbb{C}$ depending on the application. These notes use $\mathbb{F}$ to maintain generality when possible, though many results will be presented only for $\mathbb{R}$.

We will also consider many **constrained optimization** problems of the form

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \Psi(\boldsymbol{x}), \tag{1.2}$$

where $\mathcal{X} \subset \mathbb{F}^N$ denotes a **feasible set** of possible latent variables such as the **nonnegative orthant**.

Characteristic functions

To unify (1.1) and (1.2), we define the **characteristic function** used in convex analysis as

$$\chi_{\mathcal{X}}(\boldsymbol{x}) = \begin{cases} \infty, & \boldsymbol{x} \in \mathcal{X} \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

Note that the **image** of this function is $\{0, \infty\}$ and the function's **codomain** is the **extended real numbers** that includes $\pm\infty$ (but is not a **field**).

With this definition, we rewrite the constrained optimization problem (1.2) in "unconstrained" form as

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{F}^N}{\arg\min}\, \Psi(\boldsymbol{x}) + \chi_{\mathcal{X}}(\boldsymbol{x}). \tag{1.4}$$

Caution. The term **characteristic function** has multiple meanings. In probability it means the Fourier transform of the pdf of a distribution (EECS 501). It also is a synonym for the related **indicator function**:

$$1_{\mathcal{X}}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \notin \mathcal{X} \\ 0, & \text{otherwise.} \end{cases} \tag{1.5}$$

These notes will always refer to (1.3) as the **characteristic function** and (1.5) as the **indicator function**.

Subsequent sections consider applications having cost functions of roughly increasing difficulty: linear, quadratic, convex and smooth, composite, non-smooth.

1.1 Linear programming applications

Noiseless compressed sensing

In unrealistic **compressed sensing** formulations that disregard noise, one might wish to find the sparsest solution to a set of linear equations:

(1.6)

This is also called a **sparse representation** problem. Unfortunately, this optimization problem is **NP-hard**. Each of the N elements of $\hat{\boldsymbol{x}}$ is either zero or non-zero; one could solve (1.6) by trying all 2^N possible supports of $\hat{\boldsymbol{x}}$ and choosing one with the smallest $\|\boldsymbol{x}\|_0$ that satisfies $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$.

Example. For $\boldsymbol{A}$ and $\boldsymbol{y}$ shown below, there are two optimal solutions, both with $\|\hat{\boldsymbol{x}}\|_0 = 2$: (Read)

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 0 & 1 & 2 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}, \qquad \hat{\boldsymbol{x}} = \begin{bmatrix} 0 \\ -4 \\ 0 \\ 3 \end{bmatrix}, \quad \hat{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 3 \end{bmatrix}.$$

To find those solutions, we tried all $\begin{pmatrix} 4 \\ 1 \end{pmatrix} = 4$ cases where $\|\boldsymbol{x}\|_0 = 1$, then all $\begin{pmatrix} 4 \\ 2 \end{pmatrix} = 6$ cases with $\|\boldsymbol{x}\|_0 = 2$, two of which sufficed, so there was no need to look at further cases.

There are many approximate solution methods for (1.6), such as **matching pursuit**.

An alternative to using approximate methods is to use the **convex relaxation** of the problem (see p. 1.10), as follows:

(1.7)

where $\boldsymbol{A} \in \mathbb{F}^{M \times N}$ is a known sensing matrix and $\boldsymbol{y} \in \mathbb{F}^{M}$ denotes a given measurement vector.

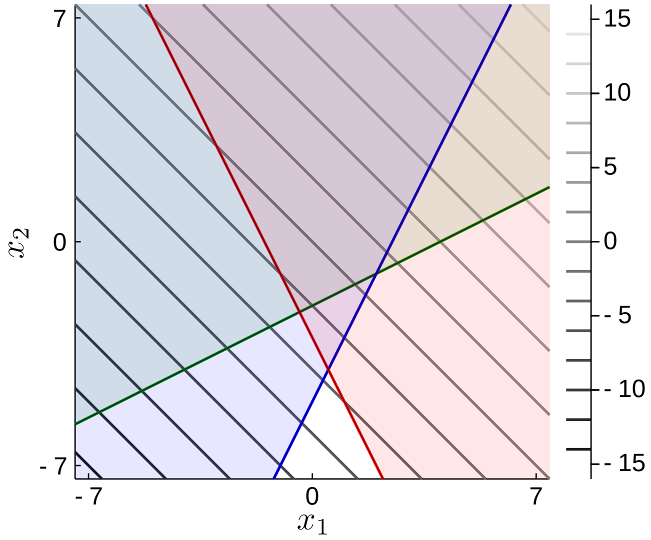
This relaxation (1.7) is still a nontrivial optimization problem because $\|\cdot\|_1$ is not a differentiable function, so gradient descent is not applicable.

When $\mathbb{F} = \mathbb{R}$, by expressing $\boldsymbol{x} = \boldsymbol{u} - \boldsymbol{v}$ where $\boldsymbol{u} \geq \boldsymbol{0}$ and $\boldsymbol{v} \geq \boldsymbol{0}$ so that $\|\boldsymbol{x}\|_1 = \boldsymbol{1}_N' \boldsymbol{u} + \boldsymbol{1}_N' \boldsymbol{v} = \boldsymbol{1}_{2N}' \boldsymbol{z}$ with $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}$, we rewrite (1.7) as a **linear programming** problem as follows (here the $\leq$ means element-wise):

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \hat{\boldsymbol{z}}, \quad \hat{\boldsymbol{z}} = \mathop{\arg\min}_{\boldsymbol{z} \in \mathcal{Z}} \Psi(\boldsymbol{z}), \quad \underbrace{\Psi(\boldsymbol{z}) \triangleq \boldsymbol{1}_{2N}' \boldsymbol{z},}_{\text{linear}} \quad \mathcal{Z} \triangleq \left\{ \boldsymbol{z} \in \mathbb{R}^{2N} : \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{A} \\ -\boldsymbol{A} & \boldsymbol{A} \\ -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} \end{bmatrix} \boldsymbol{z} \leq \begin{bmatrix} \boldsymbol{y} \\ -\boldsymbol{y} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \right\}.$$

However, this approach does not generalize to $\mathbb{C}$ nor to noisy formulations considered later.

Example. This figure illustrates the contours of a linear cost function $\mathbf{1}'\boldsymbol{z}$ for a case where the constraints $\mathcal{Z}$ consist of three half planes.



Where is the minimizer $\hat{z}$?

$\boxed{\textbf{??}}$

How many half spaces in $\mathcal{Z}$ on preceding page? *(Introduce neighbors...)*

A: $2(N+M)$ B: $4M$ C: 4 D: $M+N$ E: None of these $\boxed{\textbf{??}}$

Minimax sparse filter design (Read)

Minimizing the worst-case error, by using $\|\cdot\|_\infty$, arises in some applications including sparse filter design [1]. A typical formulation looks like the following **Chebyshev approximation** problem [2, p. 6]:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_\infty, \tag{1.8}$$

for some given $\boldsymbol{A} \in \mathbb{F}^{M \times N}$ and $\boldsymbol{y} \in \mathbb{F}^M$. This is also called **minimax** design.

An equivalent form to (1.8) is the following constrained problem:

$$\mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \min_{t \in \mathbb{R}} t \ \text{ such that } \ \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_\infty \leq t. \tag{1.9}$$

When $\mathbb{F} = \mathbb{R}$, another equivalent form (that eliminates the infinity norm) is:

$$\mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{R}^N} \min_{t \in \mathbb{R}} t \ \text{ such that } \ -t\mathbf{1}_M \leq \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \leq t\mathbf{1}_M. \tag{1.10}$$

We can rewrite that problem as the following standard **linear program**:

$$\mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{R}^N} \min_{t \in \mathbb{R}} t \ \text{ such that } \ \begin{bmatrix} \boldsymbol{A} & -\mathbf{1}_M \\ -\boldsymbol{A} & -\mathbf{1}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{y} \\ -\boldsymbol{y} \end{bmatrix}. \tag{1.11}$$

To be precisely in standard linear program form we can rewrite the latent variables as

$$t = \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix}.$$

Exercise. What is the size of the preceding $\mathbf{0}$? $\boxed{\textbf{??}}$

MRI RF pulse design (Read)

Designing MRI RF pulses is somewhat similar to filter design [3] except there are also peak amplitude constraints to to hardware limits, so one formulation of interest is:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_\infty \text{ such that } \|\boldsymbol{x}\|_\infty \leq b. \tag{1.12}$$

For $\mathbb{F} = \mathbb{R}$ one can also rewrite this constrained optimization problem as a **linear program**. (HW)

JULIA has linear programming tools (`https://www.juliaopt.org/`) [4]. Unfortunately, in general, linear program solvers do not scale well to large scale problems where $\boldsymbol{A}$ is computed **on the fly**.

Convex relaxation

Define. The **sublevel sets** of a function $f : \mathbb{F}^N \mapsto \mathbb{R}$ are defined as follows:

Example. The **sublevel sets** of the p-norms $\|\boldsymbol{x}\|_p = \left(\sum_{n=1}^N |x_n|^p\right)^{1/p}$ for $p = 1$ and $p = 2$.

$\{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\|_1 \leq c\}$ $\{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\|_2 \leq c\}$

Define. A function whose **sublevel sets** are all **convex sets** is called a **quasiconvex function**.

Every **convex function** is a **quasiconvex function**.

A: True B: False $\boxed{??}$

$\boxed{??}$

The converse does not hold.

Example. The function $f(x) = \sqrt{|x|}$ is **quasiconvex** on $\mathbb{R}$, but is not convex. Likewise for $f(\boldsymbol{x}) = \sqrt{\|\boldsymbol{x}\|_1}$ on $\mathbb{F}^N$.

Now consider the **sublevel sets** of the so-called "p-norms" $\|\boldsymbol{x}\|_p = \left(\sum_{n=1}^N |x_n|^p\right)^{1/p}$ for $0 \leq p < 1$.

The **convex hull** of any such **sublevel set** for $0 \leq p < 1$ *i.e.*, the set of all convex combinations of points from any such set, is the **sublevel set** for $p = 1$.

This is a geometric sense in which $\|\cdot\|_1$ is the **convex relaxation** of $\|\cdot\|_p$ for $0 \leq p < 1$.

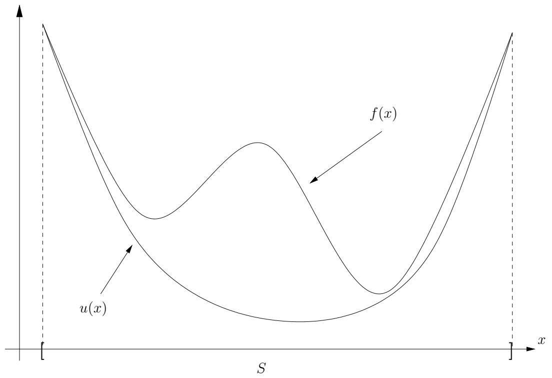
For more general constructions, one uses the following definitions. (Read)

Define. For a function $f : \mathcal{S} \mapsto \mathbb{R}$, a **convex relaxation** of f on $\mathcal{S}$ is any convex function $u : \mathcal{S} \mapsto \mathbb{R}$ such that

$$u(\boldsymbol{x}) \leq f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{S}.$$

Example. Figure from

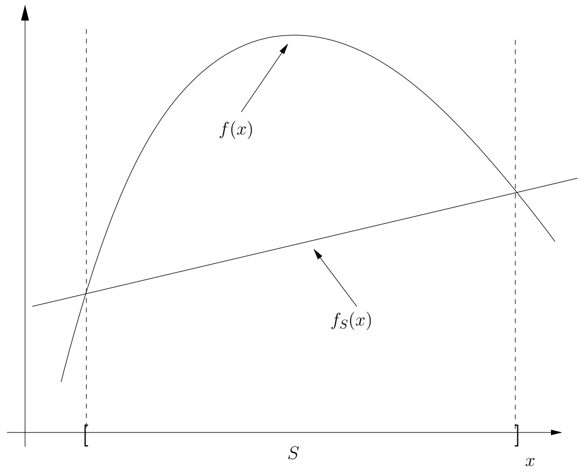
http://www.winlab.rutgers.edu/~crose/322_html/convexity2.pdf



By the way, the above URL is an excellent read for an overview of topics like **convex function**, **local minimizer**, **global minimizer**, **stationary point**, etc.

Define. The **lower convex envelope** of f on $\mathcal{S}$ is the **supremum** of all **convex relaxations** of f on $\mathcal{S}$.

It is sometimes denoted $f_{\mathcal{S}}$ or $\breve{f}$ or $\breve{f}_{\mathcal{S}}$.



In other words, the **lower convex envelope** is the tightest **convex relaxation** of f on $\mathcal{S}$.

When people say "the convex relaxation" of a function, typically they are referring to the **lower convex envelope**, and usually they disregard $\mathcal{S}$ and constant scale factors.

Example. For $f(x) = \sqrt{|x|}$ and $\mathcal{S} = [-a, a]$, we have $\breve{f}_{\mathcal{S}}(x) = |x| / \sqrt{a}$.

More reading: [rutgers] [quora] [dual] [boyd]

1.2 Quadratic / LS problems

(Read)

Many applications involve linear models (with noise):

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}, \tag{1.13}$$

where $\boldsymbol{y} \in \mathbb{F}^M$ is a measurement vector, $\boldsymbol{A} \in \mathbb{F}^{M \times N}$ is a known matrix $\boldsymbol{x} \in \mathbb{F}^N$ is a latent parameter vector, and $\boldsymbol{\varepsilon} \in \mathbb{F}^M$ is an additive noise vector. For such linear models, a common approach to estimating $\boldsymbol{x}$ is to solve a linear **least-squares** (**quadratic**) optimization problem:

$$\hat{\boldsymbol{x}} = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 . \tag{1.14}$$

When the noise has a zero-mean Gaussian distribution with covariance $\sigma^2 \boldsymbol{I}$, *i.e.*, $\boldsymbol{y} \sim \mathsf{N}(\boldsymbol{A}\boldsymbol{x}, \sigma^2 \boldsymbol{I})$, then the LS estimator (1.14) is the **maximum likelihood** estimate, *i.e.*,

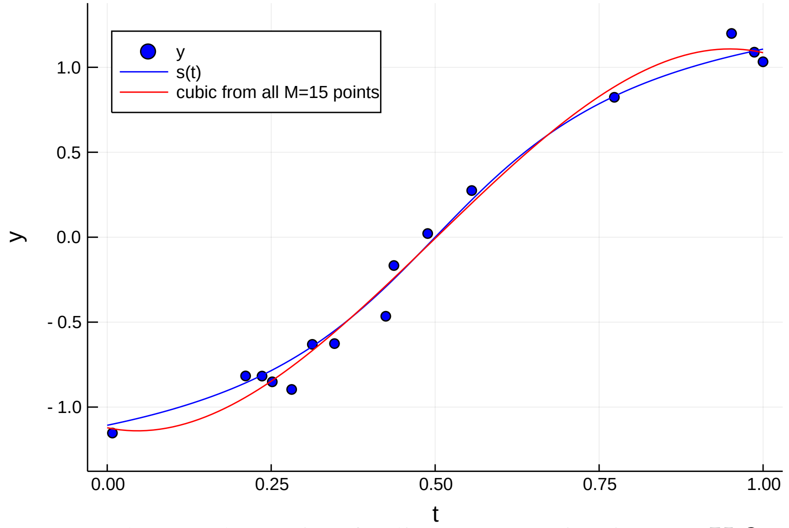
$$\hat{\boldsymbol{x}} = \operatorname*{arg\,max}_{\boldsymbol{x} \in \mathbb{F}^N} \mathsf{p}(\boldsymbol{y}; \boldsymbol{x}) .$$

These types of LS problems (and variations thereof) arise in at least two quite different settings.

- In **inverse problems**, $\boldsymbol{x}$ represents the input to a linear system modeled by $\boldsymbol{A}$ and we want to recover $\boldsymbol{x}$ from the system output $\boldsymbol{y}$. Examples include image deblurring and tomographic image reconstruction. In these problems the matrix $\boldsymbol{A}$ represents a (known) model for the system based on the physics of the system. Often $\boldsymbol{A}$ is too large to store as a matrix!

- In **linear regression** problems, the rows of $\boldsymbol{A}$ a consist of *data* where we want to learn a linear model relating to the input and the output for the purpose of making future predictions. Examples include fitting polynomial model like the following cubic.

$$y_m = s(t_m) + \epsilon_m, \quad m = 1, \ldots, M$$
$$s(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3$$
$$\boldsymbol{A} = \begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & & & \vdots \\ 1 & t_M & t_M^2 & t_M^3 \end{bmatrix}$$
$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$



In statistics (and some machine learning literature) the usual notation for linear regression is $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ where the matrix $\boldsymbol{X}$ has the data in it and the coefficients are $\boldsymbol{\beta}$.
Note that the term "linear" here refers to coefficients $\boldsymbol{\beta}$. The matrix $\boldsymbol{X}$ can include *nonlinear* functions of the data features, like the powers of t_m in the **Vandermonde matrix** $\boldsymbol{A}$ above.

Solving least-squares problems

A necessary condition for $\hat{\boldsymbol{x}}$ to be a minimizer of (1.14) is $\nabla \Psi(\hat{\boldsymbol{x}}) = \boldsymbol{0}$ leading to the **normal equations**

$$\boldsymbol{A}'\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}'\boldsymbol{y}.$$

Conventional wisdom is to avoid using this form numerically due to finite precision issues.

If $\boldsymbol{A}$ has full column rank, then $\boldsymbol{A}'\boldsymbol{A}$ is invertible and the solution becomes

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{y}.$$

This expression is useful for analysis but not for practical implementation when N is large. When N is large, we need iterative algorithms to compute $\hat{\boldsymbol{x}}$.

Furthermore, when $\boldsymbol{A}$ is under-determined (wide) or poorly conditioned, then we need constraints or regularizers to make the solution behave better.

The quadratic LS optimization problem will be our starting point for discussing several iterative algorithms:
- (preconditioned) gradient descent (**PGD**)
- (preconditioned) steepest descent (**PSD**)
- (preconditioned) conjugate gradients (**PCG**)
- quasi-Newton methods (**BFGS**)
- majorize-minimize (**MM**) methods
- accelerated first-order methods with momentum (**FGM**), (**OGM**).

Regularization

When $\boldsymbol{A}$ is wide or is poorly conditioned, then one must use **regularization**. The simplest approach, called **ridge regression** in statistics, is **Tikhonov regularization** that leads to another quadratic optimization problem:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{1}{2}\beta \|\boldsymbol{x}\|_2^2 .$$

This quadratic problem again has a (unique, for any $\boldsymbol{A}$, when $\beta > 0$) closed-form solution:

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}'\boldsymbol{A} + \beta \boldsymbol{I})^{-1} \boldsymbol{A}'\boldsymbol{y}.$$

Again we need iterative algorithms to compute $\hat{\boldsymbol{x}}$ when N is large.

The Tikhonov regularizer $R(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{x}\|_2^2$ discourages large values of $\boldsymbol{x}$, and it is a reasonable starting point for linear regression problems.

For many imaging problems, we expect neighboring pixels to have similar values, so a more reasonable regularizer discourages differences between neighbors:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{1}{2}\beta \|\boldsymbol{T}\boldsymbol{x}\|_2^2 , \tag{1.15}$$

where $\boldsymbol{T}$ is a regularization matrix such as the following matrix that computes **finite differences** in 1D:

$$\boldsymbol{T} = \boldsymbol{D}_N \triangleq \begin{bmatrix} -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ & & \ddots & \ddots & & \\ 0 & \dots & 0 & -1 & 1 & 0 \\ 0 & \dots & 0 & 0 & -1 & 1 \end{bmatrix}, \text{ so } \boldsymbol{T}\boldsymbol{x} = \begin{bmatrix} x_2 - x_1 \\ \vdots \\ x_N - x_{N-1} \end{bmatrix}. \tag{1.16}$$

For 2D problems with $M \times N$ images, usually we want differences along both rows and columns, so a typical choice uses the **Kronecker product** $\otimes$ as follows (EECS 551 HW problem):

$$\tag{1.17}$$

When $(\boldsymbol{A}'\boldsymbol{A} + \beta \boldsymbol{T}'\boldsymbol{T})$ is invertible, this quadratic problem (1.15) again has a (unique) closed-form solution:

$$\hat{\boldsymbol{x}} = \tag{1.18}$$

Again we need iterative algorithms to compute $\hat{\boldsymbol{x}}$ when N is large.

These various quadratic optimization problems are important in their own right, and many such problems arise as intermediate steps for other more complicated optimization problems, so we will describe several iterative algorithms for solving quadratic problems.

Example. This figure shows a 1D example of the estimate (1.18) for an interpolation problem where $\boldsymbol{A}$ is a down-sampling operator:

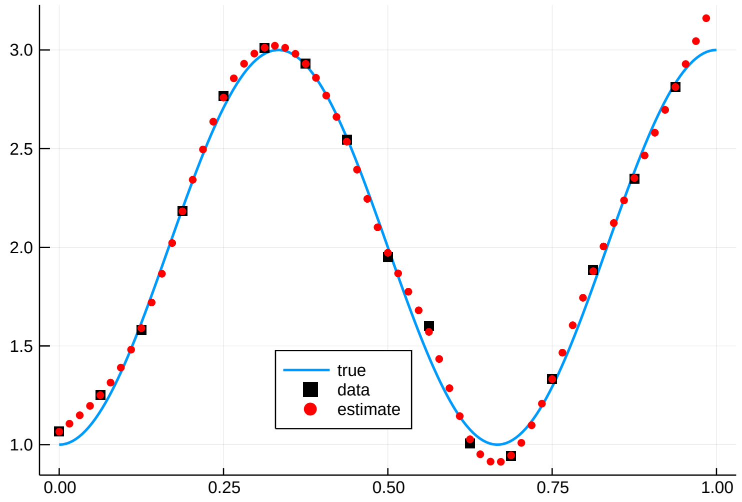
```
A = sparse(I, N, N)[1:4:N,:]
```

and $\boldsymbol{T}$ corresponds to 2nd-order finite differences:

```
T = spdiagm(-1 => -ones(N-1), 0 => 2*ones(N), 1 => -ones(N-1))[2:end-1,:]
```

This 1D problem is small enough to use (1.18) directly: `xh = (A'A + beta*T'T) \ y`

The resulting estimate $\hat{\boldsymbol{x}}$ agrees reasonably well with the (noisy) data while also providing a smooth result.



Constrained LS problems (Read)

Often there are constraints on the parameters so we want to solve

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\, \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{1.19}$$

where $\mathcal{X} \subset \mathbb{F}^N$ is a set of allowable parameters called the **feasible set**.

Common constraints (choices for $\mathcal{X}$):

- **Nonnegativity constraint** $\boldsymbol{x} \geq \boldsymbol{0}$, *e.g.*, in imaging problems with quantities like intensity or density.
- **Box constraint** or **bound constraint** $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$, where $\boldsymbol{l}$ and $\boldsymbol{u}$ denote lower and upper bounds, respectively.
- **Simplex constraint** $\boldsymbol{x} \geq \boldsymbol{0}$ and $\boldsymbol{1}'\boldsymbol{x} = 1$, when $\boldsymbol{x}$ represents fractions or probabilities.
- ...

Constrained problems do not have closed-form solutions in general so they usually require iterative algorithms even if N is not large. Often they require different iterative algorithms than unconstrained problems.

Iterative algorithms for constrained problems include

- gradient projection method (with diagonal preconditioner),
- majorize-minimize (MM) methods (with separable majorizers),
- other methods discussed under composite cost functions on p. 1.27.

There is one important case where there is a simple solution: when $\boldsymbol{A}$ is **diagonal**, $\mathbb{F} = \mathbb{R}$, and $\mathcal{X}$ is a **nonnegativity constraint** or more generally a **box constraint**.

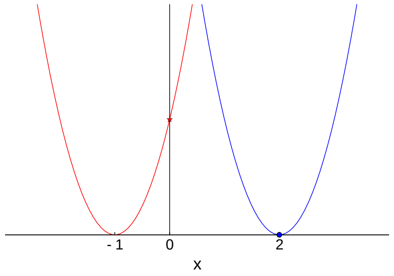
If $\boldsymbol{A}$ is square and diagonal with nonzero diagonal elements $a_1, \ldots, a_N$, then the quadric cost function (1.19) simplifies to the sum of N 1D terms:

$$\Psi(\boldsymbol{x}) = \sum_{n=1}^{N} \frac{1}{2} (a_n x_n - y_n)^2 . \tag{1.20}$$

Focusing on the **nonnegativity constraint**, the optimization problem **separates** into N 1D problems of the form

$$\hat{x}_n = \underset{x_n \geq 0}{\arg\min} \frac{1}{2} (a_n x_n - y_n)^2 = \underset{x_n \geq 0}{\arg\min} \frac{a_n^2}{2} (x_n - y_n/a_n)^2 . \tag{1.21}$$

The solution is $\hat{x}_n = \begin{cases} y_n/a_n, & y_n/a_n > 0 \\ 0, & \text{otherwise} \end{cases} = \max(y_n/a_n, 0).$



Code: `xh = max.(y ./ a, 0)`

Note the use of JULIA's powerful broadcast syntax here, shorthand for this:

```
broadcast((y,a) -> max(y/a,0), y, a)
```

1.3 Strictly convex smooth problems

Next we leave behind the simplicity of quadratic problems and consider applications involving optimization problems that are **convex** and **smooth**, *i.e.*, where the cost function is at least differentiable.

Edge-preserving regularization

A drawback of the quadratic finite-difference regularizer (1.15) is that it causes blurring across image edges. To reduce this blurring, we replace the quadratic regularizer with a **non-quadratic** function as follows:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{F}^N}{\arg\min}\, \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) \triangleq \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta R(\boldsymbol{x}),$$

where now the **regularizer** is

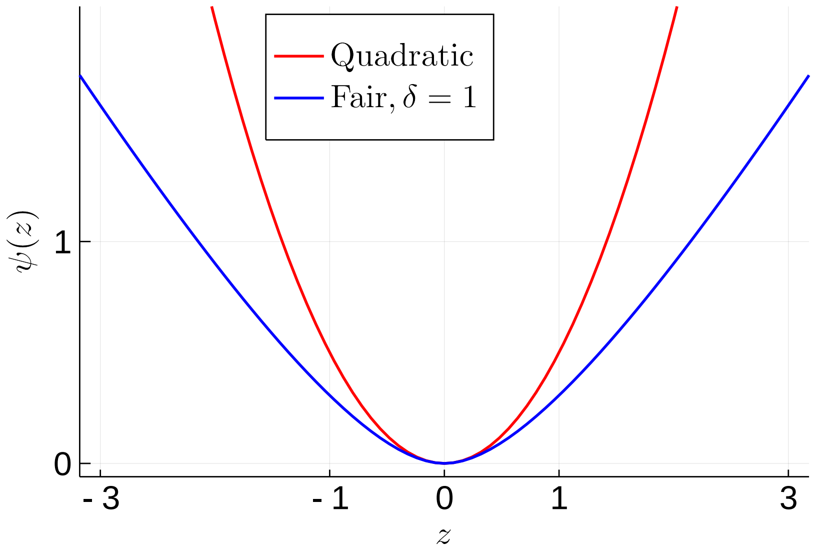
$$R(\boldsymbol{x}) = \sum_{k=1}^{K} \psi([\boldsymbol{C}\boldsymbol{x}]_k) \,.$$

Here ψ is a **potential function**. In (1.15), the potential function is quadratic: $\psi(z) = |z|^2 /2$, but for **edge-preserving** regularization we prefer potential functions that increase linearly rather than quadratically, such as the **Fair potential** [5–8]: $\psi(z) = \delta^2 \left(|z| /\delta - \log(1 + |z| /\delta)\right)$. (See figure below.)

We will focus first on the case where ψ is convex and smooth, and describe suitable iterative algorithms, including all of the algorithms considered for LS problems!

Example. This figure compares two potential functions:

- quadratic function $\psi(z) = |z|^2/2$
- **Fair potential** function $\psi(z) = \delta^2 \left(|z|/\delta - \log(1 + |z|/\delta)\right)$



Example. See p. 1.45 for an figure showing the benefits of non-quadratic regularization.

M-estimation for robustness to noise outliers in regression

We can rewrite the **least-squares** estimation problem as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \sum_{m=1}^{M} q([\boldsymbol{A}\boldsymbol{x}]_m - y_m), \quad q(z) = \frac{1}{2}|z|^2, \quad [\boldsymbol{A}\boldsymbol{x}]_m = \sum_{n=1}^{N} a_{mn} x_n.$$

As mentioned above, this estimator is the **maximum-likelihood** estimate if the noise is **Gaussian**, and hence is **asymptotically unbiased** and **asymptotically efficient** (see EECS 564) if $\boldsymbol{A}$ has full column rank, which are very desirable properties.

But often data is corrupted by **non-Gaussian noise**, particularly **heavy-tailed noise**, and the LS estimator is *not robust* to outliers in the data.

One popular method for **robust estimation** is to use an **M-estimator**

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \sum_{m=1}^{M} \psi([\boldsymbol{A}\boldsymbol{x}]_m - y_m), \tag{1.22}$$

where ψ is a (typically **convex** and **smooth** function that increases less rapidly than the quadratic function, making it more robust to outliers [9]. A particularly popular choice is the **Huber function**

$$\psi(z) = \begin{cases} \frac{1}{2}|z|^2, & |z| \leq \delta \\ \delta|z| - \delta^2/2, & |z| > \delta, \end{cases}$$

for some parameter $\delta > 0$. This choice has min-max optimality properties [10].

Under certain regularity conditions, one can show that the minimizer satisfies

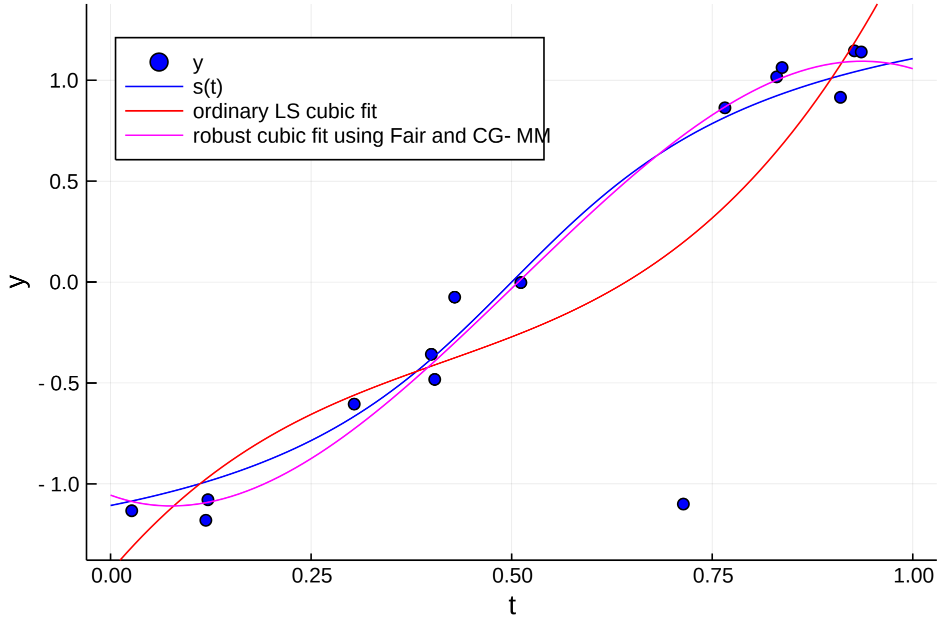
$$\hat{\boldsymbol{x}} = (\boldsymbol{A}'\boldsymbol{W}(\hat{\boldsymbol{x}})\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{W}(\hat{\boldsymbol{x}})\boldsymbol{y}, \quad \boldsymbol{W}(\boldsymbol{x}) \triangleq \mathsf{diag}\{\omega_\psi([\boldsymbol{A}\boldsymbol{x}]_m)\}, \quad \omega_\psi(z) \triangleq \frac{\dot{\psi}(z)}{z}.$$

This is a **recursive** expression or **implicit function**, not a closed-form solution for $\hat{\boldsymbol{x}}$ in terms of $\boldsymbol{y}$, so it is useful for insight but not directly for computation. However, we will later derive a closely related **majorize-minimize** (**MM**) algorithm of the form

$$\boldsymbol{x}_{t+1} = (\boldsymbol{A}'\boldsymbol{W}(\boldsymbol{x}_t)\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{W}(\boldsymbol{x}_t)\boldsymbol{y}, \quad t = 0, 1, 2, \ldots,$$

and show that it is convergent under mild assumptions on ψ when $\boldsymbol{A}$ has full column rank. Unfortunately this algorithm does not scale well to large-dimension problems, due to the matrix inverse. So we will also consider other more suitable algorithms.

Example. This figure compares an ordinary least-squares cubic polynomial fit using (1.14) with a robust M-estimator (1.22) using the Fair potential function, minimized using a CG algorithm. The robust fit is much less sensitive to the outlier and better agrees with the latent function.



1.4 Convex composite problems

Many modern SIPML problems involve minimizing **composite** cost functions of the form

$$\Psi(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}), \tag{1.23}$$

where $f(\boldsymbol{x})$ is convex and smooth and $g(\boldsymbol{x})$ is convex but **nonsmooth**, *i.e.*, not **differentiable**.

The two most important nonsmooth functions are

- the 1-norm $\|\cdot\|_1$ that arises in optimization problems involving sparsity models,
- the **characteristic function** for convex constraints as shown in (1.4).

There are numerous applications involving composite cost functions, and numerous minimization algorithms. Cost functions of this type will be a major theme in the class. Indeed this section is long because so many applications have this form. We focus here on applications involving the 1-norm,

ℓ_1 regularized problems

Probably the most ubiquitous **composite** cost function is:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) \triangleq \tag{1.24}$$

Here we focus on the unconstrained case, but there are also constrained variants.

We focus on the following application settings that motivate (1.24).

- statistics: regressors / covariate selection, related to **model selection**
  machine-learning: **feature selection**
- **sparse approximation** problems
- regularization using synthesis model assuming **sparsity**
  - signal/image **denoising**
  - **compressed sensing** [11]
  - **inverse problems**

Selecting features / covariates / regressors (LASSO)

Ordinary linear regression measurement model:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{A}$ is $M \times N$. When $N \ll M$ then usually $\boldsymbol{A}$ has full column rank and the LS estimate of $\boldsymbol{x}$ is fine. But when $N > M$ then the problem is **under-determined** and even if we use Tikhonov regularization (aka ridge regression) to improve the problem conditioning, the resulting model can be hard to *interpret*. For a more interpretable (parsimonious) model, often we prefer many of the $\boldsymbol{x}$ elements to be zero, so that only a subset of the features (columns of $\boldsymbol{A}$) are used for **prediction**.

A classical way to do this is called **subset selection** [12–14] and one formulation uses the K best features:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \,:\, \|\boldsymbol{x}\|_0 \leq K} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 , \tag{1.25}$$

where $K < N$ is the maximum number of features of interest. Alternative formulations are

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \text{ s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 < \varepsilon$$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_0 .$$

These optimization problems involving $\|\cdot\|_0$ are **nonconvex** and **NP hard**.

Thus, we often use the **convex relaxation** of one of the two latter formulations:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \text{ s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 < \varepsilon$$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 .$$

The latter has the form (1.24) and, in statistics, is called the **least absolute shrinkage and selection operator LASSO** [15] [16].

The **LASSO** leads to more interpretable prediction methods than ridge regression, because as β increases, more elements of $\hat{\boldsymbol{x}}$ become zero.

The first formulation (1.25) with the constraint $\|\boldsymbol{x}\|_0 \leq K$ is rarely relaxed to $\|\boldsymbol{x}\|_1 \leq c$ because we rarely have an upper bound for $\|\cdot\|_1$.

Sparse approximation

For typical **sparse approximation** problems we are given a wide matrix $\boldsymbol{D}$ called an **over-complete dictionary** and we want to approximate a signal $\boldsymbol{x}$ using a linear combination of a subset of the columns of $\boldsymbol{D}$, *i.e.*, we want $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}$ where $\boldsymbol{z}$ is a **sparse** coefficient vector. This problem is also called **basis pursuit**.

Again there are three natural **nonconvex** formulations for finding a **sparse** coefficient vector $\boldsymbol{z}$:

$$\hat{\boldsymbol{z}} = \mathop{\arg\min}_{\boldsymbol{z} \,:\, \|\boldsymbol{z}\|_0 \leq K} \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{x}\|_2^2 \qquad \hat{\boldsymbol{z}} = \mathop{\arg\min}_{\boldsymbol{z}} \|\boldsymbol{z}\|_0 \text{ s.t. } \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{x}\|_2^2 < \varepsilon$$

$$\hat{\boldsymbol{z}} = \mathop{\arg\min}_{\boldsymbol{z}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{x}\|_2^2 + \beta \|\boldsymbol{z}\|_0 .$$

Again it is typical to relax the latter problem to the following form (1.24) called **basis pursuit denoising**:

$$\hat{\boldsymbol{z}} = \mathop{\arg\min}_{\boldsymbol{z}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{x}\|_2^2 + \beta \|\boldsymbol{z}\|_1 . \qquad (1.26)$$

This problem looks so similar to subset/feature selection that one might wonder why even mention it?

- In feature selection, typically the matrix $\boldsymbol{A}$ contains *data* that is completely unstructured.
- For sparse approximation, often the dictionary $\boldsymbol{D}$ was designed mathematically, *e.g.*, using **wavelets**, and for certain assumptions about $\boldsymbol{D}$ there are mathematical guarantees about uniqueness and convergence of some iterative algorithms to a correct $\boldsymbol{z}$.
- Recent work relates sparse approximation to CNNs [17].

Signal models

Two primary applications of interest here are
- Denoising: $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\varepsilon} = \boldsymbol{I}\boldsymbol{x} + \boldsymbol{\varepsilon}$ where $M = N$
- Compressed sensing: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}$ where $M < N$

Solving these problems requires some form of **dimensionality reduction**.

EECS 551 focused on **subspace models**: $\boldsymbol{x} \approx \boldsymbol{B}\boldsymbol{c}$, assuming $\boldsymbol{x}$ is near $\mathcal{R}(\boldsymbol{B})$, where $\boldsymbol{B}$ is $N \times r$ with $r \ll N$.
Often we need richer models involving **sparsity**.

Two primary sparsity models are **synthesis sparsity** and **analysis sparsity**.
The terms come from introductory signals and systems, *e.g.*, the **discrete Fourier transform** (**DFT**):

$$c[k] = \sum_{n=0}^{N-1} x[n]\, \mathrm{e}^{-\imath 2\pi k n/N} \qquad \text{(analysis)} \qquad \boldsymbol{c} = \boldsymbol{W}\boldsymbol{x}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} c[k]\, \mathrm{e}^{-2\pi k n/N} \qquad \text{(synthesis)} \qquad \boldsymbol{x} = \boldsymbol{W}^{-1}\boldsymbol{c} = \frac{1}{N}\boldsymbol{W}'\boldsymbol{c},$$

where $\boldsymbol{W}$ is a $N \times N$ matrix having orthogonal columns, so $\boldsymbol{W}^{-1}$ is easily computed.
Here the columns of $\boldsymbol{W}^{-1}$ serve as a **basis** for $\mathbb{C}^N$.

For sparsity signal models we generalize such relationships to non-square matrices.

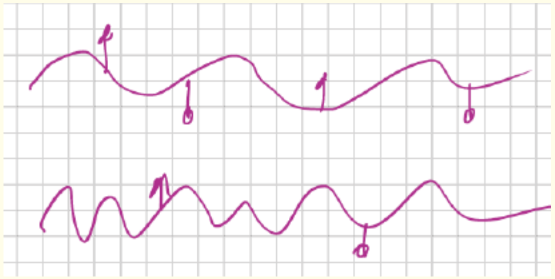
Synthesis sparsity with over-complete dictionary

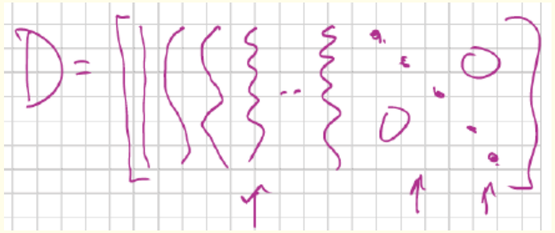
Consider signals that are approximately a couple (at most) sinusoids plus a few Kronecker impulses. For such “wave+spike” signals, a natural synthesis model is

$$\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}, \quad \boldsymbol{D} = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{I} \end{bmatrix}$$

where

- $\boldsymbol{Q}$ is the $N \times N$ (unitary) inverse **discrete cosine transform** (**DCT**) for which $\boldsymbol{Q}'\boldsymbol{Q} = \boldsymbol{I}_N$
- $\boldsymbol{I}$ is the $N \times N$ identity matrix for representing spikes
- $\boldsymbol{D}$ is $N \times K$ with $N < K = 2N$ (wide) and a **Parseval tight frame** (EECS 551)





Pseudo-inverse coefficients (from 551):

$$\hat{\boldsymbol{x}} = \underbrace{\boldsymbol{D}\hat{\boldsymbol{c}},}_{\text{synthesis}} \qquad \hat{\boldsymbol{c}} = \boldsymbol{D}^{+}\boldsymbol{x} = \boldsymbol{D}'\boldsymbol{x},$$

would provide an exact representation, but would not be sparse.

Instead we seek a sparse synthesis representation using (1.26):

$$\hat{\boldsymbol{x}} = \underbrace{\boldsymbol{D}\hat{\boldsymbol{z}},}_{\text{synthesis}} \qquad \hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{x}\|_2^2 + \beta \underbrace{\|\boldsymbol{z}\|_p}_{\hookrightarrow \text{ synthesis sparsity}},$$

where typically $p = 0$ or $p = 1$, for which only a few elements of $\hat{\boldsymbol{z}}$ are nonzero.

This is one of many applications using composite cost functions and 1-norm regularization.
Such sparse representations are also useful for **data compression**.

An in-class task and HW problem will explore the wave+spike model with Ch. 6.

The example above is considered a "hand crafted" signal model because we specified the dictionary $\boldsymbol{D}$ using signal components chosen mathematically. Later we will consider modern **adaptive** or **data-driven** methods (inspired by **machine learning**) where we learn $\boldsymbol{D}$ from **training data**.

We contrast the synthesis and analysis approach a bit later in the chapter.

Denoising using synthesis sparsity models

The previous section considered the case where we have a signal $\boldsymbol{x}$ in hand that we model as being sparse. Often we have a noisy version:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}, \qquad \boldsymbol{z} \text{ is sparse.}$$

This is called a **synthesis** model because we are synthesizing $\boldsymbol{x}$ using the linear combination $\boldsymbol{D}\boldsymbol{z}$.

We want to **denoise** the signal $\boldsymbol{y}$ and recover $\boldsymbol{x}$, using this sparsity model.

One way to estimate $\boldsymbol{x}$ is as follows

(1.27)

where typically $p = 0$ or $p = 1$. An alternative formulation is

(1.28)

Note the $\min$ over $\boldsymbol{z}$ here is not an "arg min."

A HW problem will examine relationships between such formulations.

Denoising with unitary bases

In signal and image processing, the most famous special case is when $\boldsymbol{D}$ is a **unitary matrix**, such as a discrete **orthogonal wavelet** transform. In this special case, because the Euclidean norm is **unitarily invariant**, an equivalent form of (1.27) is

(1.29)

This form has a closed-form solution for both $p = 0$ (hard thresholding) and $p = 1$ (soft thresholding).
For $p = 1$ the solution is

$$\hat{\boldsymbol{x}} = \boldsymbol{D}\,\text{soft.}(\boldsymbol{D}'\boldsymbol{y}, \beta)$$

where $\text{soft}(y, \beta) \triangleq \text{sign}(y) \max(|y| - \beta, 0)$.

For small-scale problems where $\boldsymbol{D}$ can be stored as a matrix, the following two lines of JULIA code look very similar to the math and solve (1.29) non-iteratively:

```
soft = (y,reg) -> sign(y) * max(abs(y) - reg, 0)
xh = D * soft.(D'*y, reg)
```

When $\boldsymbol{D}$ is not unitary, then solving (1.27) or (1.28) requires iterative algorithms in general.

Challenge. If $\boldsymbol{D}$ is a **Parseval tight frame**, does (1.27) simplify?

Compressed sensing with synthesis form sparsity

For **compressed sensing** the usual measurement model with $\boldsymbol{A} \in \mathbb{F}^{M \times N}$ is: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}$,
where "compressed" here means $M < N$, so the problem of finding $\boldsymbol{x}$ is under-determined.

Typical assumption is that $\boldsymbol{x}$ has a **sparse representation** or **sparse transformation**. We focus first on the **synthesis** form where one assumes $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}$ where the coefficient vector $\boldsymbol{z}$ is modeled as **sparse**.

As usual one can have nonconvex formulations using $\|\boldsymbol{z}\|_0$. Two typical convex formulations are

(1.30)

(1.31)

The latter allows $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}$, whereas the former enforces strict "synthesis" equality.
A HW problem examines the convexity of (1.31).

There are recovery guarantees for (1.30) under certain assumptions about the product matrix $\boldsymbol{A}\boldsymbol{D}$ that may not hold for your favorite application [18–21].

These approaches are popular for solving inverse problems even when the conditions are not guaranteed to hold, and often still described as "compressed sensing."

From nonsmooth to smooth

The challenge with minimizing nonsmooth cost functions like (1.24) is that $\|\cdot\|_1$ is not differentiable.

Consider the simplest form (1.24) of a 1-norm regularizer, repeated here:

$$\hat{\boldsymbol{x}} = \mathop{\arg\min}_{\boldsymbol{x} \in \mathbb{F}^N} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 . \tag{1.32}$$

When $\mathbb{F} = \mathbb{R}$, here is a technique for converting this nonsmooth cost function into a box-constrained smooth cost function [22, 23] for which one can use iterative algorithms designed for constrained problems like **gradient projection** method [24].

Let $\boldsymbol{u} = \max(\boldsymbol{x}, 0)$ and $\boldsymbol{v} = \max(-\boldsymbol{x}, 0)$ denote the positive and negative parts of $\boldsymbol{x}$ so that $\boldsymbol{x} = \boldsymbol{u} - \boldsymbol{v}$. Then

$$\|\boldsymbol{x}\|_1 = \mathbf{1}'\boldsymbol{u} + \mathbf{1}'\boldsymbol{v}$$

so (1.32) becomes the following bound-constrained quadratic program:

(1.33)

A HW problem explores this approach.

Elastic-net regularizer

(Read)

For problems where columns of $\boldsymbol{A}$ are highly correlated, the 1-norm regularizer has been reported to be suboptimal [25] and the following combination of the 1-norm and 2-norm, dubbed the **elastic net regularizer**, has been proposed:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 + \alpha \frac{1}{2} \|\boldsymbol{x}\|_2^2 , \tag{1.34}$$

where now one must tune both α and β somehow.

Although proposed originally for variable selection problems, it has also been used for inverse problems [26].

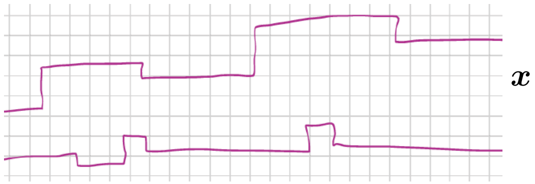
Although this might at first look like yet another cost function that requires its own iterative algorithms to minimize, one can combine the quadratic terms and simplify the **elastic net** problem into the form of the original LASSO problem (1.24). See HW.

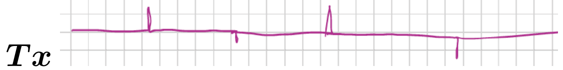
Analysis sparsity

So far we have focused on 1-norm regularizers where either $\boldsymbol{x}$ itself is **sparse**, or where $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{z}$ where $\boldsymbol{z}$ is sparse. These lead to "simple" regularizers of the form $\|\boldsymbol{x}\|_1$ and $\|\boldsymbol{z}\|_1$.

An alternative family of models assumes that some transformation of $\boldsymbol{x}$ is sparse, *e.g.*, one assumes $\boldsymbol{T}\boldsymbol{x}$ is sparse for some type of transformation matrix $\boldsymbol{T}$. Some evidence suggests analysis priors outperform synthesis priors [27].

Example. Consider 1D signals $\boldsymbol{x}$ that are piece-wise smooth or even piece-wise constant, as illustrated below. It is unclear what basis one would use for a parsimonious synthesis representation. But if we compute $\boldsymbol{T}\boldsymbol{x}$ using $\boldsymbol{T} = \boldsymbol{D}_N$ in (1.16), then $\boldsymbol{T}\boldsymbol{x}$ is sparse, as illustrated below.





Note that $\|\boldsymbol{T}\boldsymbol{x}\|_0 \ll N$ here, so this sparsifying transform provides a form of **dimensionality reduction**.

As usual there are both non-convex and convex variations of transform sparsity models, also called **analysis** regularizers. The most popular convex formulation is

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{T}\boldsymbol{x}\|_1 , \tag{1.35}$$

where the $\|\boldsymbol{T}\boldsymbol{x}\|_1$ term captures the model that $\boldsymbol{T}\boldsymbol{x}$ is sparse.

We write an ordinary 1-norm here, but in practice (*e.g.*, in clinical MRI scanners), a weighted norm may be used that weights/regularizes the high-frequency components more when $\boldsymbol{T}$ is a wavelet transform [28].

If $\boldsymbol{T}$ is an **invertible** matrix (*e.g.*, unitary), then we can define $\boldsymbol{z} = \boldsymbol{T}\boldsymbol{x}$ and rewrite (1.35) as

$$\hat{\boldsymbol{x}} = \boldsymbol{T}^{-1}\hat{\boldsymbol{z}}, \qquad \hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \left\|\boldsymbol{A}\boldsymbol{T}^{-1}\boldsymbol{z} - \boldsymbol{y}\right\|_2^2 + \beta \|\boldsymbol{z}\|_1 , \tag{1.36}$$

which is the simple 1-norm regularizer.

However, $\boldsymbol{T} = \boldsymbol{D}_N$ is not invertible, so that simplification is not directly applicable when using finite differences as the sparsifying transform.

Total variation (TV)

The most famous analysis regularizer is called **total variation** (**TV**) and in discrete-space problems corresponds to using the finite-difference matrix $\boldsymbol{T}$ in (1.16) or (1.17).

For 1D problems, the **TV regularizer** is simply $\|\boldsymbol{T}\boldsymbol{x}\|_1$, where $\boldsymbol{T}$ is the finite difference matrix in (1.16).
For 2D problems, the simplest form of TV regularization is called **anisotropic TV** and uses the stacked matrix in (1.17) as follows (*cf.* EECS 551 HW):

$$\|\boldsymbol{T}\boldsymbol{x}\|_1 = \left\| \begin{bmatrix} \boldsymbol{I}_M \otimes \boldsymbol{D}_N \\ \boldsymbol{D}_M \otimes \boldsymbol{I}_N \end{bmatrix} \boldsymbol{x} \right\|_1 = \|(\boldsymbol{I}_M \otimes \boldsymbol{D}_N)\,\boldsymbol{x}\|_1 + \|(\boldsymbol{D}_M \otimes \boldsymbol{I}_N)\,\boldsymbol{x}\|_1\,, \tag{1.37}$$

which is the sum of the absolute horizontal and vertical finite differences.

For a continuous-space 2D function $f(x,y)$, this **anisotropic TV** regularizer looks like

$$\iint \left|\frac{\partial}{\partial x} f(x,y)\right| + \left|\frac{\partial}{\partial y} f(x,y)\right| \mathrm{d}x\,\mathrm{d}y\,.$$

There is also an **isotropic TV** version that for a continuous-space 2D function $f(x,y)$ looks like

$$\iint \sqrt{\left(\frac{\partial}{\partial x} f(x,y)\right)^2 + \left(\frac{\partial}{\partial y} f(x,y)\right)^2}\, \mathrm{d}x\,\mathrm{d}y\,.$$

The discrete version is not easy to write in matrix form, but in summation form for a 2D function $f[m, n]$, the isotropic TV is roughly

$$\sum_m \sum_n \sqrt{(f[m,n] - f[m-1,n])^2 + (f[m,n] - f[m,n-1])^2}.$$

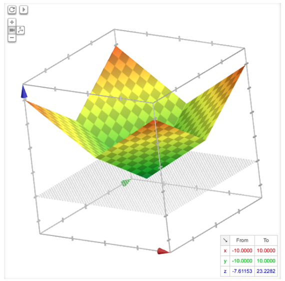
This is a convex function of f, but is nonsmooth due to the square root.

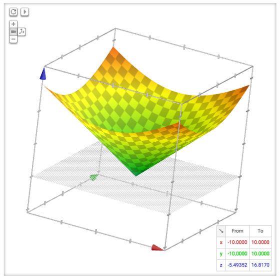
The simplified form (1.36) is applicable to

A: 1D TV using (1.16) B: 2D anisotropic TV using (1.37) C: both D: neither $\boxed{??}$

https://www.google.com/search?q=plot(abs(x)%2B(abs(y)) $|z_1| + |z_2|$

https://www.google.com/search?q=plot(sqrt(x^2%2By^2)) $\sqrt{|z_1|^2 + |z_2|^2}$

Anisotropic: 

Isotropic: 

Algorithms for composite optimization problems

The general problem (1.35) is quite challenging typically, and has motivated the development of numerous iterative algorithms and we will discuss many of them.

- interior-point methods [29]
- ADMM [30] [31]
- coordinate descent [32, 33]
- primal-dual methods [34]
- **proximal methods**, with inner iterations to compute the **proximal operator**
  - ISTA [35]
  - FISTA [36]
  - POGM [37, 38]
  - [39]

Converting 1D TV to LASSO

(Read)

In general the analysis problem (1.35) is harder to solve than the LASSO-type problem (1.24), due to the term $\|\boldsymbol{T}\boldsymbol{x}\|_1$. However, for the specific case where $\boldsymbol{T} = \boldsymbol{D}_N$, as defined in (1.16), there is a special trick (*cf.* [40]). Let $\boldsymbol{W} = \mathrm{diag}\{\boldsymbol{w}\}$ where $\boldsymbol{w} = \begin{bmatrix} 0 \\ \mathbf{1}_{N-1} \end{bmatrix}$ and define the (invertible!) $N \times N$ matrix

$$\boldsymbol{S} = \begin{bmatrix} 1 \ \mathbf{0}_{1\times N-1} \\ \boldsymbol{T} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ & & \ddots & \ddots & & \end{bmatrix} \Longrightarrow \boldsymbol{S}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & 1 & \ddots & \ddots \end{bmatrix}.$$

We can now rewrite (1.35) for this particular $\boldsymbol{T}$ as

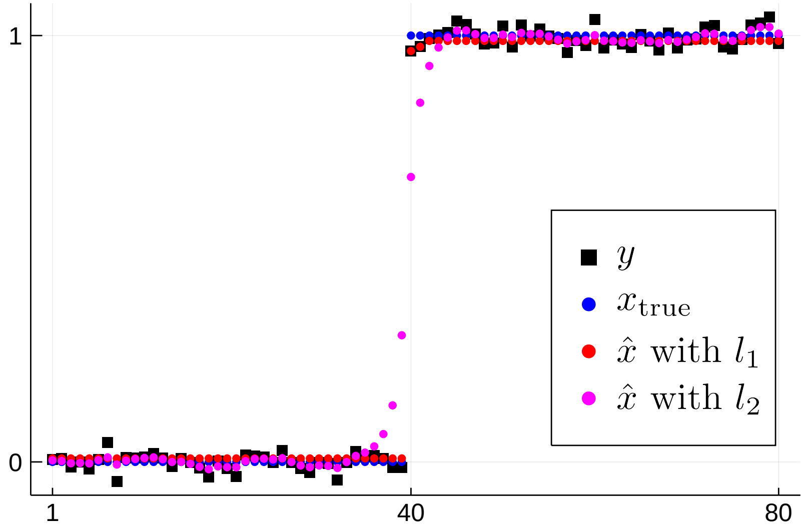
$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathbb{F}^N} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{W}\boldsymbol{S}\boldsymbol{x}\|_1 .$$

Now make the change of variables $\boldsymbol{z} = \boldsymbol{S}\boldsymbol{z}$, leading to the optimization problem

$$\hat{\boldsymbol{x}} = \boldsymbol{S}^{-1}\hat{\boldsymbol{z}}, \qquad \hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}\in\mathbb{F}^N} \frac{1}{2} \left\|\boldsymbol{A}\boldsymbol{S}^{-1}\boldsymbol{z} - \boldsymbol{y}\right\|_2^2 + \beta \|\boldsymbol{W}\boldsymbol{z}\|_1 . \tag{1.38}$$

Here, $\boldsymbol{S}^{-1}\boldsymbol{x} =$ `cumsum(x)` and $(\boldsymbol{S}^{-1})'\boldsymbol{x} =$ `reverse(cumsum(reverse(x)))`.

Example. Here is a 1D step-function signal with noise that is denoised using (1.38) with **POGM** from Ch. 5. The step function is well preserved using the 1-norm regularizer. In contrast, using the 2-norm regularizer leads to "blur" of the signal edge. The NRMSE of the original data $\boldsymbol{y}$, of $\hat{\boldsymbol{x}}$ with 1-norm and of $\hat{\boldsymbol{x}}$ with 2-norm are 2.9%, 1.6%, 8.0%, respectively.



One can apply a modified version of (1.33) to solve (1.38). (HW)
Unfortunately this approach does not seem to generalize to 2D TV.

To smooth or not smooth? (corner rounding) (Read)

Given the challenge of minimizing composite cost functions like those involving the 1-norm, one might wonder if a smooth approximation to the 1-norm would be sufficient in practice.

Often papers claim in the introduction to use TV or the 1-norm, but later in the details one sees approximations such as

$$|z| \approx \sqrt{z^2 + \epsilon^2} - \epsilon,$$

which is essentially the **hyperbola potential function**.

Another approximation is $|z| \approx \frac{1}{\delta} \psi(z; \delta)$, where $\psi(\cdot; \delta)$ is the **Huber function** or **Fair potential**.

Such approximations to the 1-norm are sometimes called **corner rounding**.

In low-dose CT reconstruction, the smooth version provides images that doctors find preferable [41].

Although such approximations are always differentiable (and typically have Lipschitz smooth derivatives), if δ is too small then $1/\delta \, \psi(z; \delta)$ has very high curvature (large **Lipschitz constant**). This property can cause slow convergence of conventional gradient-based algorithms. So algorithms designed for non-smooth problems can be useful even for smooth but "nearly non-smooth" cost functions.

Proximal operator for complex arguments (Read)

Section 6.3 of the EECS 551 notes describes the **proximal operator** and gives some examples involving $\mathbb{R}$. Here is an illustration of how to determine the proximal operator when using complex numbers. We focus on a 1D problem for simplicity. (More details in Ch. 5.) (This example should help in a HW problem.)

For $v \in \mathbb{C}$ and $\psi : \mathbb{R} \mapsto \mathbb{R}$, consider

$$\operatorname{prox}_\psi(v) = \arg\min_{z \in \mathbb{C}} \frac{1}{2} |v - z|^2 + \psi(|z|) .$$

To simplify this minimization problem, write $z \in \mathbb{C}$ in polar coordinates as $z = m\, \mathrm{e}^{\imath\phi}$ where $m \in [0, \infty)$ is the magnitude and $\phi \in \mathbb{R}$ is the phase. Then

$$\operatorname{prox}_\psi(v) = \hat{m}\, \mathrm{e}^{\imath\hat{\phi}} , \qquad (\hat{m}, \hat{\phi}) = \arg\min_{m \in [0,\infty),\ \phi \in \mathbb{R}} \frac{1}{2} \left| v - m\, \mathrm{e}^{\imath\phi} \right|^2 + \psi(m) .$$

We first consider the inner minimization over the phase ϕ:

$$\arg\min_{\phi \in \mathbb{R}} \frac{1}{2} \left| v - m\, \mathrm{e}^{\imath\phi} \right|^2 = \arg\min_{\phi \in \mathbb{R}} \frac{1}{2} \left| |v|\, \mathrm{e}^{\imath\angle v} - m\, \mathrm{e}^{\imath\phi} \right|^2 = \angle v.$$

Now consider the minimization over m:

$$\hat{m} = \mathop{\arg\min}_{m \in [0,\infty)} \frac{1}{2} \left| |v| \, \mathrm{e}^{\imath \angle v} - m \, \mathrm{e}^{\imath \hat{\phi}} \right|^2 + \psi(m) = \mathop{\arg\min}_{m \in [0,\infty)} \frac{1}{2} \left| |v| \, \mathrm{e}^{\imath \angle v} - m \, \mathrm{e}^{\imath \angle v} \right|^2 + \psi(m)$$

$$= \mathop{\arg\min}_{m \in [0,\infty)} \frac{1}{2} \left| |v| - m \right|^2 + \psi(m) = \mathrm{prox}_{\psi}(|v|) \, .$$

Combining, our final solution is

$$\mathrm{prox}_{\psi}(v) = \mathrm{e}^{\imath \angle v} \, \mathrm{prox}_{\psi}(|v|) \, .$$

So when the proximal operator depends only on the magnitude, as in $\psi(|z|)$ above, we simply determine the proximal operator of the magnitude $|v|$ of the argument, and then apply the `sign` at the end.

In JULIA code: `prox_complex(v) = sign(v) * prox_real(abs(v))`

Example. EECS 551 lecture notes derive the soft thresholding proximal mapping

$$\mathrm{prox}_{\beta|\cdot|}(\sigma) = \max(\sigma - \beta, 0)$$

when σ is real and nonnegative. The generalization to complex numbers is then

$$\mathrm{prox}_{\beta|\cdot|}(v) = \mathrm{sign}(v) \max(|v| - \beta, 0).$$

The generalization to complex vectors in JULIA code is:

```
soft(v,reg) = sign(v) * max(abs(v) - reg,0)
prox(v,reg) = soft.(v, reg)
```

1.5 Non-smooth problems

So far, all of the cost functions considered above have at least one smooth term in them, so gradient operations are at least partially applicable.

There are also applications where all terms in the cost function are non-smooth. Some applications are convex, and some are not.

Convex non-smooth problems

Robust regression with sparsity regularization (Read)

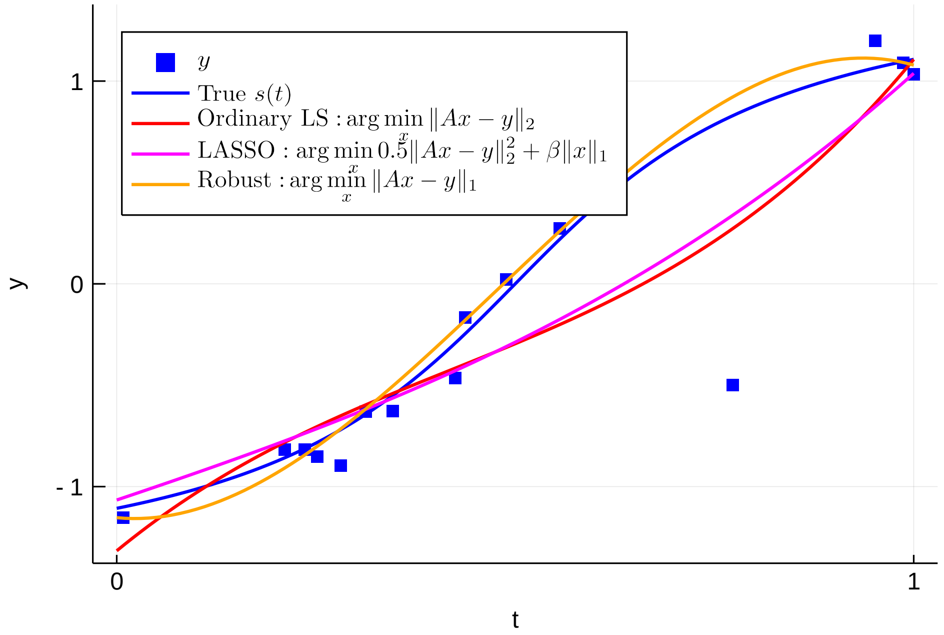
Many of the M-estimators on p. 1.24 involve smooth functions, but a particularly robust convex function is the 1-norm. Including sparsity based regularization leads to the following non-smooth optimization problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}), \quad \Psi(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_1 + \beta \|\boldsymbol{x}\|_1 . \tag{1.39}$$

This is a convex optimization problem, but the two non-smooth 1-norms make it challenging. The **subgradient method** and the **alternating direction method of multipliers** (**ADMM**) are the two algorithms we will discuss for this application.

See [42] for a linear programming approach and a Newton-like method.

Example. This figure shows polynomial regression from noisy samples of a smooth function, corrupted by an outlier point. Compared to ordinary LS estimation, the robust cost function (1.39) yields a fit that better agrees with the underlying truth.



Supervised machine learning for binary classification (Read)

(Mostly review from EECS 551.)

To learn weights $\boldsymbol{x} \in \mathbb{R}^N$ of a **binary classifier** given feature vectors $\{\boldsymbol{v}_i\} \in \mathbb{R}^N$ (training data) and labels $\{y_i = \pm 1 \ : \ i = 1, \ldots, M\}$, we can minimize a cost function with a regularization parameter $\beta > 0$:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}), \qquad \Psi(\boldsymbol{x}) = \sum_{i=1}^M h(y_i \langle \boldsymbol{x},\, \boldsymbol{v}_i \rangle) + \beta \sum_{n=1}^N \psi(x_n) = \mathbf{1}'_M h.(\boldsymbol{A}\boldsymbol{x}) + \beta \mathbf{1}'_N \psi\,.(\boldsymbol{x}), \tag{1.40}$$

where the mth row of the $M \times N$ matrix $\boldsymbol{A}$ is $y_m \boldsymbol{v}_m^T$, and where the second term is a regularizer like $\mathbf{1}'_N \psi\,.(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ when $\psi(z) = |z|$. Such a regularization term is especially important in the typical case where the feature vector dimension N is large relative to the sample size M. **Tikhonov regularization**, where $\psi(z) = z^2/2$, is also common.

For good classification accuracy using the (almost linear) classifier $\operatorname{sign}(\langle \boldsymbol{x},\, \boldsymbol{v}_i \rangle)$, we want:

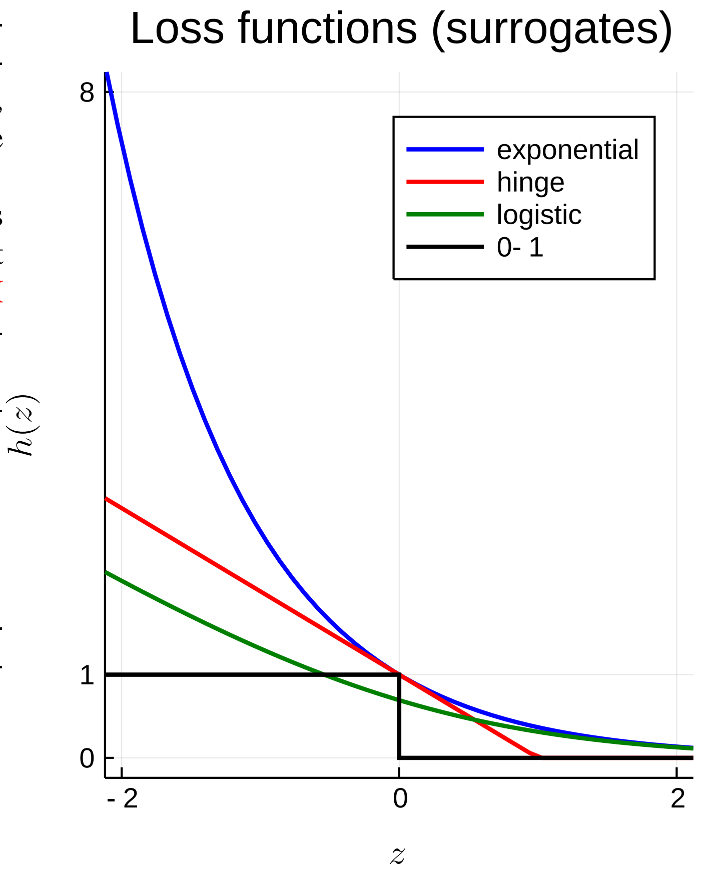
- $\langle \boldsymbol{x},\, \boldsymbol{v}_i \rangle > 0$ if $y_i = +1$ and
- $\langle \boldsymbol{x},\, \boldsymbol{v}_i \rangle < 0$ if $y_i = -1$,
- *i.e.*, $\langle \boldsymbol{x},\, y_i \boldsymbol{v}_i \rangle > 0$, or equivalently $[\boldsymbol{A}\boldsymbol{x}]_i > 0$.

Solving (1.40) is related to classification using a **support-vector machine** (**SVM**). SVM descriptions often include a learned offset parameter b in the data term by using $h(y_i(\langle \boldsymbol{x},\, \boldsymbol{v}_i \rangle - b))$. By replacing $\boldsymbol{v}_i$ with $(\boldsymbol{v}_i, 1)$, the effect of b is absorbed into the inner product, where x_N (or x_{N+1}) has the role of b, so the form (1.40) is sufficiently general to include an offset.

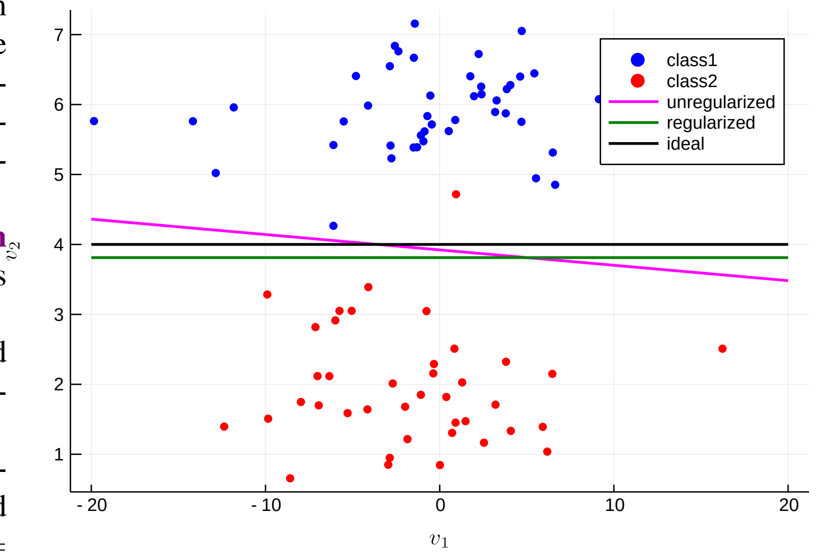
- The 0-1 loss function $h(z) = \mathbb{I}_{\{z \leq 0\}}$ is natural because it counts how many training samples are misclassified, but it is nonconvex and nondifferentiable, so very difficult to use for optimization. Instead one usually uses **surrogate loss functions**.
- The **hinge loss** function $h(z) = \max(1 - z, 0)$ is related to the **soft-margin SVM** and is convex, but non-differentiable. We will use **ADMM** and **SGM** for it. Here the data-fit term turns out to be proportional to the distance to the margin.
- The **logistic** loss function is convex and has a Lipschitz continuous derivative:

$$\psi(z) = \log\left(1 + e^{-z}\right).$$

  We will use gradient-based methods for it.
- The exponential loss function is convex and differentiable, but its derivative is not Lipschitz continuous. We will not consider it further.



Example. This figure illustrates the design of a binary classifier using (1.40) with the **hinge loss** function. This is a synthetic example designed to illustrate in 2D the potential benefits of sparsity-based regularization. Here $N = 3$ and $\boldsymbol{v} = [v_1, v_2, 1]$.
The black line shows the ideal **decision boundary** for separating the two classes (based on the ensemble distribution).
The magenta line shows the unregularized design where $\beta = 0$. There is an unnecessary dependence on feature v_1.
The green line shows the decision boundary when $\beta = 23$, and where we used a weighted 1-norm $\|\boldsymbol{W}\boldsymbol{x}\|_1$ where $\boldsymbol{W} = \text{diag}\{\boldsymbol{w}\}$ and here $\boldsymbol{w} = [1, 1, 0]$ so that we avoid shrinking the offset term.

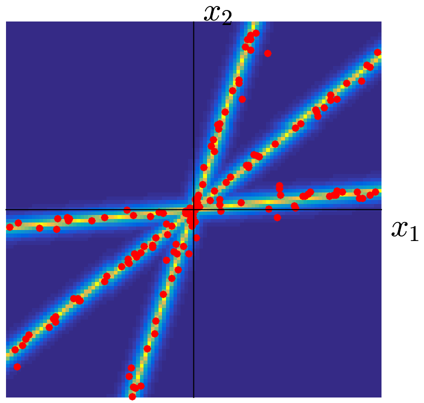


Note that when we use the hinge loss and a 1-norm regularizer, every term in the the cost function (1.40) is nonsmooth. So Ψ is not of the composite form (1.23) and specialized optimization algorithms are required. The above example used **ADMM**; see Ch. 6.

Unsupervised clustering using a union of subspaces (Read)

The previous binary classification application is a **supervised** machine learning problem because we are given labeled training data. Some machine learning problems are **unsupervised**, such as **clustering**, where we are given a set of vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, each in $\mathbb{F}^M$, and we want to assign each of them to one of, say, K classes. There are many unsupervised clustering methods (see EECS 545).

One approach is to assume each $\boldsymbol{x}_n$ lies approximately in a **union of subspaces**, as illustrated in the figure for $M = 2$. One union-of-subspaces approach builds upon **sparse subspace clustering** methods [43–47]. If we expect that the columns of the $M \times N$ matrix $\boldsymbol{X} \triangleq \begin{bmatrix} \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_N \end{bmatrix}$ lie in a union of subspaces, then we should be able to express any column of $\boldsymbol{X}$ in terms of a linear combination of a small (say K) number of other columns of $\boldsymbol{X}$. In other words, $\boldsymbol{X}$ should have **self similarity**:



$$\boldsymbol{X} \approx \boldsymbol{X}\boldsymbol{Z},$$

where $\boldsymbol{Z}$ is a $N \times N$ matrix in the (nonconvex) set of matrices with K-sparse columns:

$$\mathcal{Z} \triangleq \mathcal{Z}_0 \cap \mathcal{Z}_K, \quad \mathcal{Z}_0 \triangleq \left\{ \boldsymbol{Z} \in \mathbb{F}^{N \times N} \;:\; Z_{nn} = 0, \; \forall n \right\}, \quad \mathcal{Z}_K \triangleq \left\{ \boldsymbol{Z} \in \mathbb{F}^{N \times N} \;:\; \left\| \boldsymbol{Z}_{:,n} \right\|_0 \leq K, \; \forall n \right\}.$$

What should K be in the above figure?

A: 0 B: 1 C: 2 D: 3 E: 4 $\boxed{??}$

One could try to learn the coefficients $\boldsymbol{Z}$ by tackling the following hard nonconvex optimization problem:

$$\hat{\boldsymbol{Z}} \triangleq \underset{\boldsymbol{Z}\in\mathcal{Z}}{\arg\min}\, \|\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{X}\|_{\mathrm{F}}^2.$$

In the absence of noise, one could use the following convex optimization problem [47]:

$$\hat{\boldsymbol{Z}} \triangleq \underset{\boldsymbol{Z}\in\mathcal{Z}_0}{\arg\min}\, \|\mathrm{vec}(\boldsymbol{Z})\|_1 \text{ s.t. } \boldsymbol{X} = \boldsymbol{X}\boldsymbol{Z}.$$

In the presence of noise, a convex cost function that balances the **self similarity** criterion with sparsity may be preferable:

$$\hat{\boldsymbol{Z}} \triangleq \underset{\boldsymbol{Z}\in\mathcal{Z}_0}{\arg\min}\, \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{Z}\|_{\mathrm{F}}^2 + \beta\, \|\mathrm{vec}(\boldsymbol{Z})\|_1 .$$

After computing $\hat{\boldsymbol{Z}}$, one can apply **spectral clustering** [48, 49] to it to complete the unsupervised clustering.

We will explore such optimization problems for **unsupervised machine learning** in HW problems.

1.6 Summary

This chapter has focused on applications involving convex optimization problems (but not exclusively so), most of which require iterative algorithms to solve.
Signal processing applications considered include: compressed sensing, sparse representation, sparse filter design, regularized inverse problems sparse approximation, data compression, signal denoising, and TV regularization,
Machine learning applications considered include: linear regression, ridge regression, robust regression, sparse regression, supervised binary classification, and unsupervised clustering.

The next chapter has more applications involving nonconvex optimization problems.

Bibliography

[1] T. Baran, D. Wei, and A. V. Oppenheim. "Linear programming algorithms for sparse filter design". In: *IEEE Trans. Sig. Proc.* 58.3 (Mar. 2010), 1605–17 (cit. on p. 1.8).

[2] S. Boyd and L. Vandenberghe. *Convex optimization*. UK: Cambridge, 2004 (cit. on p. 1.8).

[3] H. Sun, D. S. Weller, A. Chu, S. Ramani, D. Yoon, J. F. Nielsen, and J. A. Fessler. "Spoke pulse design in magnetic resonance imaging using greedy minimax algorithm". In: *Proc. IEEE Intl. Symp. Biomed. Imag.* 2013, 696–9 (cit. on p. 1.9).

[4] I. Dunning, J. Huchette, and M. Lubin. "JuMP: A modeling language for mathematical optimization". In: *SIAM Review* 59.2 (2017), 295–320 (cit. on p. 1.9).

[5] R. C. Fair. "On the robust estimation of econometric models". In: *Ann. Econ. Social Measurement* 2 (Oct. 1974), 667–77 (cit. on p. 1.22).

[6] P. W. Holland and R. E. Welsch. "Robust regression using iteratively reweighted least-squares". In: *Comm. in Statistics—Theory and Methods* 6.9 (1977), 813–27 (cit. on p. 1.22).

[7] W. J. J. Rey. *Introduction to robust and quasi-robust statistical methods*. Berlin: Springer, 1983 (cit. on p. 1.22).

[8] K. Lange. "Convergence of EM image reconstruction algorithms with Gibbs smoothing". In: *IEEE Trans. Med. Imag.* 9.4 (Dec. 1990). Corrections, T-MI, 10:2(288), June 1991., 439–46 (cit. on p. 1.22).

[9] P. J. Huber. *Robust statistics*. New York: Wiley, 1981 (cit. on p. 1.24).

[10] P. J. Huber. "Robust estimation of a location parameter". In: *Ann. Math. Stat.* 35.1 (Mar. 1964), 73–101 (cit. on p. 1.25).

[11] K. Bryan and T. Leise. "Making do with less: an introduction to compressed sensing". In: *SIAM Review* 55.3 (2013), 547–66 (cit. on p. 1.28).

[12] C. L. Mallows. "Some comments on C_p". In: *Technometrics* 15.4 (Nov. 1973), 661–75 (cit. on p. 1.28).

[13] L. Breiman. "Better subset regression using the nonnegative garrote". In: *Technometrics* 37.4 (Nov. 1995), 373–84 (cit. on p. 1.28).

[14] C. Couvreur and Y. Bresler. "On the optimality of the backward greedy algorithm for the subset selection problem". In: *SIAM J. Matrix. Anal. Appl.* 21.3 (2000), 797–808 (cit. on p. 1.28).

[15] R. Tibshirani. "Regression shrinkage and selection via the LASSO". In: *J. Royal Stat. Soc. Ser. B* 58.1 (1996), 267–88 (cit. on p. 1.29).

[16] E. J. Candes and Y. Plan. "Near-ideal model selection by ℓ_1 minimization". In: *Ann. Stat.* 37.5a (2009), 2145–77 (cit. on p. 1.29).

[17] V. Papyan, Y. Romano, and M. Elad. "Convolutional neural networks analyzed via convolutional sparse coding". In: *J. Mach. Learning Res.* 18.83 (2017), 1–52 (cit. on p. 1.30).

[18] E. J. Candes, J. Romberg, and T. Tao. "Stable signal recovery from incomplete and inaccurate measurements". In: *Comm. Pure Appl. Math.* 59.8 (2006), 1207–23 (cit. on p. 1.36).

[19] E. J. Candes, J. Romberg, and T. Tao. "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information". In: *IEEE Trans. Info. Theory* 52.2 (Feb. 2006), 489–509 (cit. on p. 1.36).

[20] R. G. Baraniuk, V. Cevher, and M. B. Wakin. "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective". In: *Proc. IEEE* 98.6 (June 2010), 959–971 (cit. on p. 1.36).

[21] D. L. Donoho and J. Tanner. "Precise undersampling theorems". In: *Proc. IEEE* 98.6 (June 2010), 913–924 (cit. on p. 1.36).

[22] R. M. Leahy and B. D. Jeffs. "On the design of maximally sparse beamforming arrays". In: *IEEE Trans. Attenas. Propagat.* 39.8 (Aug. 1991), 1178–87 (cit. on p. 1.37).

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders. "Atomic decomposition by basis pursuit". In: *SIAM J. Sci. Comp.* 20.1 (1998), 33–61 (cit. on p. 1.37).

[24] M. Figueiredo, R. Nowak, and S. J. Wright. "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems". In: *IEEE J. Sel. Top. Sig. Proc.* 1.4 (Dec. 2007), 586–97 (cit. on p. 1.37).

[25] H. Zou and T. Hastie. "Regularization and variable selection via the elastic net". In: *J. Royal Stat. Soc. Ser. B* 67.2 (2005), 301–20 (cit. on p. 1.38).

[26] T. Nguyen-Duc and W-K. Jeong. "Compressed sensing dynamic MRI reconstruction using multi-scale 3D convolutional sparse coding with elastic net regularization". In: *Proc. IEEE Intl. Symp. Biomed. Imag.* 2018, 332–5 (cit. on p. 1.38).

[27] M. Elad, P. Milanfar, and R. Rubinstein. "Analysis versus synthesis in signal priors". In: *Inverse Prob.* 23.3 (June 2007), 947–68 (cit. on p. 1.39).

[28] J. Liu, J. Rapin, T. Chang, A. Lefebvre, M. Zenge, E. Mueller, and M. S. Nadar. "Dynamic cardiac MRI reconstruction with weighted redundant Haar wavelets". In: *Proc. Intl. Soc. Mag. Res. Med.* 2012, p. 4249 (cit. on p. 1.40).

[29] S-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. "An interior-point method for large-scale l_1-regularized least squares". In: *IEEE J. Sel. Top. Sig. Proc.* 1.4 (Dec. 2007), 606–17 (cit. on p. 1.43).

[30] J. Eckstein and W. Yao. "Relative-error approximate versions of Douglas-Rachford splitting and special cases of the ADMM". In: *Mathematical Programming* 170.2 (Aug. 2018), 417–44 (cit. on p. 1.43).

[31] D. Kim. *Accelerated proximal point method for maximally monotone operators*. 2019 (cit. on p. 1.43).

[32] T. T. Wu and K. Lange. "Coordinate descent algorithms for lasso penalized regression". In: *annapplstat* 2.1 (2008), 224–44 (cit. on p. 1.43).

[33] Y. Li and S. Osher. "Coordinate descent optimization for l_1 minimization with application to compressed sensing; a greedy algorithm". In: *Inverse Prob. and Imaging* 3.3 (Aug. 2009), 487–503 (cit. on p. 1.43).

[34] A. Chambolle and T. Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *J. Math. Im. Vision* 40.1 (2011), 120–145 (cit. on p. 1.43).

[35] I. Daubechies, M. Defrise, and C. De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Comm. Pure Appl. Math.* 57.11 (Nov. 2004), 1413–57 (cit. on p. 1.43).

[36] A. Beck and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM J. Imaging Sci.* 2.1 (2009), 183–202 (cit. on p. 1.43).

[37] A. B. Taylor, J. M. Hendrickx, and Francois Glineur. "Exact worst-case performance of first-order methods for composite convex optimization". In: *SIAM J. Optim.* 27.3 (Jan. 2017), 1283–313 (cit. on p. 1.43).

[38] D. Kim and J. A. Fessler. "Adaptive restart of the optimized gradient method for convex optimization". In: *J. Optim. Theory Appl.* 178.1 (July 2018), 240–63 (cit. on p. 1.43).

[39] A. S. Lewis and S. J. Wright. “A proximal method for composite minimization”. In: *Mathematical Programming* 158.1 (July 2016), 501–46 (cit. on p. 1.43).

[40] V. M. Patel, R. Maleh, A. C. Gilbert, and R. Chellappa. “Gradient-based image recovery methods from incomplete Fourier measurements”. In: *IEEE Trans. Im. Proc.* 21.1 (Jan. 2012), 94–105 (cit. on p. 1.44).

[41] J-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh. “A three-dimensional statistical approach to improved image quality for multi-slice helical CT”. In: *Med. Phys.* 34.11 (Nov. 2007), 4526–44 (cit. on p. 1.46).

[42] T. F. Coleman and Y. Li. “A globally and quadratically convergent affine scaling method for linear l_1 problems”. In: *Mathematical Programming* 56 (Aug. 1992), 189–222 (cit. on p. 1.49).

[43] E. Elhamifar and R. Vidal. “Sparse subspace clustering”. In: *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*. Vol. 2790-7. 2009 (cit. on p. 1.54).

[44] R. Vidal. “Subspace clustering”. In: *IEEE Sig. Proc. Mag.* 28.2 (Mar. 2011), 52–68 (cit. on p. 1.54).

[45] M. Soltanolkotabi and E. J. Candes. “A geometric analysis of subspace clustering with outliers”. In: *Ann. Stat.* 40.4 (2012), 2195–238 (cit. on p. 1.54).

[46] A. Adler, M. Elad, and Y. Hel-Or. “Probabilistic subspace clustering via sparse representations”. In: *IEEE Signal Proc. Letters* 20.1 (Jan. 2013), 63–6 (cit. on p. 1.54).

[47] E. Elhamifar and R. Vidal. “Sparse subspace clustering: algorithm, theory, and applications”. In: *IEEE Trans. Patt. Anal. Mach. Int.* 35.11 (Nov. 2013), 2765–81 (cit. on pp. 1.54, 1.55).

[48] A. Ng, Y. Weiss, and M. Jordan. “On spectral clustering: analysis and an algorithm”. In: *Neural Info. Proc. Sys.* 2001, 849–56 (cit. on p. 1.55).

[49] U. von Luxburg. “A tutorial on spectral clustering”. In: *statcomp* 17.4 (2007), 395–416 (cit. on p. 1.55).