Machine Learning for Finance

Neal Parikh

Cornell University

Spring 2018

Model Assessment and Model Selection

Outline

Statistical Science
2001, Vol. 16, No. 3, 199–231

Statistical Modeling: The Two Cultures

Leo Breiman

Abstract. There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.

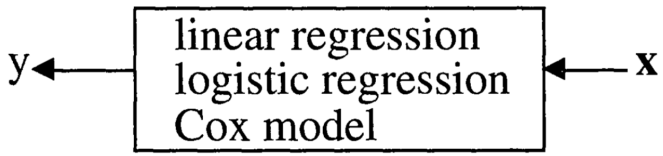
Data analysis

two goals in analyzing data

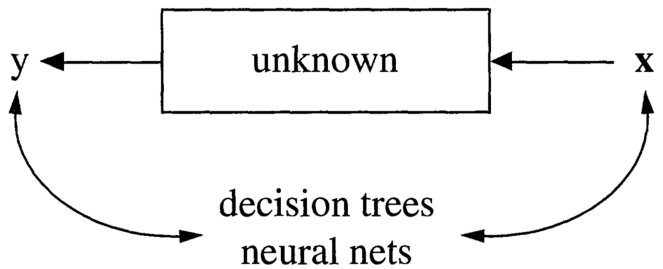
- prediction: predict responses given future input variables
- information: extract information about how nature associates outputs to inputs

Data modeling and algorithmic modeling

- data modeling
  - assume $y = f(x, \text{parameters}, \text{noise})$ and **we know** f
  - estimate parameters, then use for prediction and information
  - validate with goodness-of-fit tests and residual examination



- algorithmic modeling
  - assume nothing about how y is produced from x
  - use learning procedure to build prediction rule
  - validate with prediction accuracy



Data modeling and algorithmic modeling

- Breiman's culture breakdown for statisticians in 2001: 98%/2%
- updated estimate (McAuliffe): 50%/50%
- culture dynamic accelerating towards algorithmic modeling
- quantitative finance culture breakdown $\approx$ statistics c. 2001

Data modeling vs algorithmic modeling

- machine learning (algorithmic modeling) focuses on **accuracy of future predictions**
- classical quant finance methods (data modeling) typically focus on
  - goodness of fit based on in-sample residuals (r^2)
  - significance of, *e.g.*, regression coefficients (p-values)
  - interpretability
- these metrics do **not** directly relate to predictive accuracy

Outline

Statistical decision theory

- given random inputs x and outputs y associated by $p(x, y)$
- call map $f : x \mapsto \hat{y}$ a 'decision rule'
- accuracy of decision rule measured with loss function
  - **0-1 loss**: $L_{01}(y, \hat{y}) = [y \neq \hat{y}]$
  - **squared error loss**: $L_2(y, \hat{y}) = \|y - \hat{y}\|_2^2$

Statistical decision theory

- **risk** is a function of the rule f, and is the expected loss of the rule

$$R(f) = \mathrm{E}[L(y, f(x))]$$

  where expectation is over $(x, y) \sim p(x, y)$

  - 0-1 loss gives 'probability of error' risk $\mathrm{P}(f(x) \neq y)$
  - squared error loss gives 'mean squared error' risk $\mathrm{E}[\|f(x) - y\|_2^2]$

- optimal decision rule $f^\star$ obtained by choosing decision rule to minimize desired risk function

Statistical learning theory

- in statistical *learning* theory, the setup is that we do **not** know $p(x, y)$; only have a collection of training examples $\mathcal{D} = \{(x_i, y_i)\}$ from **unknown** $p(x, y)$
- now cannot compute risk or obtain $f^\star$
- two options
  - (A) use $\mathcal{D}$ to estimate $p(x, y)$, then derive optimal rule from there (hard)
  - (B) use $\mathcal{D}$ to directly estimate a decision rule (do this)

Empirical risk minimization

- we want to find $f^\star$ minimizing risk function R
- idea: approximate R (an expectation) with empirical average

$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i)$$

  called **empirical risk**, *i.e.*, average loss over training set

- **empirical risk minimization** (ERM): given collection of possible decision rules $\mathcal{F}$, select

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{R}(f)$$

Error decomposition

- given
  - $\mathcal{F}$: class of decision rules being used for modeling
  - $\hat{f}$: minimizer of empirical risk $\hat{R}$ in $\mathcal{F}$
  - $f^\star_{\mathcal{F}}$: minimizer of true risk R in $\mathcal{F}$
  - $f^\star$: minimizer of true risk R, not constrained to $\mathcal{F}$
- can decompose **excess risk** of estimate $\hat{f}$ as

$$\underbrace{R(\hat{f}) - R(f^\star)}_{\text{excess risk}} = \underbrace{(R(\hat{f}) - R(f^\star_{\mathcal{F}}))}_{\text{estimation error}} + \underbrace{(R(f^\star_{\mathcal{F}}) - R(f^\star))}_{\text{approximation error}}$$

  - approximation error ('bias'): cost of using model class $\mathcal{F}$
  - estimation error ('variance'): cost of using empirical risk on $\mathcal{D}$
  - (note: also 'optimization error', since may be hard to find $\hat{f}$)

Error decomposition

- previous decomposition useful for qualitative understanding of sources of error in choosing particular model, and possible remedies
- increasing $\mathcal{F}$ (*i.e.*, use more complex model) reduces bias
- restricting $\mathcal{F}$ increases bias, but helps reduce variance
- increasing amount of data (*e.g.*, to infinite dataset) reduces variance
- could consider models that increase bias if they sufficiently reduce variance (will come back to this)

Bias-variance decomposition

- let's look at this in some detail for squared error loss to make this more concrete and motivate the bias/variance terminology
- making this concrete is tricky, because it requires being very careful about which quantities are known/unknown, fixed/random
- people go through this setup differently and use slightly different terminology, but the end result is the same

Bias-variance decomposition

- suppose we have a training set $\mathcal{D} = \{(x_i, Y_i)\}$ generated from

$$Y \sim \mathrm{N}(w_0^T x, \sigma^2)$$

  - w_0 is 'true parameter', fixed but unknown
  - σ^2 is fixed but unknown
  - x_i are fixed and known
  - Y_i are random (according to model above)

- because Y_i are random, $\mathcal{D}$ is random

- could also set up as $Y = f(x) + \epsilon$ where $\mathrm{E}[\epsilon] = 0$, $\mathbf{var}[\epsilon] = \sigma^2$, or also consider x random, or ...; main story is the same

Bias-variance decomposition

- we now compute the MLE $\hat{w}$ (equivalently, $\hat{f}$, ...)
- $\hat{w}$ is a random variable because $\mathcal{D}$ is random
- question: how close is $\hat{w}$ to w_0?
- different ways one might think about this
- idea: want to estimate out of sample error, so measure error between prediction and random new point drawn according to data distribution
- *i.e.*, there are *two* sources of randomness

Bias-variance decomposition

- question: how close is $\hat{w}$ to w_0?
- evaluate by MSE between *expected* prediction and response

$$\text{MSE} = \mathrm{E}_{\mathcal{D}}[(\hat{w}^T x^{\text{new}} - w_0^T x^{\text{new}})^2]$$

  for possible new input x^{new}

- here, everything is fixed except $\hat{w}$
- MSE decomposes (writing x instead of x^{new})

$$\begin{aligned}
\text{MSE} &= \mathrm{E}[(\hat{w}^T x)^2] - 2\mathrm{E}[\hat{w}^T x](w_0^T x) + (w_0^T x)^2 \\
&= \mathrm{E}[(\hat{w}^T x)^2] - 2\mathrm{E}[\hat{w}^T x](w_0^T x) + (w_0^T x)^2 + \mathrm{E}[\hat{w}^T x]^2 - \mathrm{E}[\hat{w}^T x]^2 \\
&= \left(\mathrm{E}[(\hat{w}^T x)^2] - \mathrm{E}[\hat{w}^T x]^2\right) + \left(\mathrm{E}[\hat{w}^T x] - w_0^T x\right)^2
\end{aligned}$$

Bias-variance decomposition

- this is exactly the bias-variance decomposition

$$\text{MSE} = \underbrace{\left(\mathrm{E}[(\hat{w}^T x)^2] - \mathrm{E}[\hat{w}^T x]^2\right)}_{\text{variance}} + \underbrace{\left(\mathrm{E}[\hat{w}^T x] - w_0^T x\right)^2}_{\text{bias (squared)}}$$

- but here, we measured MSE between average prediction and *average* output $w_0^T x^{\text{new}}$

- really, x^{new} would have an associated y^{new}, which is centered at $w_0^T x^{\text{new}}$ but has some variance σ^2

Bias-variance decomposition

- want to see if $\hat{w}^T x^{\text{new}} \approx y^{\text{new}}$
- sometimes call this the expected prediction error or expected test error at a given point

$$\text{EPE} = \text{E}_{\mathcal{D}} \left[\text{E}_Y \left[(\hat{w}^T x - Y)^2 \right] \right]$$

  similar to before, but also averaging over randomness in Y

- via derivation similar to the previous one, get decomposition

$$\begin{aligned} \text{EPE} &= \mathbf{var}(Y) + \text{MSE}(\hat{w}^T x) \\ &= \sigma^2 + \text{bias}(\hat{w}^T x)^2 + \mathbf{var}(\hat{w}^T x) \end{aligned}$$

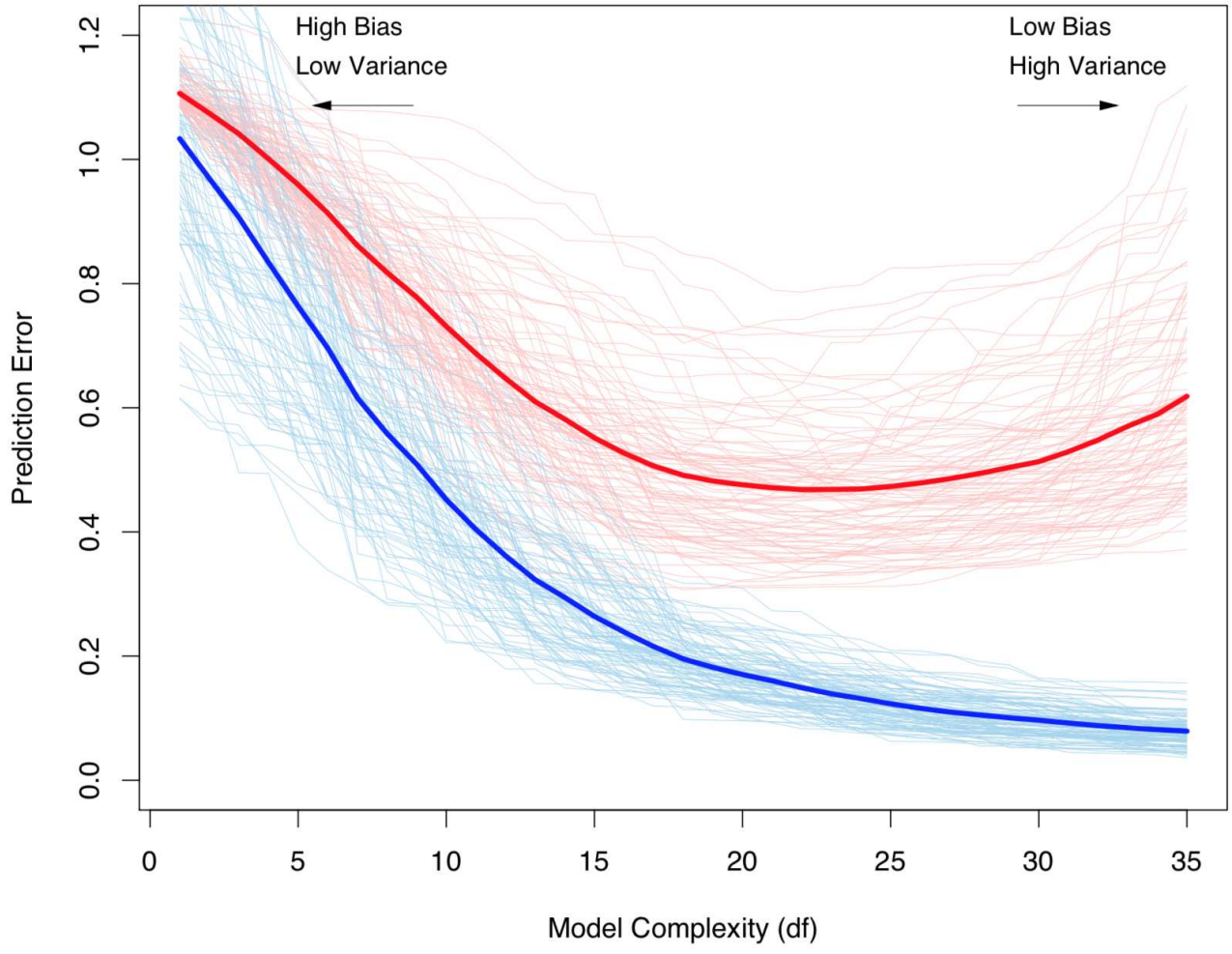
  where σ^2 is called intrinsic noise or irreducible error

- the average test MSE we would obtain if we repeatedly estimated $\hat{w}$ using different $\mathcal{D}$ and tested each at x^{new}

Bias-variance decomposition

- we want to choose method that gives lowest test MSE
- we can't compute that, but we do know a few things
  1. we can compute the MSE on the training set (training MSE or training error or empirical risk)
  2. test error has two key components (bias and variance), and we ideally want to make both small
  3. easy to be at either extreme (zero bias or variance)

Training error and test error



Overfitting and underfitting

- high bias causes algorithm to miss relevant relationships between inputs and outputs (**underfitting**)
- high variance causes algorithm to model random noise in the training set (**overfitting**)

- model too complex $\Rightarrow$ small bias, large variance $\Rightarrow$ overfitting
- model too simple $\Rightarrow$ large bias, small variance $\Rightarrow$ underfitting

- goal is to tune model complexity to problem at hand

Overfitting and underfitting

- training error is not a good estimate of test error
- instead, estimate test error using actual test data set aside

Model assessment and model selection

- **model selection**: estimating the performance of different models in order to choose the (approximate) best one
- **model assessment**: having chosen a model, estimating its performance (generalization error) on new, unseen data

Model assessment and model selection

- **hold-out cross validation**: if there's enough data, randomly divide $\mathcal{D}$ into three parts

  (A) **training set**: used to fit models

  (B) **validation set**: estimate prediction error for model selection

  (C) **test set**: assess generalization error for chosen model

- test set should be used once; otherwise, if model with lowest test error is chosen, that test error will underestimate the true error

- basic idea: you can't overfit data you didn't fit

- (**warning**: sometimes names of validation/test sets are flipped)

k-fold cross validation

1. randomly split $\mathcal{D}$ into k disjoint subsets $\mathcal{D}_i$
2. for each model, for each fold i
    1. train model on $\mathcal{D} - \mathcal{D}_i$
    2. compute test error on $\mathcal{D}_i$
3. pick model with lowest average test error (estimated generalization error) across the folds

Difficulties with cross validation

- choosing the folds
  - respect grouping
  - respect time (avoid *look-ahead bias*)
- small datasets
- **major issue**: assuming future data is similar to training data
  - non-stationarity over time (**different** from overfitting)
  - applying to different population than in training set (can lead to major ethical issues in addition to poor performance)

Training set bias

Facial detection algorithms made in the U.S. are frequently trained and evaluated using data sets that contain far more photos of white faces, and they're generally tested and quality controlled by teams of engineers who aren't likely to have dark skin. As a result, some of these algorithms are better at identifying lighter skinned people, which can lead to problems ranging from passport systems that incorrectly read Asians as having their eyes closed, to HP webcams and Microsoft Kinect systems that have a harder time recognizing black faces, to Google Photos and Flickr auto-tagging African-Americans as apes.

— C. Couch, "Ghosts in the Machine", October 2017

Outline

Evaluating classifiers

1. pick decision threshold (*e.g.*, 0) based on model output (*e.g.*, $w^T x$)
2. compute relevant evaluation metric for classifier

Prediction errors

- only four possibilities, so given names

- **true positive**: $y = +1$ and $\hat{y} = +1$
- **true negative**: $y = -1$ and $\hat{y} = -1$
- **false positive**: $y = -1$ and $\hat{y} = +1$
- **false negative**: $y = +1$ and $\hat{y} = -1$

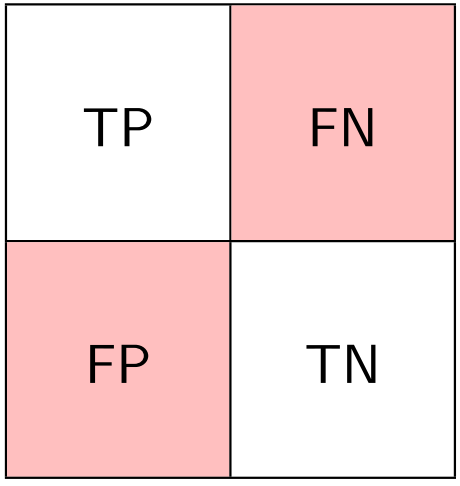
- false positive also called **type I error**
- false negative also called **type II error**

- many other metrics defined as functions of these

Confusion matrix

TP	FN
FP	TN

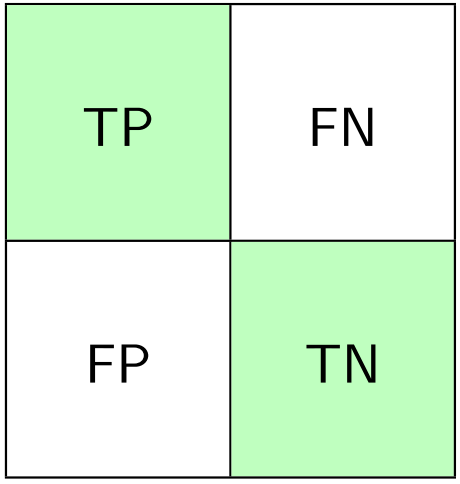
Error rate



$$\text{error rate} = \frac{\text{FP} + \text{FN}}{N}$$

what fraction of all predictions were wrong?

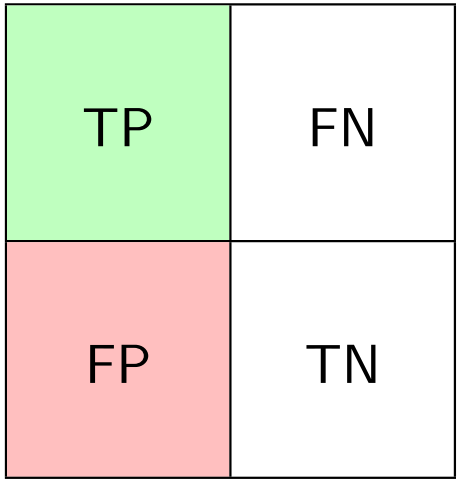
Accuracy



$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{N}$$

what fraction of all predictions were right?

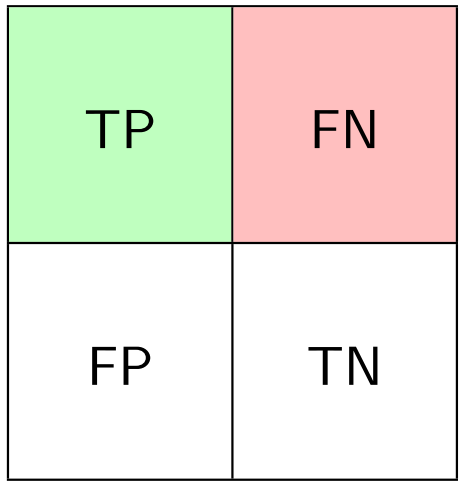
Precision



$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

of positive predictions, how many were right?

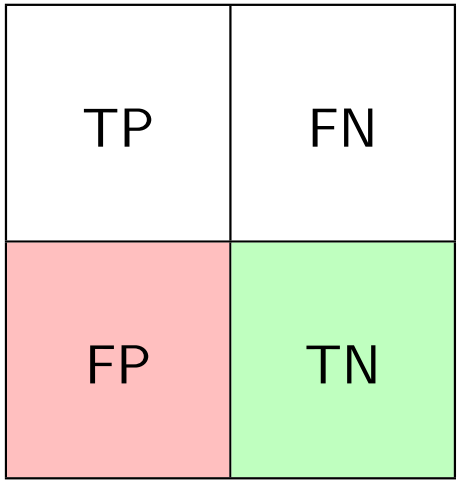
Recall / Sensitivity / True positive rate



$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

proportion of correctly identified positives

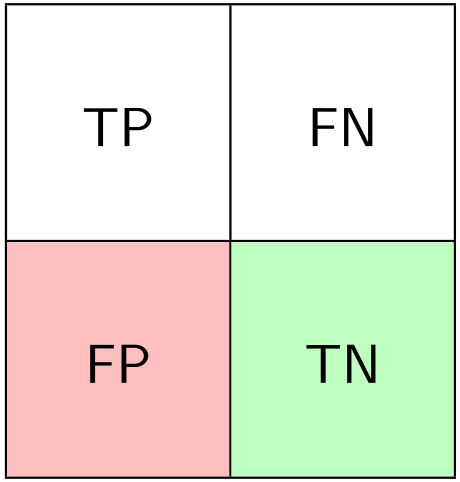
False positive rate



$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

proportion of wrongly identified positives (false alarms)

Specificity / True negative rate



$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

proportion of correctly identified negatives

F_1 score

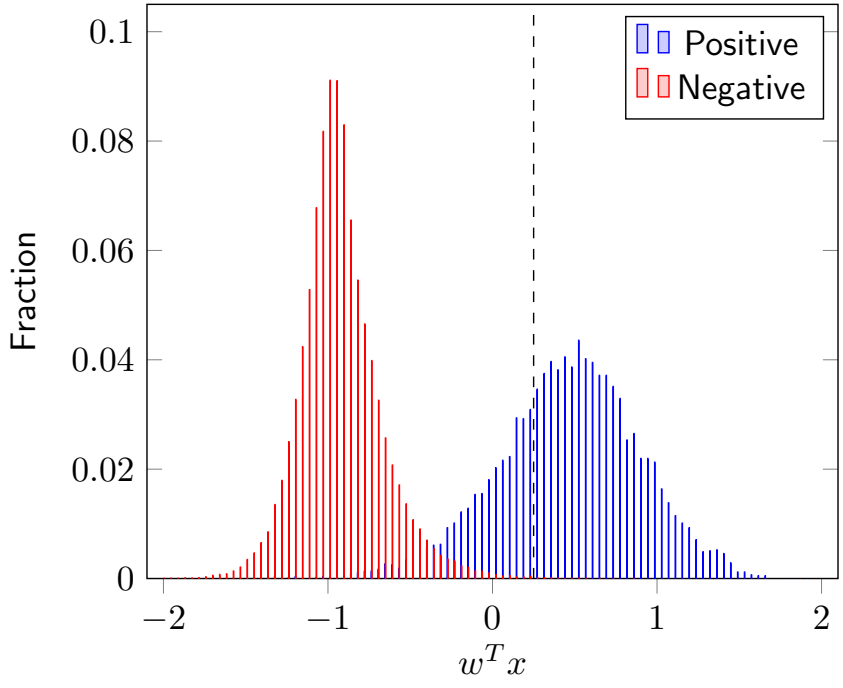
- precision and recall have to be traded off
- easy to make one small at the expense of the other
- can use combined score that is harmonic mean of precision and recall

$$F_1 = \frac{2}{1/P + 1/R}$$

Receiver operating characteristic

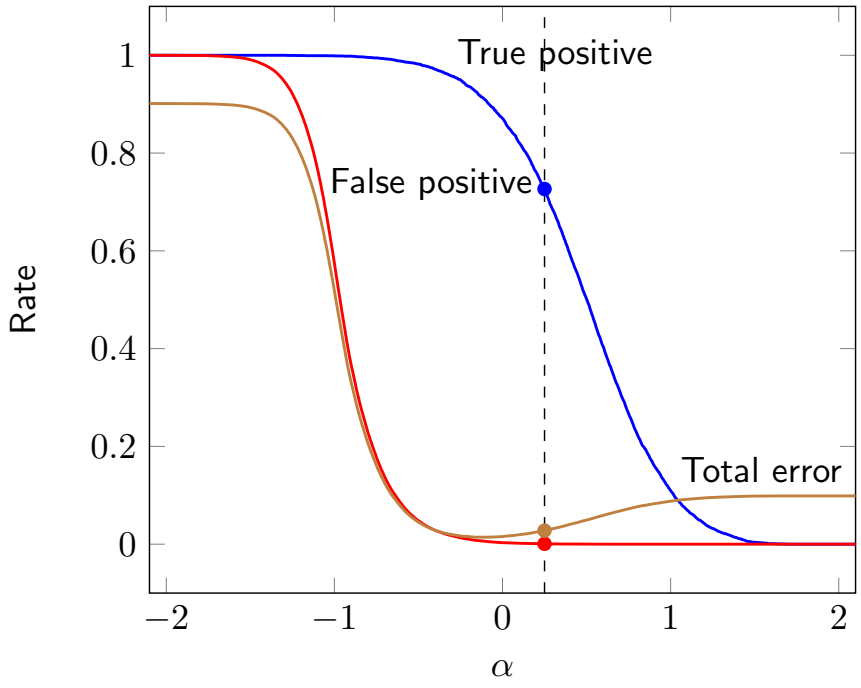
- let α be the decision threshold for the classifier
- larger α decreases TPR (bad) and FPR (good)
- smaller α increases TPR (good) and FPR (bad)
- choose α based on how much we care about tradeoff
- by varying α, get range of TPR/FPR and error rate
- standard plot for this called the 'ROC curve' (now strange terminology tracing back to radar systems in WWII)
- another popular metric is AUC, or area under the ROC curve

Receiver operating characteristic



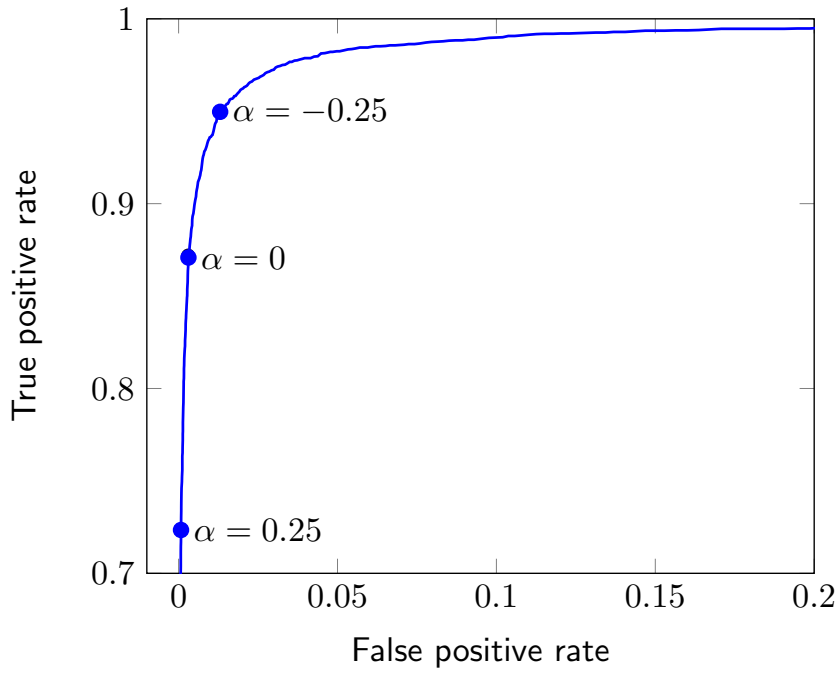
True positive, false positive, and total error rate versus decision threshold α. The vertical dashed line is shown for decision threshold $\alpha = 0.25$.

Receiver operating characteristic



True positive, false positive, and total error rate versus decision threshold α. The vertical dashed line is shown for decision threshold $\alpha = 0.25$.

Receiver operating characteristic



ROC curve.

Class imbalance

- example of a complicating issue is **class imbalance**, which occurs when one class is relatively rare
- classifier may perform well by metrics above but still be totally useless (*e.g.*, compare to blindly predicting majority class)

Class imbalance

- collecting more data
- adjusting performance metric
- resample dataset (oversample minority class, undersample majority)
- adjust loss function (*e.g.*, class-weighted SVM)
- formulate problem differently (*e.g.*, outlier detection)

Outline

Debugging learning algorithms

- consider a binary classification problem: spam filtering
- choose set of 100 words as features
- consider using SVM classifier
- suppose test error is 20%, which is too high
- now what?

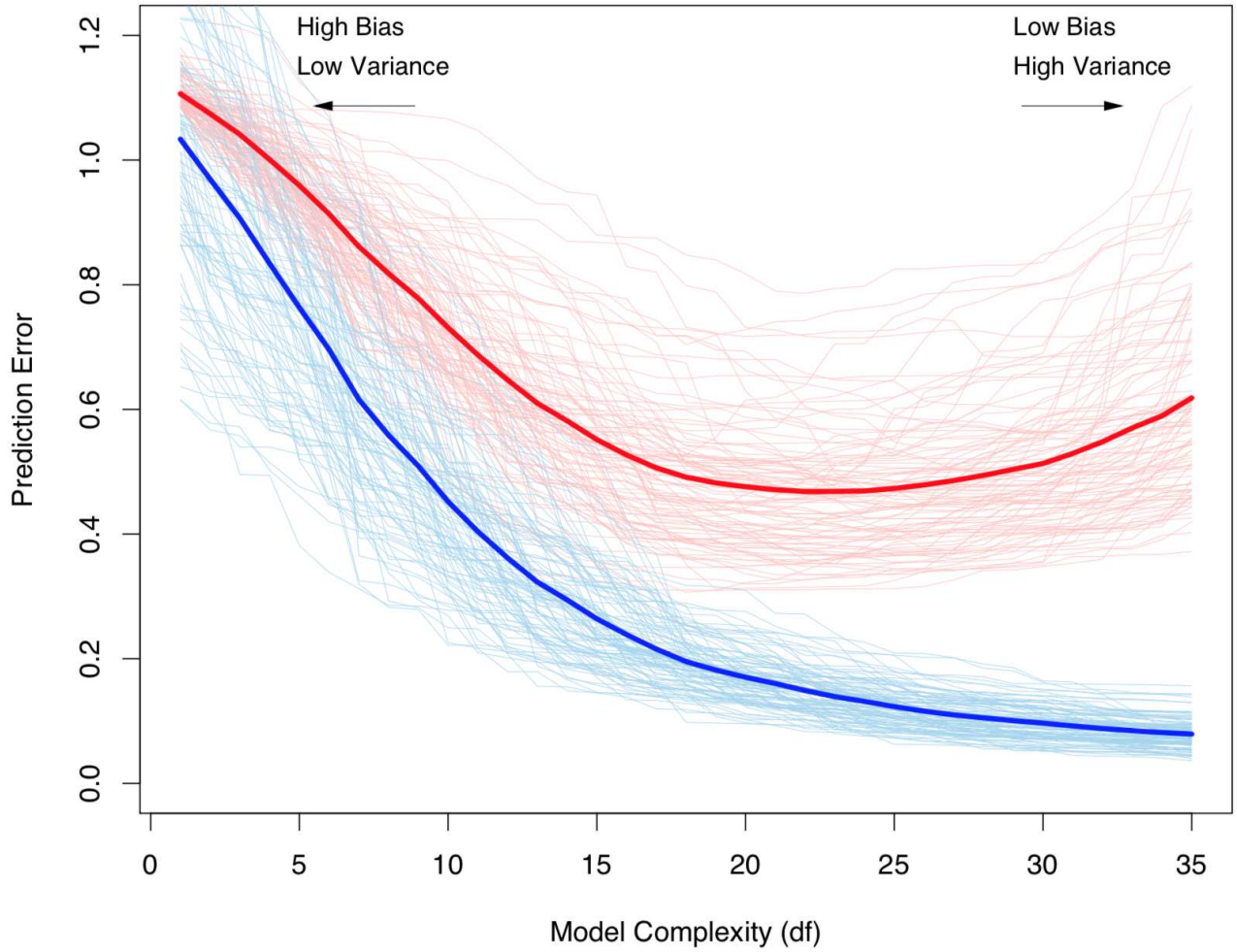
Debugging learning algorithms

- common approach is to try wide range of improvements
  - try getting more training data
  - try fewer features
  - try more features
  - try different features
  - change optimization algorithm
  - change model hyperparameter or family (*e.g.*, logistic regression)
- could work, but very time consuming, expensive; relies on luck
- better idea: try to come up with a diagnostic and fixed based on that

Bias-variance diagnostic

- suspect that the problem is either
  1. overfitting (high variance)
  2. too few features for model to perform well (high bias)
- suggests diagnostics:
  - high variance: test error will be much higher than training error
  - high bias: training error will also be high

Training error and test error



Bias-variance diagnostic

- pares down what to try
  - try getting more training data (fixes high variance)
  - try fewer features (fixes high variance)
  - try more features (fixes high bias)
  - try different features (can fix high bias)
  - change optimization algorithm
  - change model hyperparameter or family
- coming up with appropriate diagnostics can require creativity

Regularization

Regularization

- want to explore tradeoff between bias and variance, with the goal of better out of sample performance
- regularization essentially places constraints on model parameters
- increases bias due to constraints
- decreases variance and encourages simpler models
- simpler models can be more interpretable in addition to potentially performing better out of sample

Best subset selection

- fit a separate least squares regression for each of the 2^K possible combination of K features (can extend to other models)

(1) let M_0 denote model with no features (predicts sample mean for each observation)

(2) for $k \in [K]$

  (a) fit all models with exactly k predictors

  (b) pick best (smallest training MSE) among these, and call it M_k

(3) select best model from $M_0, \ldots, M_K$ with cross-validation

Forward stepwise selection

- computationally efficient alternative to best subset selection

(1) let M_0 denote model with no features

(2) for $k = 0, \ldots, K-1$

  (a) fit all $K - k$ models that add exactly one feature

  (b) pick best (smallest training MSE) among these, and call it M_{k+1}

(3) select best model from $M_0, \ldots, M_K$ with cross-validation

Shrinkage

- subset selection: fit a model containing a subset of the features
- alternatively, can fit a model with all the features, but constrain (or *regularize*) the coefficient estimates
- resulting estimators sometimes called **shrinkage estimators** since they 'shrink' coefficients towards zero
- two best-known techniques: ridge regression and the lasso

Ridge regression

- idea: constrain w, *i.e.*

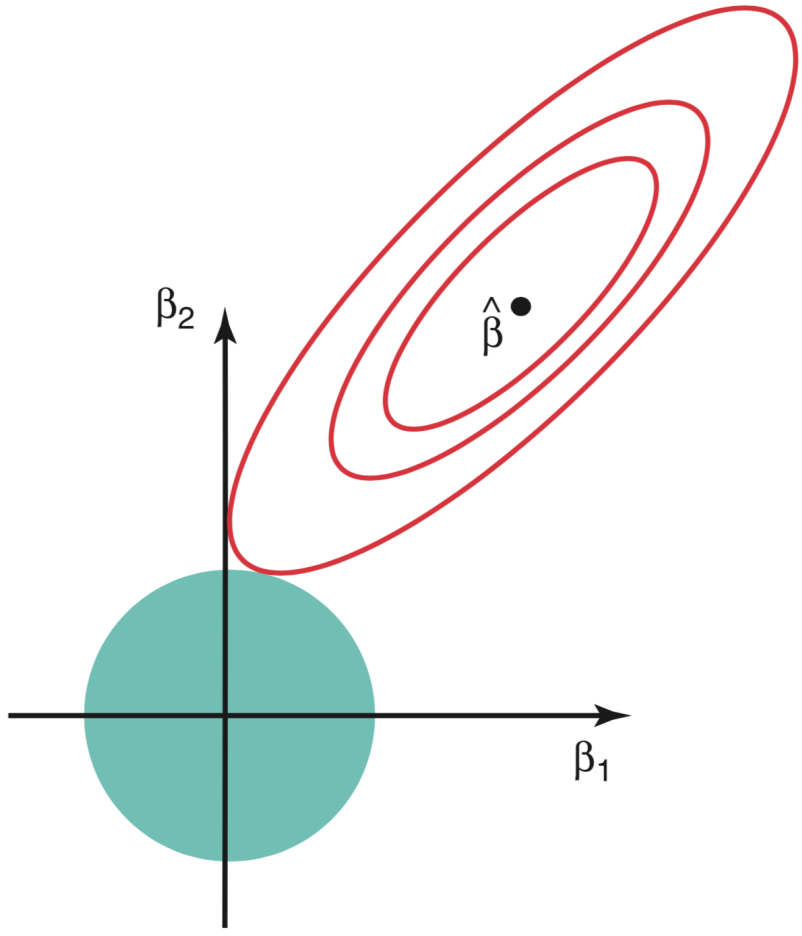
$$\begin{array}{ll} \text{minimize} & (1/2)\|Xw - y\|_2^2 \\ \text{subject to} & \|w\|_2^2 \leq r \end{array}$$

  constrains weights to lie in sphere of radius r

- solution $w^\star$ to this problem known as 'ridge estimator'

- in practice, don't penalize intercept term; also assume that X is standardized (mean zero, unit variance) and y is centered

Ridge regression



Ridge regression

- has equivalent unconstrained form

$$\text{minimize} \quad (1/2)\|Xw - y\|_2^2 + \lambda\|w\|_2^2$$

  with tradeoff parameter $\lambda > 0$

- this is the usual form used

- called ridge regression, Tikhonov regularization, weight decay, ...

Regularization parameter

- λ is chosen via cross-validation
- called a complexity, regularization, or tradeoff parameter
- higher values correspond to simpler models
- as $\lambda \to 0$, get regular least squares fit
- as $\lambda \to \infty$, get close to $\hat{w} = 0$

Regularization parameter

- problem can be viewed as scalarization of multicriterion problem

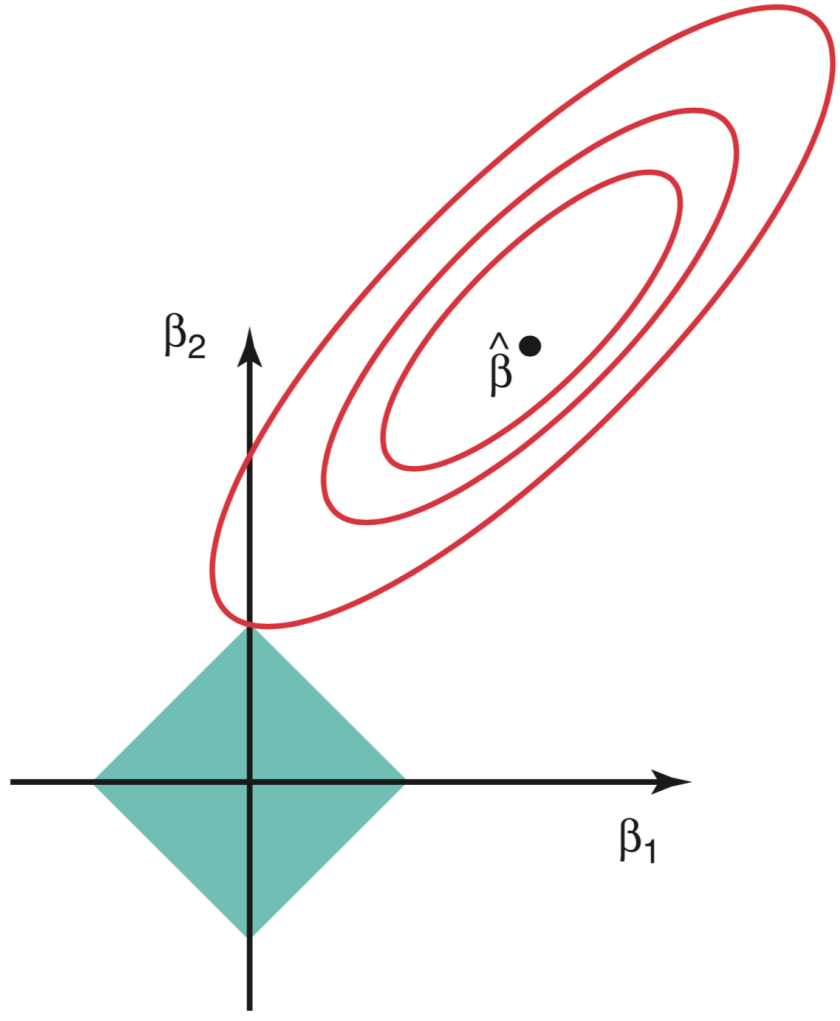
$$\text{minimize (w.r.t. } \mathbf{R}_+^2) \quad (\|Xw - y\|_2^2, \|w\|_2^2)$$

- parameter λ governs the tradeoff
- get a solution $\hat{w}$ for each value of λ, so the 'path' of these $\hat{w}$ as a function of λ is called the 'regularization path'

Ridge regression and subset selection

- subset selection gives interpretable, parsimonious models, but is computationally very inefficient
- ridge regression is efficient and can effectively manage bias-variance tradeoff, but does not simplify models, only carries out 'weight decay' or 'shrinkage'
- consider marrying aspects of these two approaches
- specifically, consider constraining weights in such a way that many are likely to be zero, *i.e.*, $\hat{w}$ will be **sparse**

ℓ_1 regularization



Lasso

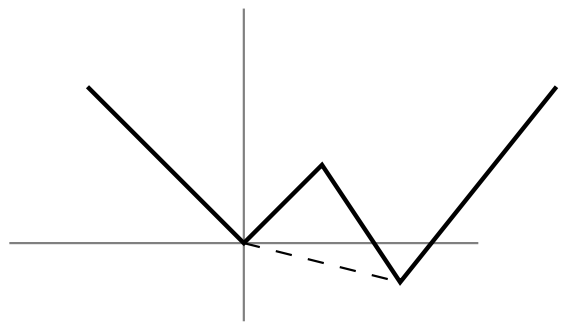
- problem in form

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \lambda\|x\|_1$$

- sparse and convex
- simultaneously carries out feature selection and learning
- unlike ridge regression, no closed form solution, and moreover, the problem is **nonsmooth**
- can be used with other loss functions (*e.g.*, logistic)

Convex envelope interpretation

- convex envelope of (nonconvex) f is the largest convex underestimator g
- *i.e.*, the best convex lower bound to a function

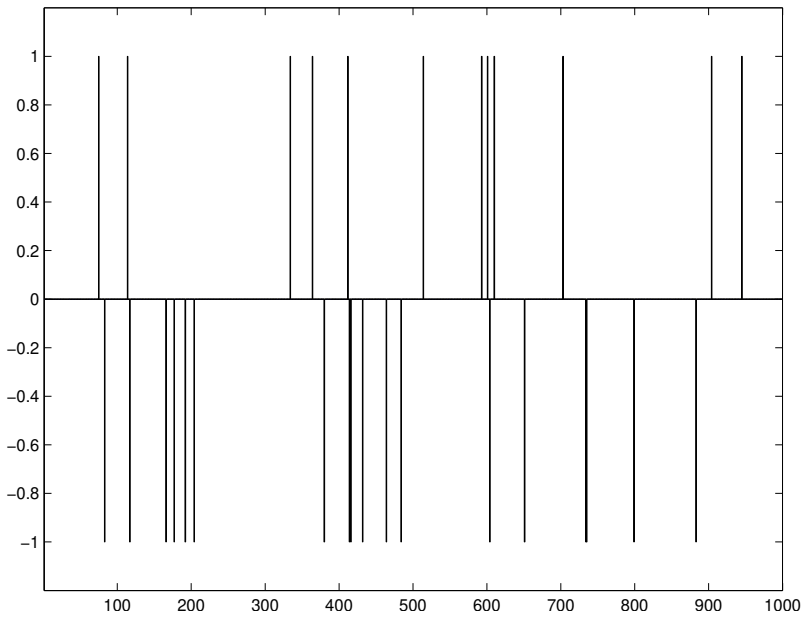


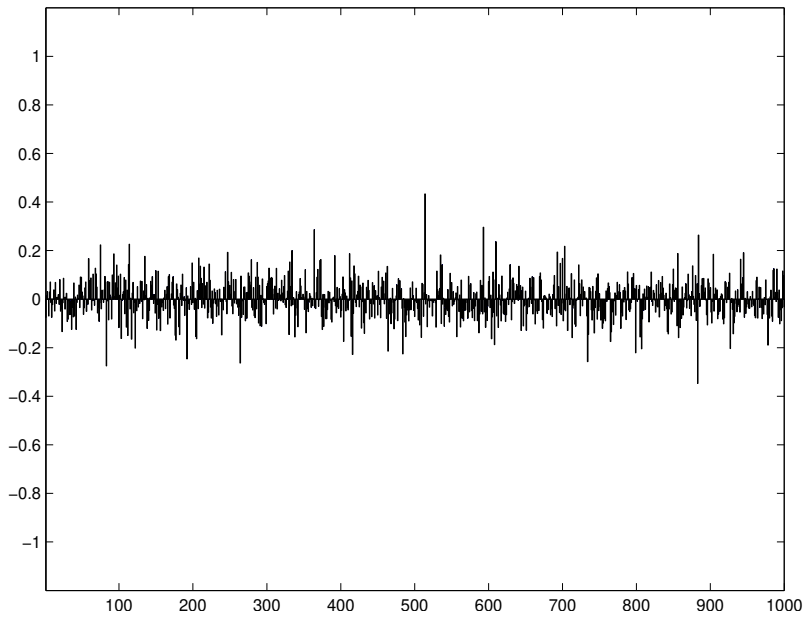
- ℓ_1 is the envelope of $\mathbf{card}$ (also called ℓ_0) (on unit ℓ_∞ ball)
- various characterizations: *e.g.*, f^{**} or convex hull of epigraph
- (similar in flavor to use of convex surrogates for 0-1 loss)

Penalty function interpretation

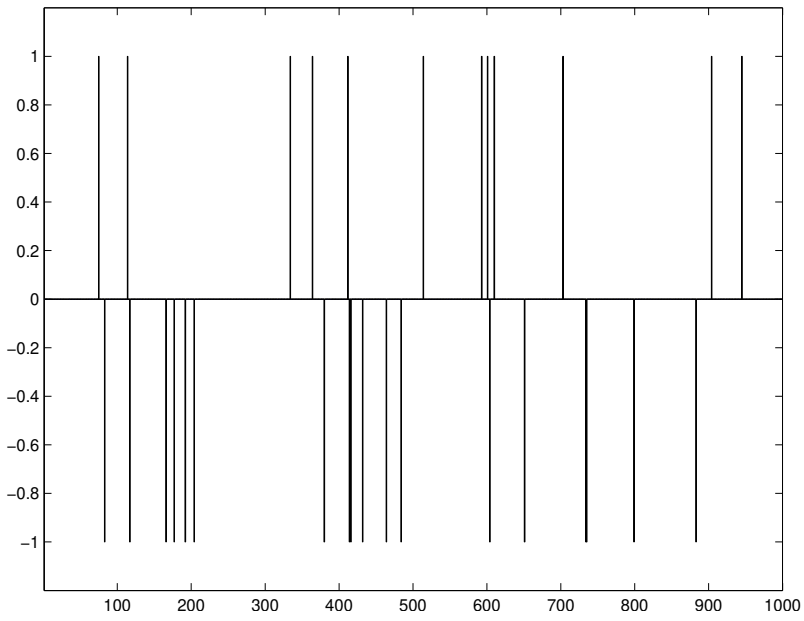
- compared to ridge penalty $\|x\|_2^2$, using ℓ_1 does two things:
  1. higher emphasis on small values to go to exactly zero
  2. lower emphasis on avoiding very large values
- thus useful for obtaining **sparse** or **robust** solutions to problems

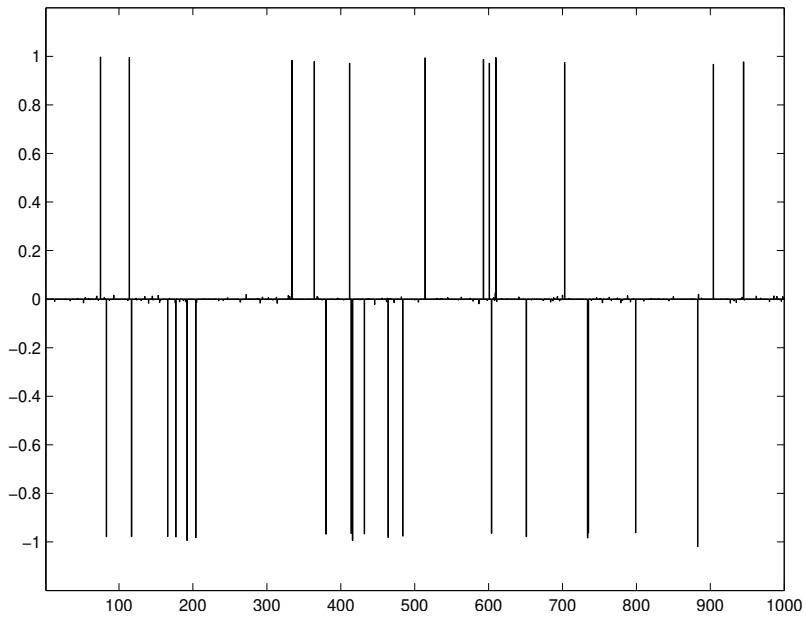
Lasso


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
100
200
300
400
500
600
700
800
900
1000


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
100
200
300
400
500
600
700
800
900
1000

Lasso


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
100
200
300
400
500
600
700
800
900
1000


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
100
200
300
400
500
600
700
800
900
1000

Extensions and variations

- there are *many* variations on the lasso or other uses of ℓ_1 regularization, or places where it can be used to interpret what's going on

- when there is an ℓ_1 (or similar nonsmooth component), should expect some kind of sparsity

Sparsity and support vector machines

- recall soft-margin SVM problem

$$\begin{array}{ll} \text{minimize} & (1/2)\|w\|_2^2 + \lambda \mathbf{1}^T t \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - t_i, \quad i = 1, \ldots, N \\ & t \succeq 0 \end{array}$$

- when $t \succeq 0$, have that $\mathbf{1}^T t = \|t\|_1$, so can be rewritten

$$\begin{array}{ll} \text{minimize} & (1/2)\|w\|_2^2 + \lambda \|t\|_1 \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - t_i, \quad i = 1, \ldots, N \\ & t \succeq 0 \end{array}$$

- suggests interpretation involving sparsity

- unconstrained form (quadratically regularized hinge loss minimization) also has this flavor

Signal reconstruction

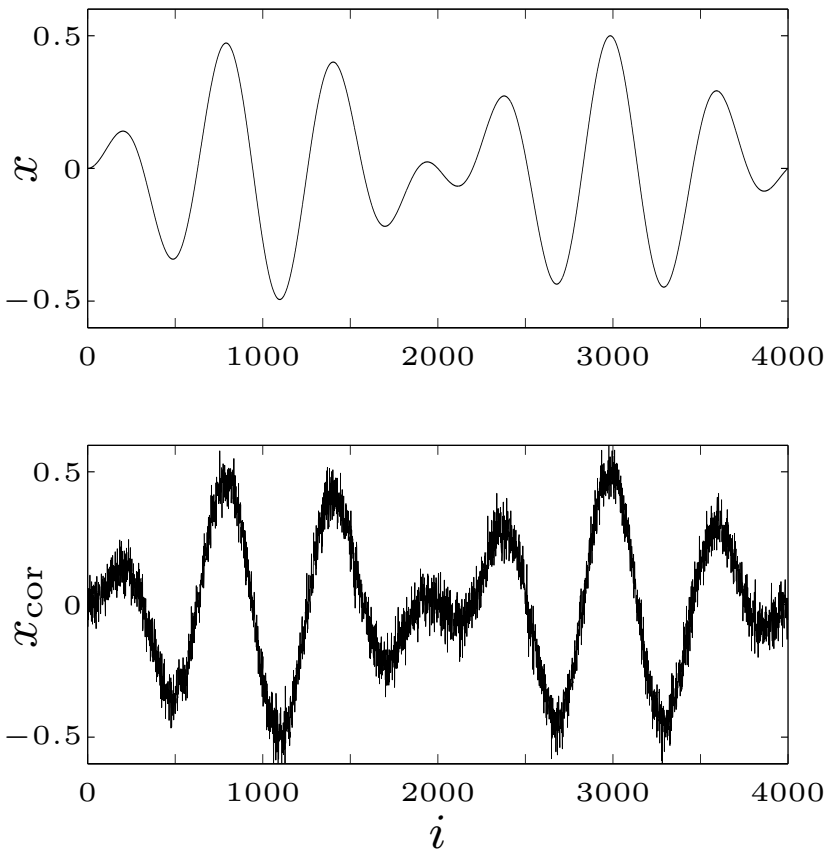
$$\text{minimize (w.r.t. } \mathbf{R}^2_+) \quad (\|\hat{x} - x_{\text{cor}}\|_2, \phi(\hat{x}))$$

- $x \in \mathbf{R}^n$ is unknown signal
- $x_{\text{cor}} = x + v$ is (known) corrupted version of x, with additive noise v
- variable $\hat{x}$ (reconstructed signal) is estimate of x
- $\phi : \mathbf{R}^n \to \mathbf{R}$ is regularization function or smoothing objective

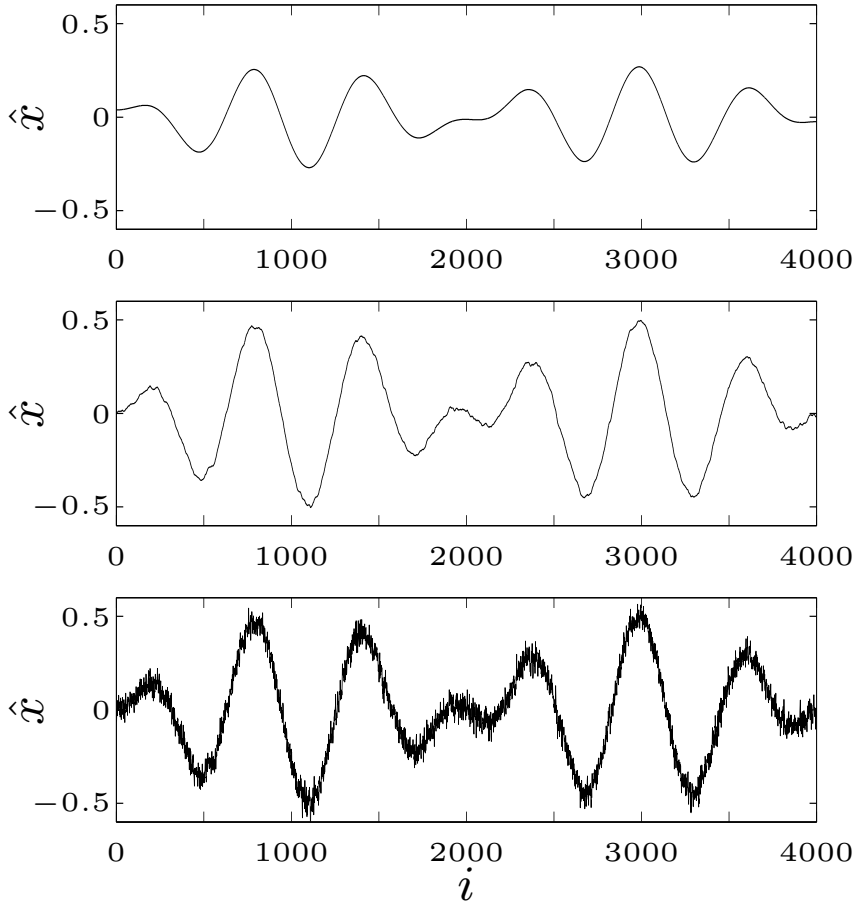
examples: quadratic smoothing, total variation smoothing:

$$\phi_{\text{quad}}(\hat{x}) = \sum_{i=1}^{n-1} (\hat{x}_{i+1} - \hat{x}_i)^2, \qquad \phi_{\text{tv}}(\hat{x}) = \sum_{i=1}^{n-1} |\hat{x}_{i+1} - \hat{x}_i|$$

quadratic smoothing example

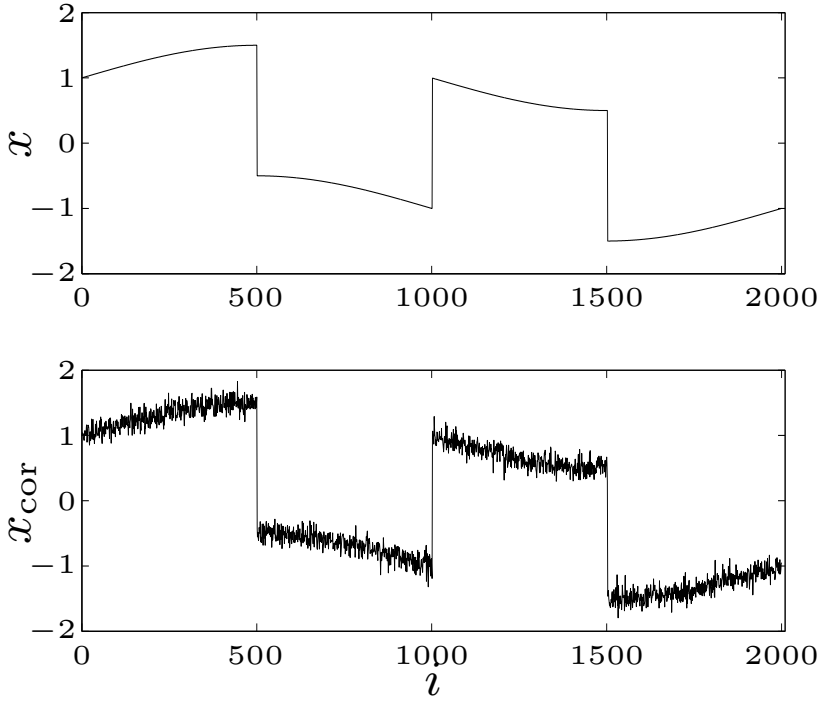


original signal x and noisy signal $x_{\rm cor}$

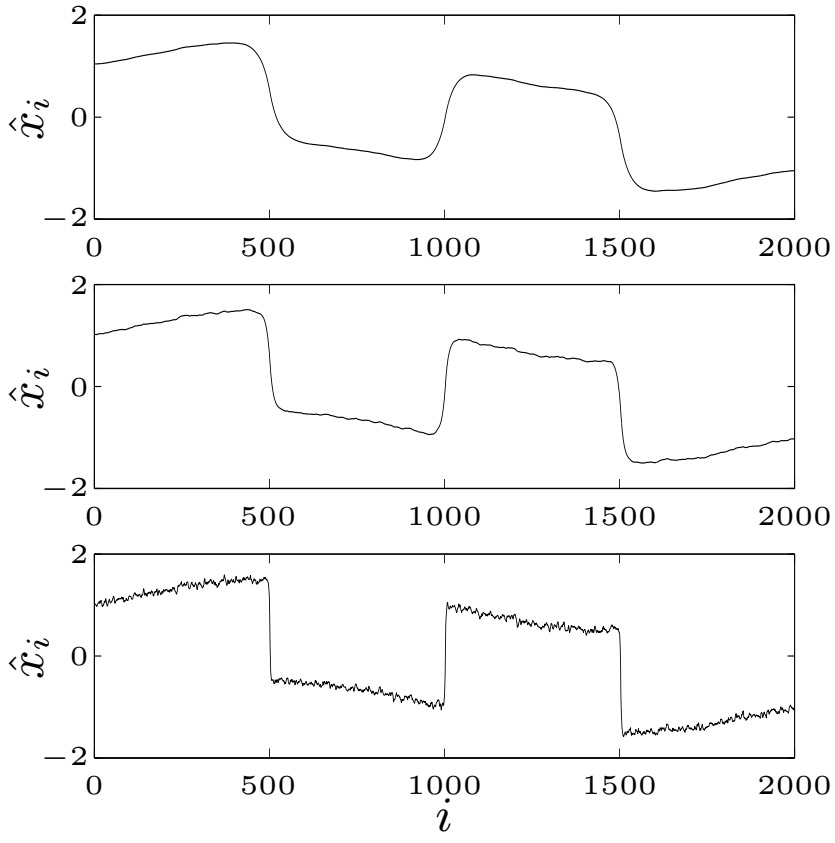


three solutions on trade-off curve $\|\hat{x} - x_{\rm cor}\|_2$ versus $\phi_{\rm quad}(\hat{x})$

total variation reconstruction example

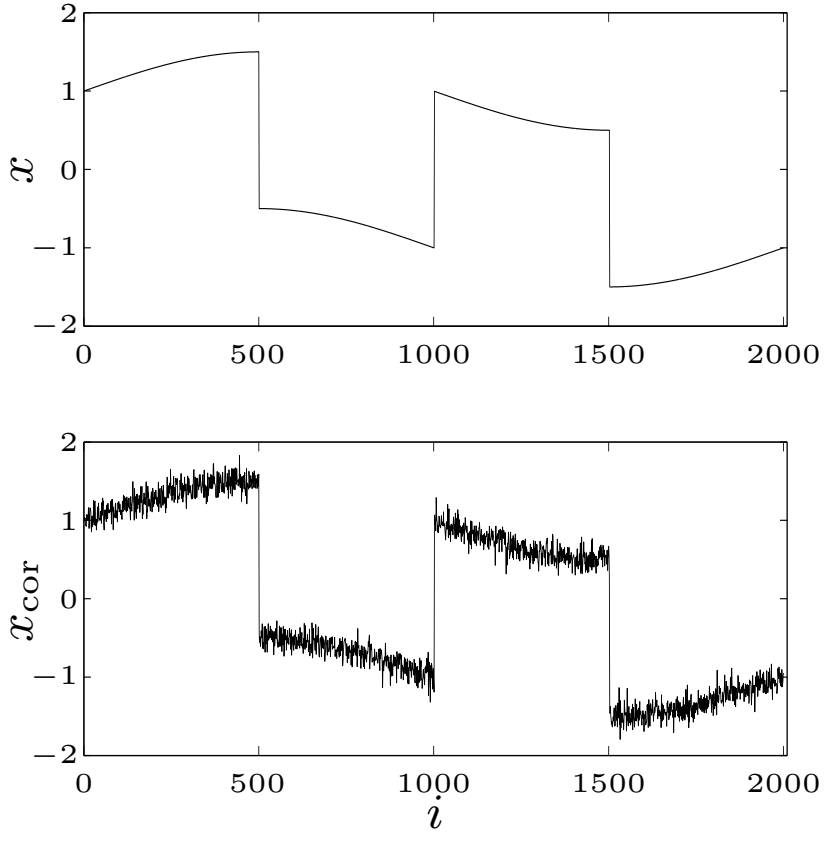


original signal x and noisy signal x_{cor}

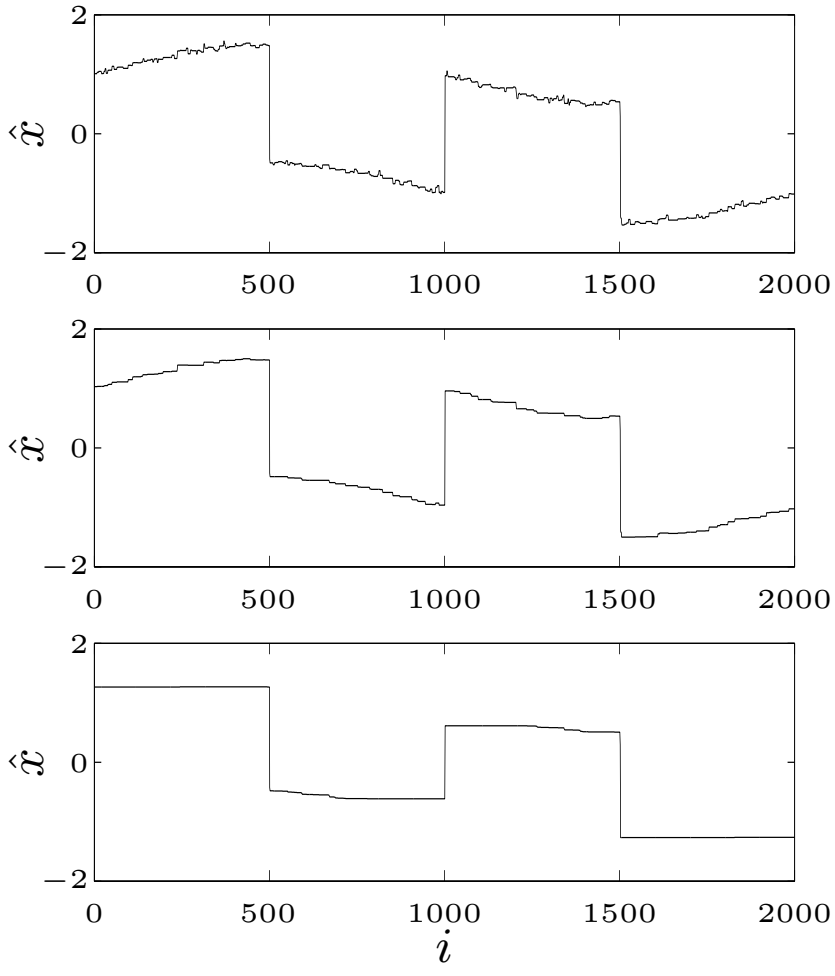


three solutions on trade-off curve $\|\hat{x} - x_{\mathrm{cor}}\|_2$ versus $\phi_{\mathrm{quad}}(\hat{x})$

quadratic smoothing smooths out noise **and** sharp transitions in signal



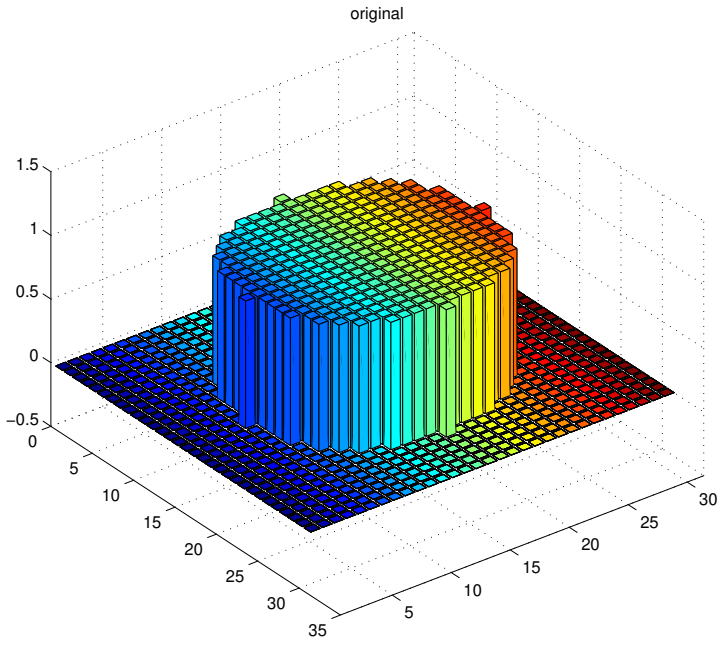
original signal x and noisy signal x_{cor}

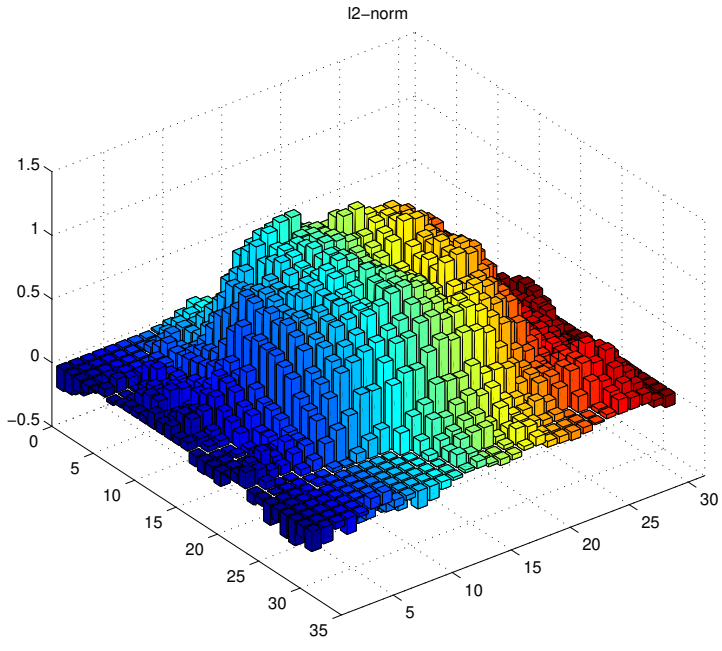


three solutions on trade-off curve $\|\hat{x} - x_{\mathrm{cor}}\|_2$ versus $\phi_{\mathrm{tv}}(\hat{x})$

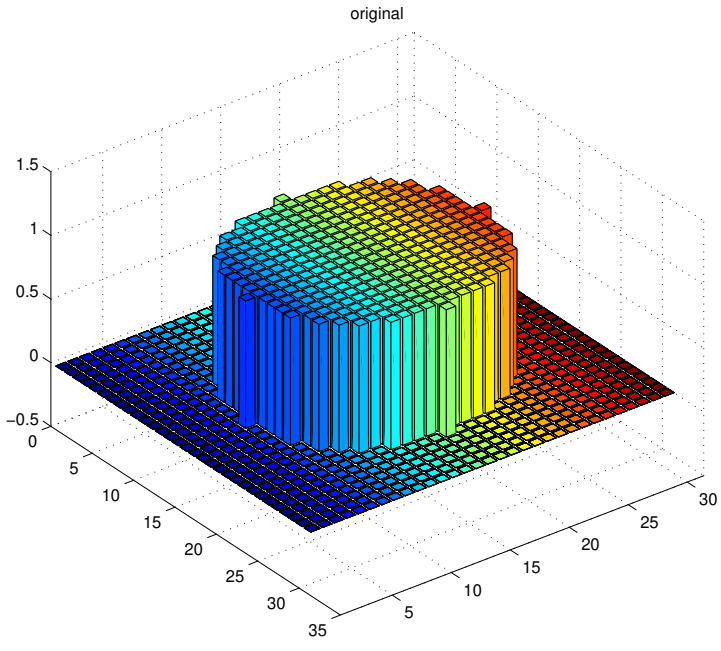
total variation smoothing preserves sharp transitions in signal

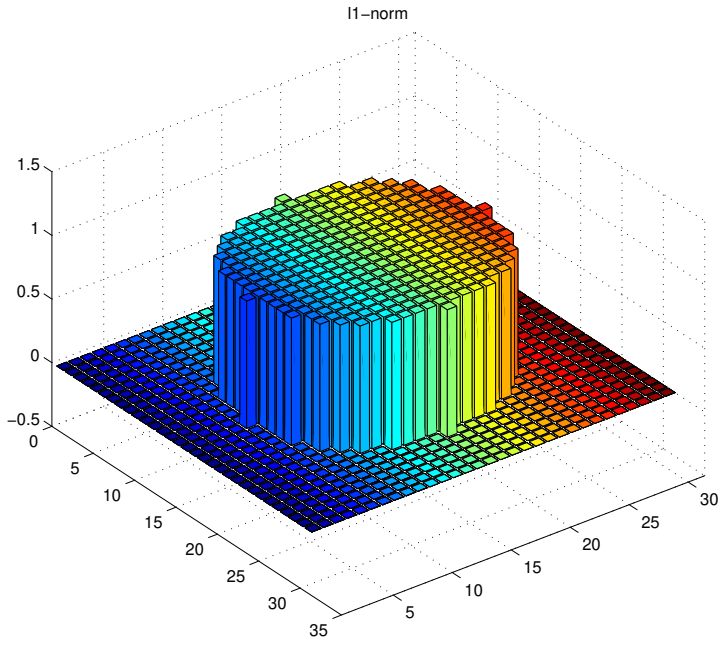
Total variation denoising





Total variation denoising





Total variation denoising

Original

Noisy image

Denoised image

Group lasso

- problem:

$$\text{minimize} \quad f(x) + \lambda \sum_{g=1}^{G} \|x_g\|_2$$

  with $x = (x_1, \ldots, x_G)$

- like lasso, but require groups of variables to be zero or not

- key is that the ℓ_2 norm is **not** squared

- by choosing groups carefully, can impose sophisticated sparsity patterns on the solutions (*e.g.*, nonzero variables being contiguous)

Regularized loss minimization

- general paradigm: **regularized loss minimization**

$$\text{minimize} \quad l(w) + \lambda r(w)$$

  - loss l measures lack of fit on training data (least squares, negative log likelihood, hinge loss, . . .); usually is additive in the training examples
  - regularizer r measures model complexity and can be used to promote useful or assumed forms of structure (*e.g.*, sparsity); usually does not depend on the training data
  - $\lambda > 0$ controls tradeoff and is set with cross validation

- includes all the models discussed so far (and more) as special cases
- l and r may have probabilistic motivations or not
- if both l and r are convex, we can solve these

Approaches to machine learning

1. regularized loss minimization (optimization)
2. Bayesian statistics (probability)