Introduction: Why Optimization?

Ryan Tibshirani
Convex Optimization 10-725

Course setup

Welcome to our course on Convex Optimization, with a focus on its ties to Machine Learning and Statistics!

Basic adminstrative details:

- Instructor: Ryan Tibshirani
- Education associate: Daniel Bird
- Teaching assistants: Chen Dan, William Guss, Po-Wei Wang, Lingxiao Zhao
- Course website: `http://www.stat.cmu.edu/~ryantibs/convexopt/`
- We will use Piazza for announcements and discussions
- We will Canvas just as a gradebook

Prerequisites: no formal ones, but class will be fairly fast paced

Assume working knowledge of/proficiency with:

- Real analysis, calculus, linear algebra
- Core problems in Machine Learning and Statistics
- Programming (R, Python, Julia, your choice ...)
- Data structures, computational complexity
- Formal mathematical thinking

If you fall short on any one of these things, it's certainly possible to catch up; but don't hesitate to talk to us

Evaluation:

- 6 homeworks
- 6 quizzes
- 1 little test

Quizzes: due at the same time as each homework. These should be pretty easy if you've attended lectures ...

Little test: same format as quizzes. Will be comprehensive (you are allowed one sheet of notes)

Scribing: sign up to scribe one lecture per semester, on the course website (multiple scribes per lecture). Can bump up your grade in boundary cases

Auditors: welcome, please audit rather than just sitting in

Heads up: class will not be easy, but should be worth it ... !

Optimization in Machine Learning and Statistics

Optimization problems underlie nearly everything we do in Machine Learning and Statistics. In other courses, you learn how to:

translate  into $P : \min_{x \in D} f(x)$

Conceptual idea *Optimization problem*

Examples of this? Examples of the contrary?

This course: how to solve P, and why this is a good skill to have

Motivation: why do we bother?

Presumably, other people have already figured out how to solve

$$P \; : \; \min_{x \in D} \; f(x)$$

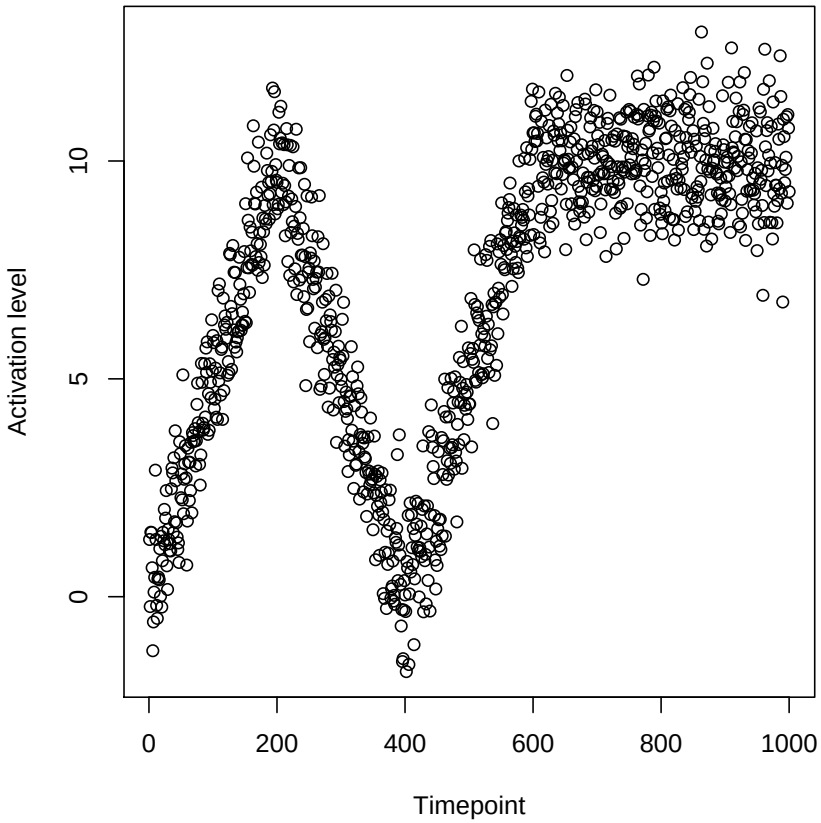
So why bother? Many reasons. Here's three:

1. Different algorithms can perform better or worse for different problems P (sometimes drastically so)
2. Studying P through an optimization lens can actually give you a deeper understanding of the task/procedure at hand
3. Knowledge of optimization can actually help you create a new problem P that is even more interesting/useful

Optimization moves quickly as a field. But there is still much room for progress, especially its intersection with ML and Stats

Example: algorithms for linear trend filtering

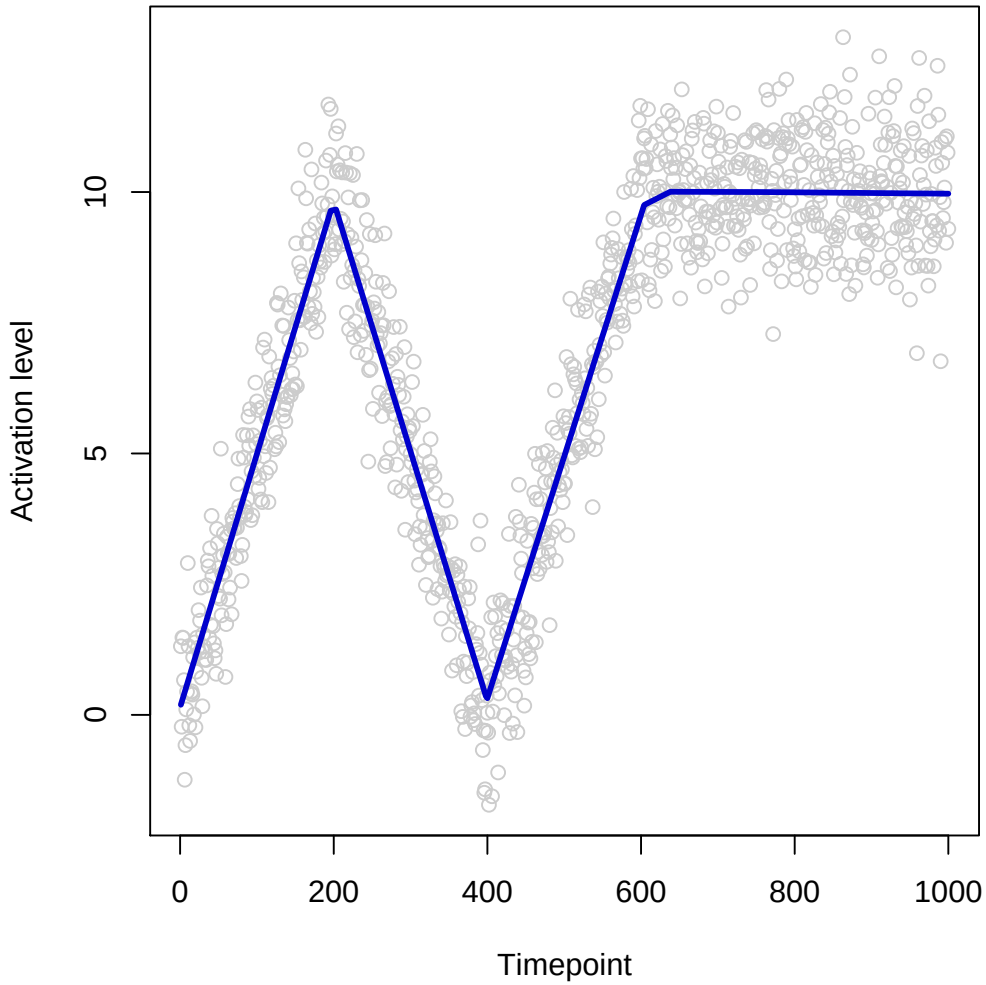
Given observations $y_i \in \mathbb{R}$, $i = 1, \ldots, n$ corresponding to underlying locations $x_i = i$, $i = 1, \ldots, n$



Linear trend filtering fits a piecewise linear function, with adaptively chosen knots (Steidl et al. 2006, Kim et al. 2009)

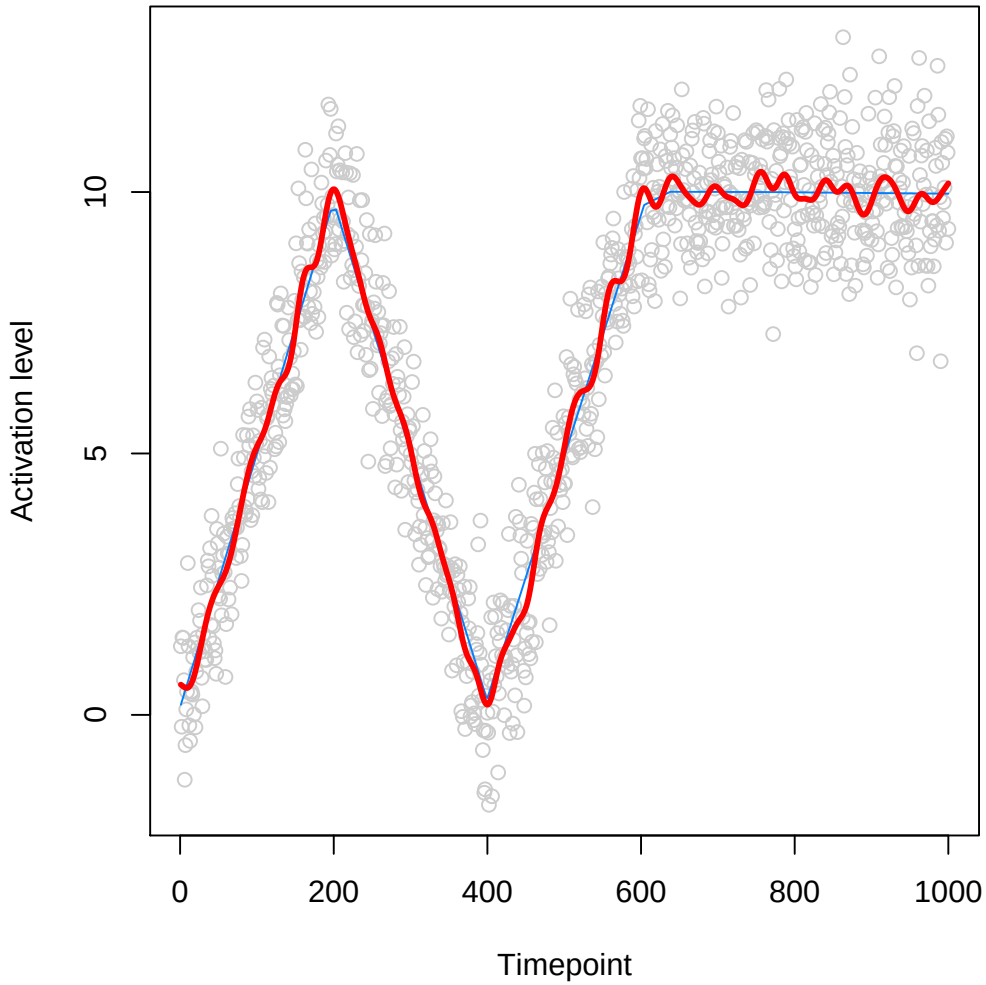
How? By solving $\min_\theta \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-2} |\theta_i - 2\theta_{i+1} + \theta_{i+2}|$

Problem: $$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\lambda\sum_{i=1}^{n-2}|\theta_i-2\theta_{i+1}+\theta_{i+2}|$$



Interior point method,
20 iterations

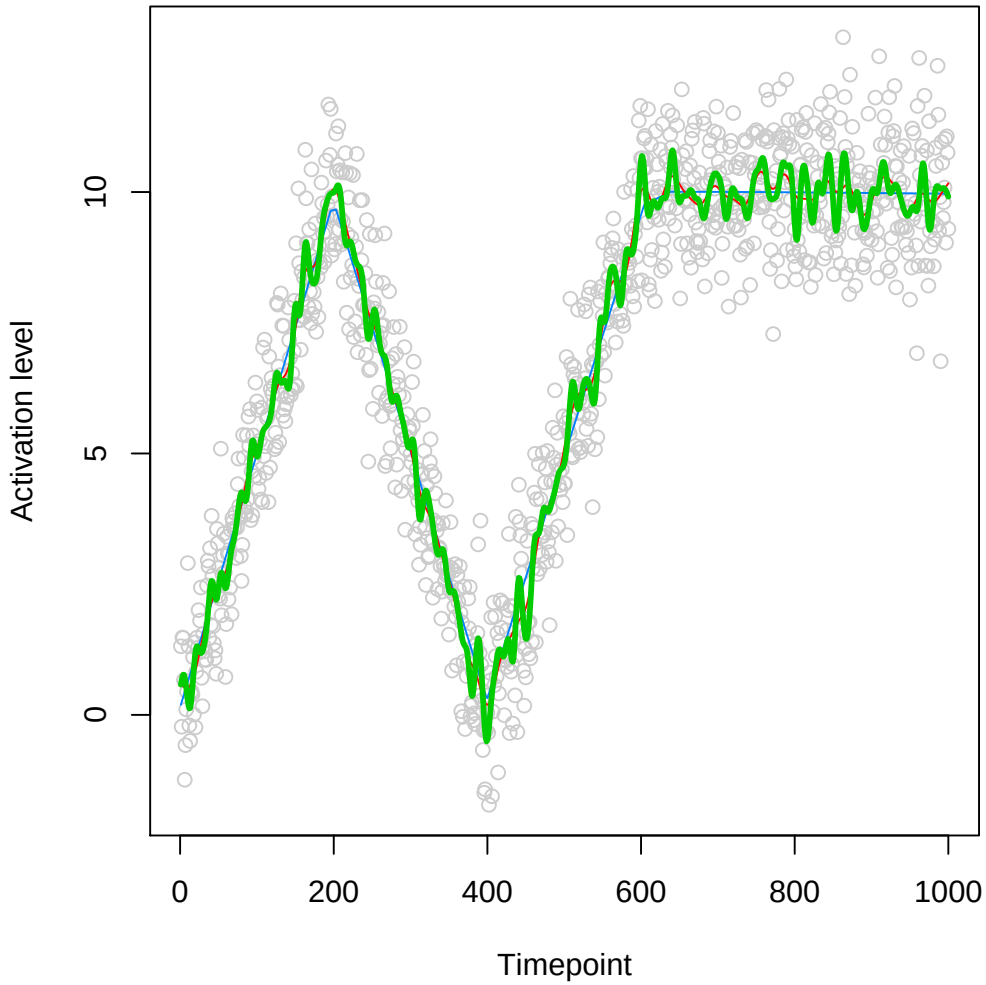
Problem: $$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-2} |\theta_i - 2\theta_{i+1} + \theta_{i+2}|$$



Interior point method, 20 iterations

Proximal gradient descent, 10K iterations

Problem: $$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\lambda\sum_{i=1}^{n-2}|\theta_i-2\theta_{i+1}+\theta_{i+2}|$$

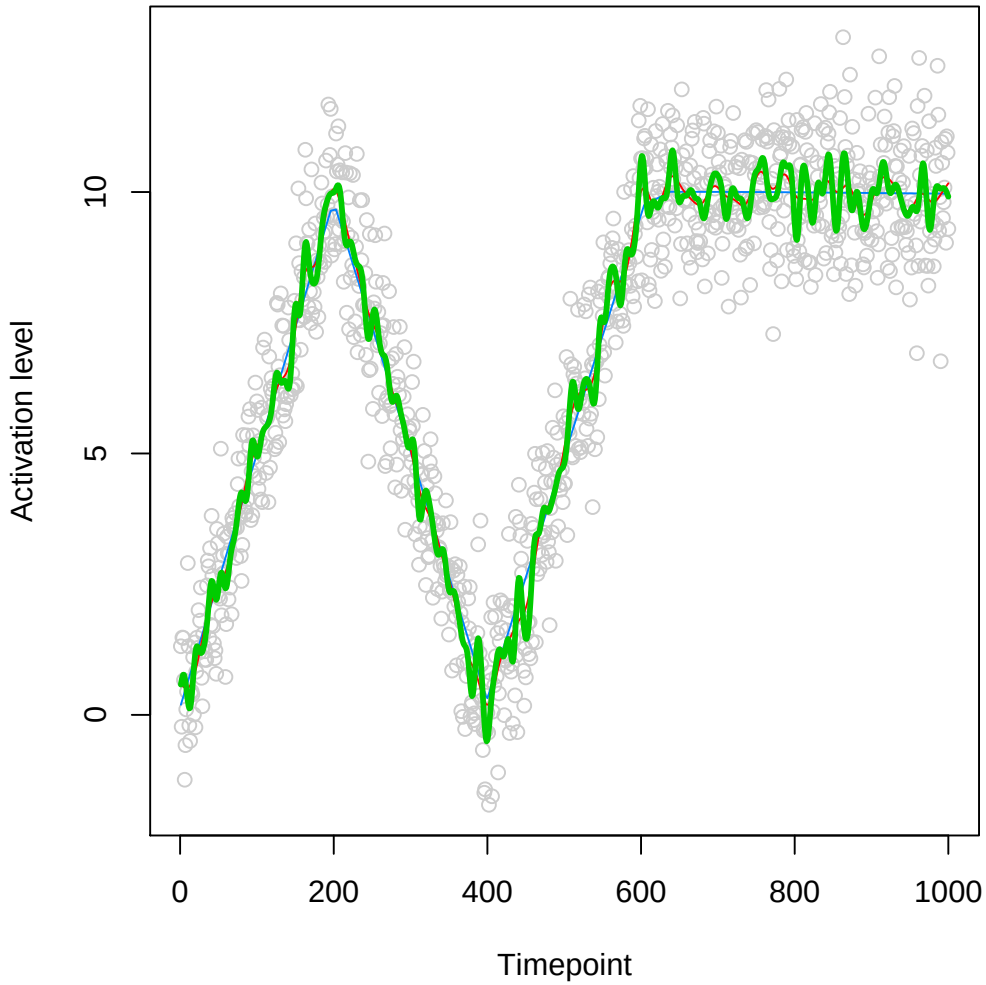


Interior point method, 20 iterations

Proximal gradient descent, 10K iterations

Coordinate descent, 1000 cycles

Problem: $$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\lambda\sum_{i=1}^{n-2}|\theta_i-2\theta_{i+1}+\theta_{i+2}|$$



Interior point method, 20 iterations

Proximal gradient descent, 10K iterations

Coordinate descent, 1000 cycles

(all from the dual)

What's the message here?

So what's the right conclusion here?

Is primal-dual interior point method simply a better method than proximal gradient descent, coordinate descent? ... No

In fact, different algorithms will work better in different situations. We'll learn details throughout the course

In the linear trend filtering problem:

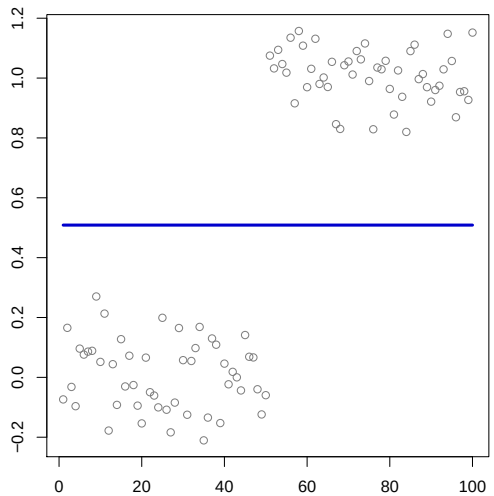
- Primal-dual: fast (structured linear systems)
- Proximal gradient: slow (conditioning)
- Coordinate descent: slow (large active set)

Example: changepoints in the fused lasso

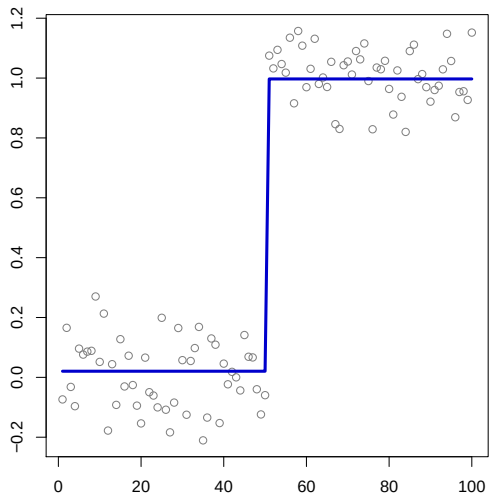
The fused lasso or total variation denoising problem:

$$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-1} |\theta_i - \theta_{i+1}|$$

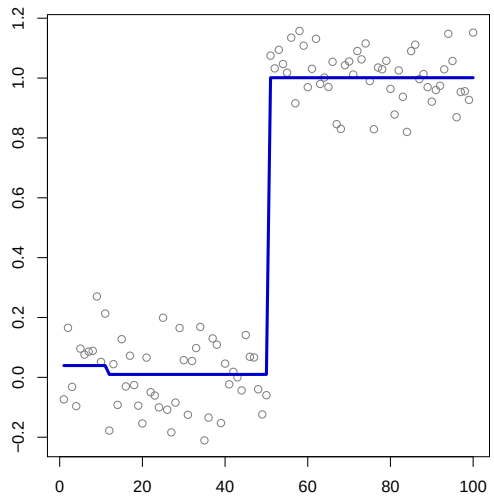
This fits a piecewise constant function, given data y_i, $i = 1, \ldots, n$. As tuning parameter λ decreases, we see more changepoints in the solution $\hat{\theta}$



$\lambda = 25$

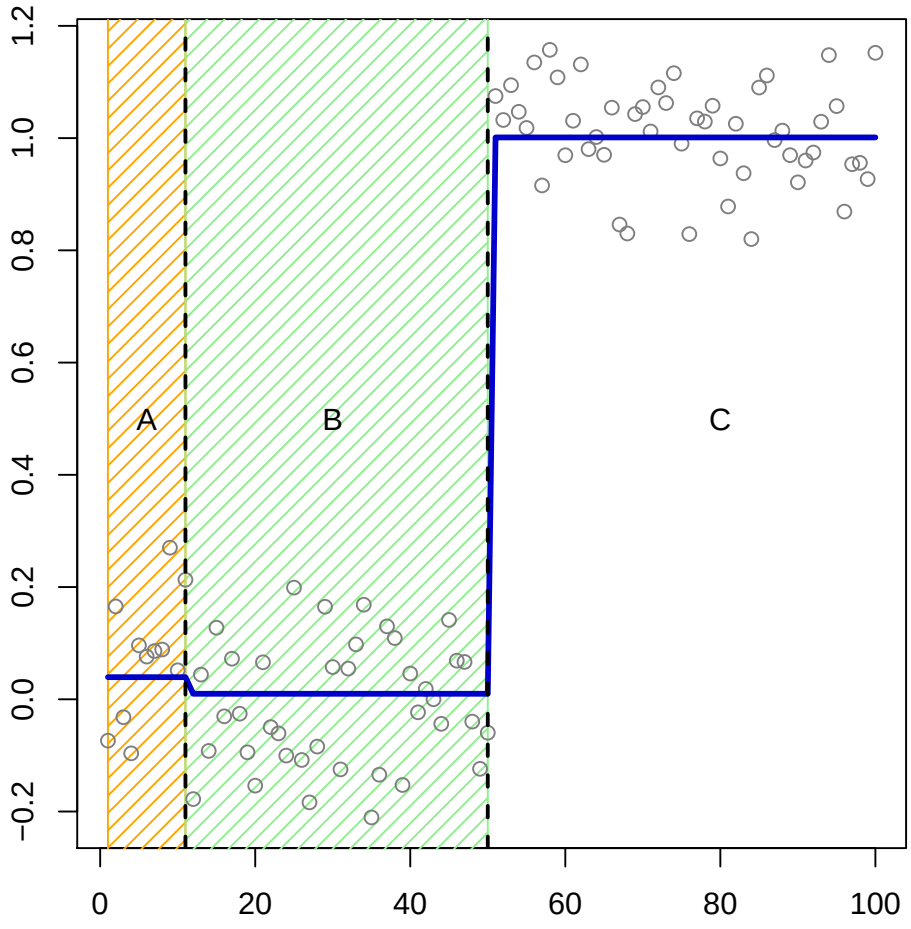


$\lambda = 0.62$



$\lambda = 0.41$

Let's look at the solution at $\lambda = 0.41$ a little more closely

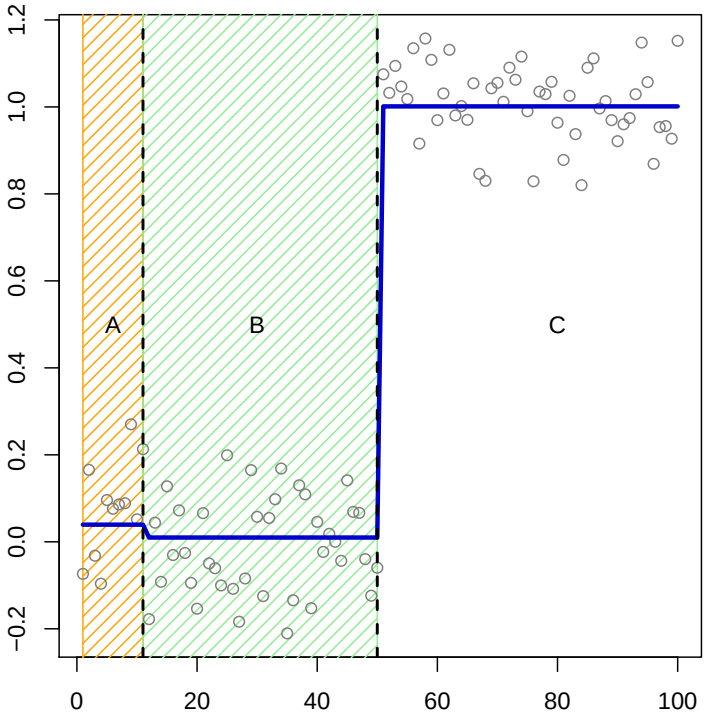


How can we test the significance of detected changepoints? Say, at location 11?

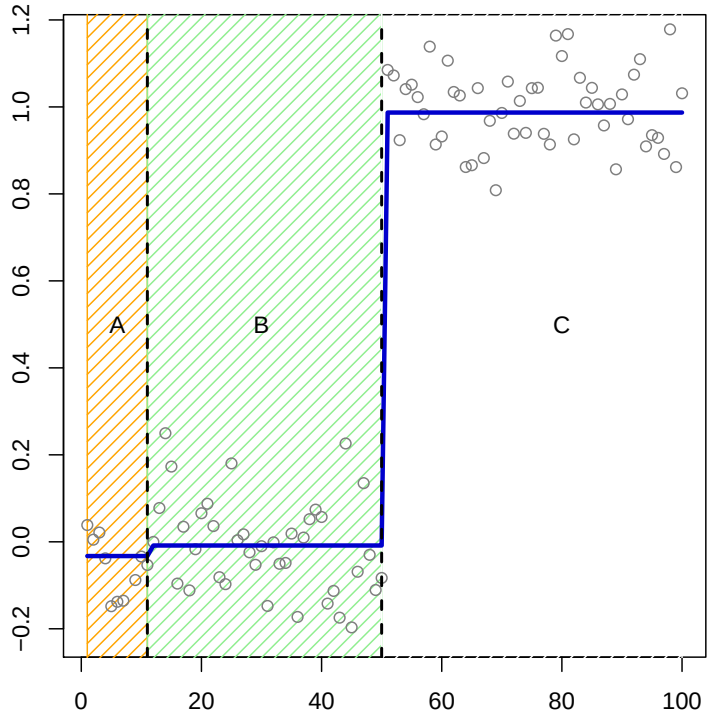
Classically: take the average of data points in region A minus the average in B, compare this to what we expect if the signal was flat

But this is incorrect, because location 11 was selected based on the data, so of course the difference in averages looks high!

What we want to do: compare our observed difference to that in reference (null) data, in which the signal was flat and we happen to select the same location 11 (and 50)



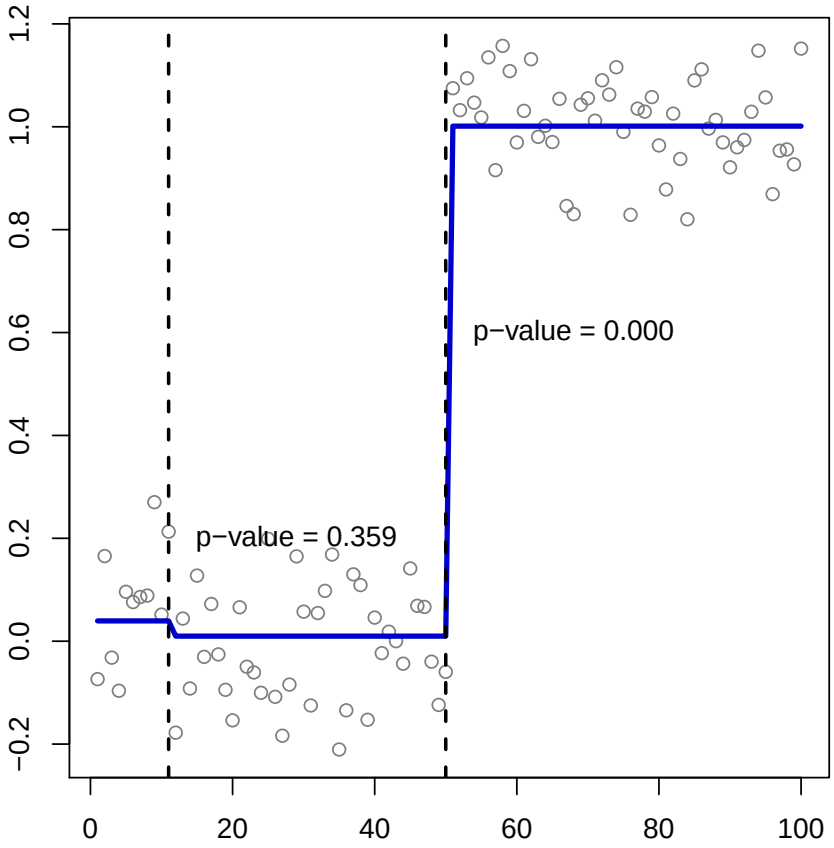
Observed data
Abs. difference ≈ 0.088



Reference data
Abs. difference ≈ 0.072

But it took 1222 simulated data sets to get one reference data set!

The role of optimization: if we understand the fused lasso, i.e., the way it selects changepoints (stems from KKT conditions), then we can come up with a reference distribution without simulation



We can use this to efficiently conduct significance tests[1]

[1] Hyun et al. 2018, "Exact post-selection inference for the generalized lasso path"

Widsom from Friedman (1985)

From Jerry Friedman's discussion of Peter Huber's 1985 projection pursuit paper, in Annals of Statistics:

> A good idea poorly implemented will not work well and will likely be judged not good. It is likely that the idea of projection pursuit would have been delayed even further if working implementations of the exploratory (Friedman and Tukey, 1974) and regression (Friedman and Stuetzle, 1981) procedures had not been produced. As data analytic algorithms become more complex, this problem becomes more acute. The best way to guard against this is to become as literate as possible in algorithms, numerical methods and other aspects of software implementation. I suspect that more than a few important ideas have been discarded because a poor implementation performed badly.

Arguably, less true today due to the advent of disciplined convex programming? But it still rings true in large part ...

Central concept: convexity

Historically, linear programs were the focus in optimization

Initially, it was thought that the important distinction was between linear and nonlinear optimization problems. But some nonlinear problems turned out to be much harder than others ...

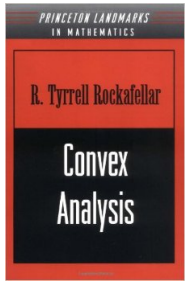
Now it is widely recognized that the right distinction is between convex and nonconvex problems

Your supplementary textbooks for the course:

Boyd and Vandenberghe (2004)

and

Rockafellar (1970)



Wisdom from Rockafellar (1993)

From Terry Rockafellar's 1993 SIAM Review survey paper:

> a convex set every locally optimal solution is global. Also, first-order necessary conditions for optimality turn out to be sufficient. A variety of other properties conducive to computation and interpretation of solutions ride on convexity as well. In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity. Even for problems that aren't themselves of convex type, convexity may enter, for instance, in setting up subproblems as part of an iterative numerical scheme.

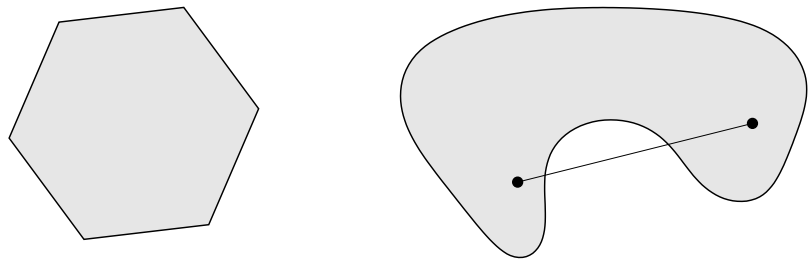
Credit to Nemirovski, Yudin, Nesterov, others for formalizing this

This view was dominant both within the optimization community and in many application domains for many decades (... currently being challenged by successes of neural networks?)

Convex sets and functions

Convex set: $C \subseteq \mathbb{R}^n$ such that

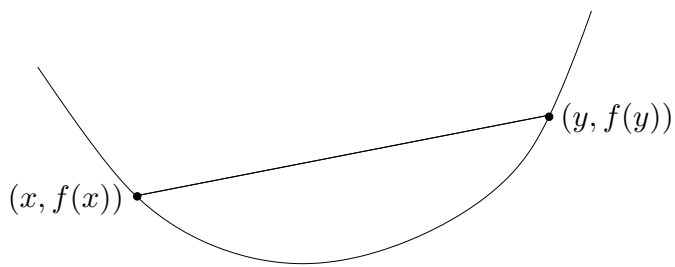
$$x, y \in C \implies tx + (1-t)y \in C \text{ for all } 0 \leq t \leq 1$$



Convex function: $f : \mathbb{R}^n \to \mathbb{R}$ such that $\mathrm{dom}(f) \subseteq \mathbb{R}^n$ convex, and

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \quad \text{for all } 0 \leq t \leq 1$$

and all $x, y \in \mathrm{dom}(f)$



Convex optimization problems

Optimization problem:

$$\begin{array}{ll} \min\limits_{x \in D} & f(x) \\ \text{subject to} & g_i(x) \leq 0,\ i = 1, \ldots, m \\ & h_j(x) = 0,\ j = 1, \ldots, r \end{array}$$

Here $D = \mathrm{dom}(f) \cap \bigcap_{i=1}^m \mathrm{dom}(g_i) \cap \bigcap_{j=1}^p \mathrm{dom}(h_j)$, common domain of all the functions

This is a convex optimization problem provided the functions f and $g_i, i = 1, \ldots, m$ are convex, and $h_j, j = 1, \ldots, p$ are affine:

$$h_j(x) = a_j^T x + b_j, \quad j = 1, \ldots, p$$

Local minima are global minima

For convex optimization problems, local minima are global minima

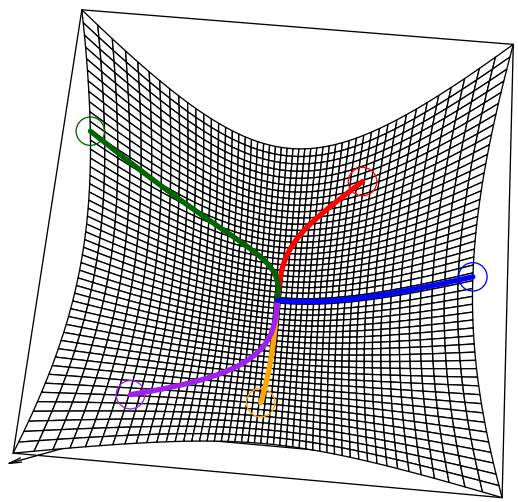
Formally, if x is feasible—$x \in D$, and satisfies all constraints—and minimizes f in a local neighborhood,

$$f(x) \leq f(y) \text{ for all feasible } y,\ \|x - y\|_2 \leq \rho,$$

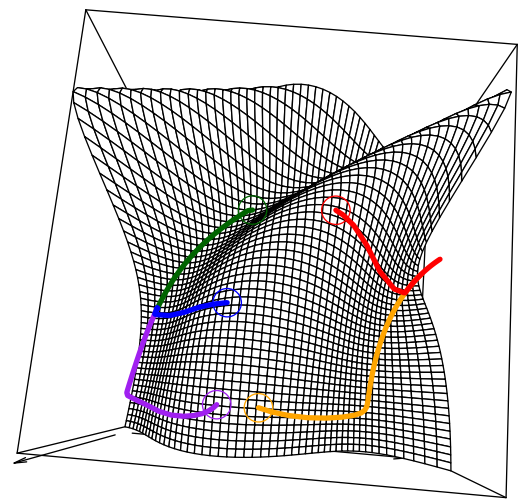
then

$$f(x) \leq f(y) \text{ for all feasible } y$$

This is a very useful fact and will save us a lot of trouble!



Convex



Nonconvex