

Optimization Formulations of Data Analysis Problems

Stephen Wright

University of Wisconsin-Madison

UQ, June 2018

Outline

Data Analysis, Machine Learning, Data Science

- Context: Data Science
- Fundamentals of Data Analysis
- Fundamentals of Optimization
- Relating Data Science and Optimization
- Formulating specific data science problems as optimization problems

Optimization and Data Science

Optimization is being revolutionized by its interactions with machine learning and data analysis.

- New algorithms, and new interest in *old* algorithms;
- Challenging formulations and new paradigms;
- Renewed emphasis on certain topics: convex optimization algorithms, complexity, structured nonsmoothness, now nonconvex optimization.
- Large research community now working on the machine learning / optimization spectrum. The optimization / ML interface is a key component of many top conferences (ISMP, SIOPT, NIPS, ICML, COLT, AISTATS, ...) and journals (Math Programming, SIOPT,).

Data Science

Related Terms: AI, Data Analysis, Machine Learning, Statistical Inference, Data Mining.

- Extract meaning from data: Understand statistical properties, learn important features and fundamental structures in the data.
- Use this knowledge to make predictions about other, similar data.

Highly multidisciplinary area!

- Foundations in Statistics;
- Computer Science: AI, Machine Learning, Databases, Parallel Systems;
- **Optimization** provides a toolkit of modeling/formulation and algorithmic techniques.

Modeling and domain-specific knowledge is vital: “80% of data analysis is spent on the process of cleaning and preparing the data.”
[Dasu and Johnson, 2003].

(Most academic research deals with the other 20%.)

The Age of “Big Data”

New “Data Science Centers” at many institutions, new degree programs (e.g. Undergrad Majors and MS in Data Science), new funding initiatives (e.g. NSF’s TRIPODS).

- Huge amounts of data are collected, routinely and continuously.
 - ▶ Consumer and citizen data: phone calls and text, social media apps, email, surveillance cameras, web activity, online shopping,...
 - ▶ Scientific data (particle colliders, satellites, biological / genomic, astronomical,...)
- **Affects everyone directly!**
- Powerful computers and new specialized architectures make it possible to handle larger data sets and analyze them more thoroughly.
- Methodological innovations in some areas. e.g. **Deep Learning**.
 - ▶ Speech recognition in smart phones
 - ▶ AlphaGo: Deep Learning for Go.
 - ▶ Image recognition

Typical Setup

After cleaning and formatting, obtain a data set of m objects:

- Vectors of features: $a_j, j = 1, 2, \dots, m$.
- Outcome / observation / label y_j for each feature vector.

The outcomes y_j could be:

- a **real number**: **regression**
- a **label** indicating that a_j lies in one of M classes (for $M \geq 2$):
classification
- **multiple labels**: classify a_j according to multiple criteria.
- **no labels** (y_j is null):
 - ▶ **subspace identification**: Locate low-dimensional subspaces that approximately contain the (high-dimensional) vectors a_j ;
 - ▶ **clustering**: Partition the a_j into a few clusters.

(Structure may reveal which features in the a_j are important / distinctive, or enable predictions to be made about new vectors a .)

Fundamental Data Analysis Task

Seek a function ϕ that:

- approximately maps a_j to y_j for each j : $\phi(a_j) \approx y_j, j = 1, 2, \dots, m$.
- if there are no labels y_j , or if some labels are missing, seek ϕ that does something useful with the data $\{a_j\}$, e.g. assigns each a_j to an appropriate cluster or subspace.
- satisfies some additional properties — simplicity, structure — that make it “plausible” for the application, robust to perturbations in the data, generalizable to other data samples.

Can usually define ϕ in terms of some parameter vector x — thus identification of ϕ becomes a **data-fitting problem**: Find the best x .

Objective function in this problem often built up of m terms that capture mismatch between predictions and observations for data item (a_j, y_j) .

The process of finding ϕ is called **learning** or **training**.

What's the use of the mapping ϕ ?

- **Analysis:** ϕ — especially the parameter x that defines it — reveals structure in the data. Examples:
 - ▶ Feature selection: reveal the components of vectors a_j that are most important in determining the outputs y_j , and quantifies the importance of these features.
 - ▶ Uncovers some hidden structure, e.g.
 - ★ finds some low-dimensional subspaces that contain the a_j ;
 - ★ find clusters that contain the a_j ;
 - ★ find a decision tree that builds intuition about how outputs y_j depend on inputs a_j .
- **Prediction:** Given new data vectors a_k , predict outputs $y_k \leftarrow \phi(a_k)$.

Complications

The data items (a_j, y_j) available for training and testing are viewed as an **empirical sample** drawn from some **underlying reality**. Want our analysis of the sample to work well on the unknown underlying set.

- **noise or errors** in a_j and y_j . Would like ϕ (and x) to be robust to this — solution should **generalize** to perturbations of the observed data. Often achieve this via **regularized** formulations.
- **avoid overfitting**: Want to avoid overfitting to the particular empirical sample. (Training should produce a similar result for other samples from the same underlying data set.) Again, **generalization / regularization** plays an important role in the formulation.
- **missing data**: Vectors a_j may be missing elements (but may still contain useful information).
- **missing labels**: Some or all y_j may be missing or null — semi-supervised or unsupervised learning.
- **online learning**: Data (a_j, y_j) is arriving in a stream rather than all known up-front.

Optimization

In optimization, we seek to minimize some **objective function** by choosing the best values for a set of **variables**, subject possibly to some **constraints** on the variables.

The three ingredients — **variables**, **objective function**, and **constraints** — are all important. They can have different characteristics that make the optimization problem easy or hard.

The **variables** could be

- vectors of real numbers;
- vectors of binary variables $\{0, 1\}$ or integers;
- square or rectangular matrices of real numbers;
- functions in a Hilbert space;

or some combination of all of these.

The **constraints** could be

- non-existent: “unconstrained” optimization;
- bounds on the real and integer variables;
- more complicated algebraic expressions or geometric restrictions;
- positive definiteness of the matrix variables: $X \succeq 0$;
- uncertain (depending also on some random variable);
- chance-constrained (e.g. in a list of 100 constraints, want at least 95 of them to be satisfied).

The **objective function** could be

- smooth (multiple-times differentiable);
- nonsmooth but continuous;
- an expectation over some random variable: $\mathbb{E}_{\xi} f(x; \xi)$ (where x is the optimization variable and ξ is the random variable).

Some optimization problems have no objective. (In this case we simply seek a point that satisfies the constraints — a *feasible* point.)

Optimization Research

Optimization research includes

- the process of **formulating / modeling** practical problems as optimization problems;
- studying the mathematical properties of optimization formulations, e.g. sensitivity to uncertainty in their data;
- developing computational algorithms for solving these problems, and studying the mathematical properties of these algorithms;
- developing and testing software that implements these algorithms.

Continuous Optimization and Data Analysis

Optimization is a major source of algorithms for machine learning and data analysis.

- **Optimization Formulations** translate statistical principles (e.g. risk, likelihood, significance, generalizability) into measures and functions that can be attacked with an algorithm.
- **Optimization Algorithms** provide practical means to solve these problems, but the “black-box” approach often doesn’t work well. Structure and context are important.
- **Duality** is valuable in several cases (e.g. kernel learning).
- **Nonsmoothness** appears often e.g. as a formulation tool to promote generalizability, but often in a highly structured way that can be exploited by algorithms.

ML's Influence on (Continuous) Optimization

The needs of ML and the ML community have also influenced optimization greatly.

ML has a different perspective on some important algorithmic issues:

- Computational complexity and global convergence rates are more interesting.
- Fast local convergence rates are less interesting, possibly because fast convergence often cuts in below the level of accuracy required for minimization of the empirical risk.
- Prefer cheaper, approximate solutions over expensive, accurate solutions (for the same reason).

ML's Influence on (Continuous) Optimization

The needs of ML (including summation form, nonsmooth regularization) has caused revival, reexamination, and development of known approaches, particularly first-order and “zero order” methods.

- stochastic gradient
- accelerated gradient
- coordinate descent
- conditional gradient (Frank-Wolfe)
- sparse and regularized optimization e.g. forward-backward.
- augmented Lagrangian, ADMM
- (sampled Newton and quasi-Newton)

This trend continues in nonconvex formulations:

- steepest descent (+ noise)
- trust-region methods
- Nonlinear conjugate gradient and L-BFGS
- Newton-conjugate gradient.

“Algorithms” in ML and Optimization

In ML, the “algorithm” is the process that maps a training data set S to a predictor. (Sometimes randomized, e.g. stochastic gradient.)

Aim for **generalizability**: Predictor predicts well on unseen data.

When the ML algorithm is implemented via optimization it's broken down into two stages: the **optimization formulation** and the **optimization algorithm**.

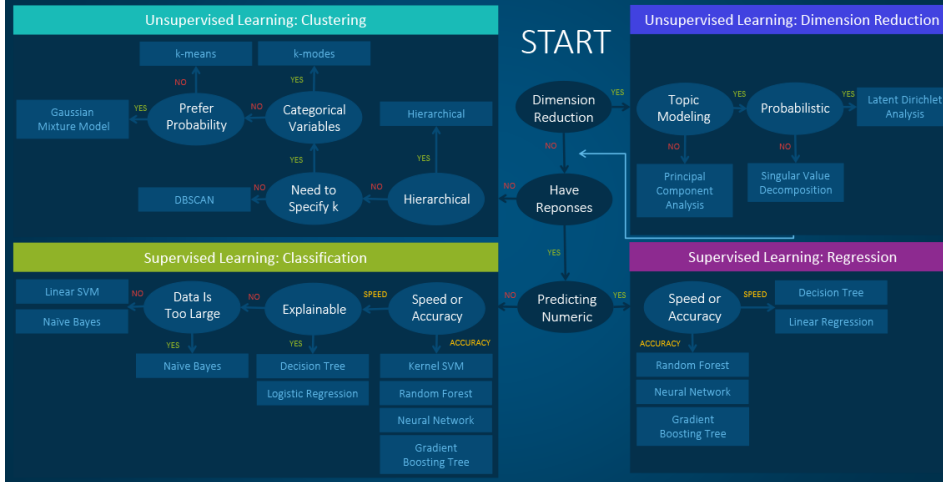
The responsibility for good generalizability thus falls both on the optimization formulation and the optimization algorithm.

This overloads the optimization algorithm! Traditionally, optimization algorithms were just tasked with finding the solution of the formulation. Nowadays they have to pursue a more nebulous and unfamiliar goal.

New questions arise, e.g.

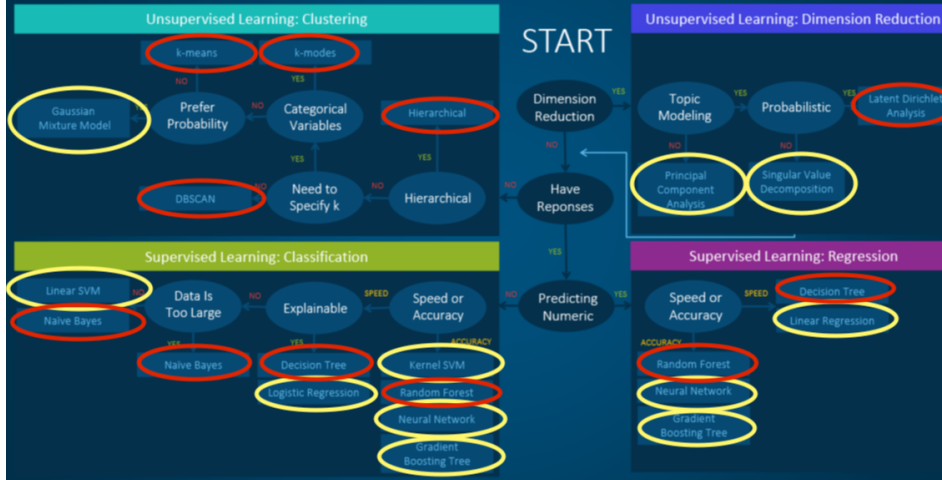
- Does small-batch SGD give results with better generalizability?
- Is the algorithm finding a “low-norm” solution that generalizes better?

Machine Learning Algorithms Cheat Sheet



¹<https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/>

Machine Learning Algorithms Cheat Sheet



There's a lot of continuous optimization here (yellow)!

Application I: (Linear) Least Squares

$$\min_x f(x) := \frac{1}{2} \sum_{j=1}^m (a_j^T x - y_j)^2 = \frac{1}{2} \|Ax - y\|_2^2.$$

[Gauss, 1799], [Legendre, 1805]; see [Stigler, 1981].

Here the function mapping data to output is linear: $\phi(a_j) = a_j^T x$.

- ℓ_2 regularization reduces sensitivity of the solution x to **noise in y** .

$$\min_x \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_2^2.$$

- ℓ_1 regularization yields solutions x with few nonzeros:

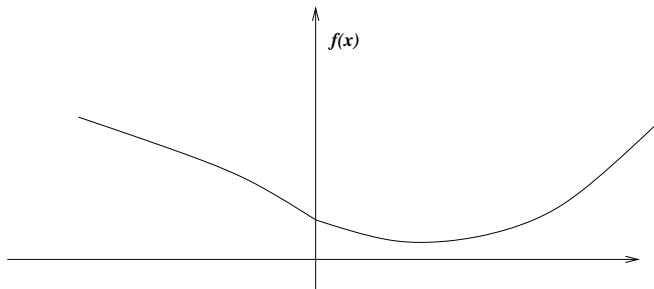
$$\min_x \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_1.$$

Feature selection: Nonzero locations in x indicate important components of a_j .

- Nonconvex separable regularizers (SCAD, MCP) have nice statistical properties, but lead to nonconvex optimization formulations.

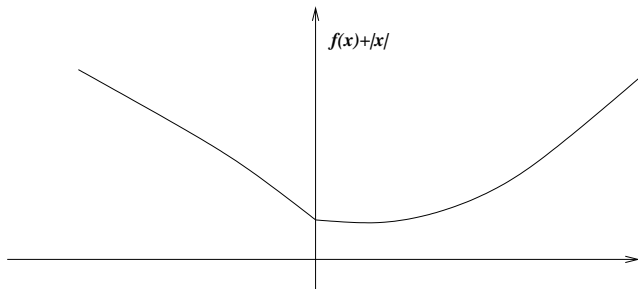
ℓ_1 regularization

Including a multiple of $\|x\|_1$ into the objective is a standard way to induce sparsity in the variable vector x .



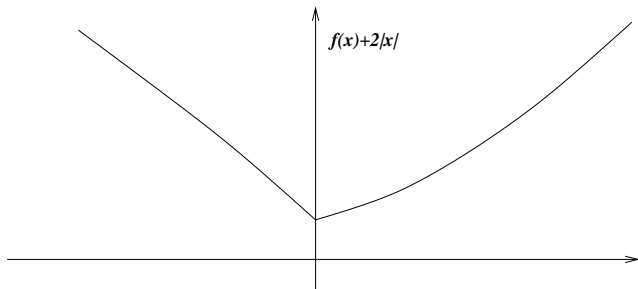
ℓ_1 regularization

Including a multiple of $\|x\|_1$ into the objective is a standard way to induce sparsity in the variable vector x .



ℓ_1 regularization

Including a multiple of $\|x\|_1$ into the objective is a standard way to induce sparsity in the variable vector x .



Application II: Matrix Completion

Regression over a structured matrix: Observe a matrix X by probing it with linear operators $\mathcal{A}_j(X)$, giving observations y_j , $j = 1, 2, \dots, m$. Solve a regression problem:

$$\min_X \frac{1}{2m} \sum_{j=1}^m (\mathcal{A}_j(X) - y_j)^2 = \frac{1}{2m} \|\mathcal{A}(X) - y\|_2^2.$$

Each \mathcal{A}_j may observe a single element of X , or a linear combination of elements. Can be represented as a matrix A_j , so that $\mathcal{A}_j(X) = \langle A_j, X \rangle$.

Seek the “simplest” X that satisfies the observations. Nuclear-norm (sum-of-singular-values) regularization term induces low rank on X :

$$\min_X \frac{1}{2m} \|\mathcal{A}(X) - y\|_2^2 + \lambda \|X\|_*, \quad \text{for some } \lambda > 0.$$

[Recht et al., 2010]

Explicit Low-Rank Parametrization

Compact, nonconvex formulation is obtained by parametrizing X directly:

$$X = LR^T, \quad \text{where } L \in \mathbb{R}^{m \times r}, R \in \mathbb{R}^{n \times r},$$

where r is known (or suspected) rank.

$$\min_{L,R} \frac{1}{2m} \sum_{j=1}^m (\mathcal{A}_j(LR^T) - y_j)^2.$$

For **symmetric** X , have $L = R$, so $X = ZZ^T$, where $Z \in \mathbb{R}^{n \times r}$.

(No need for regularizer — **rank is hard-wired** into the formulation.)

Despite the nonconvexity, near-global minima can be found when \mathcal{A}_j are **incoherent**. Use appropriate initialization [Candès et al., 2014], [Zheng and Lafferty, 2015] or the observation that all local minima are near-global [Bhojanapalli et al., 2016].

Application III: Nonnegative Matrix Factorization

Given $m \times n$ matrix Y , seek factors L ($m \times r$) and R ($n \times r$) that are element-wise positive, such that $LR^T \approx Y$.

$$\min_{L,R} \frac{1}{2} \|LR^T - Y\|_F^2 \text{ subject to } L \geq 0, R \geq 0.$$

Applications in computer vision, document clustering, chemometrics, ...

Could combine with matrix completion, when not all elements of Y are known, if it makes sense on the application to have nonnegative factors.

If positivity constraint were not present, could solve this in closed form with an SVD, since Y is observed completely.

Application IV: Sparse Inverse Covariance

Let $Z \in \mathbb{R}^p$ be a (vector) random variable with zero mean. Let z_1, z_2, \dots, z_N be samples of Z . Sample covariance matrix (estimates covariance between components of Z):

$$S := \frac{1}{N-1} \sum_{\ell=1}^N z_\ell z_\ell^T.$$

Seek a **sparse inverse covariance matrix**: $X \approx S^{-1}$.

X reveals dependencies between components of Z : $X_{ij} = 0$ if the i and j components of Z are conditionally independent.

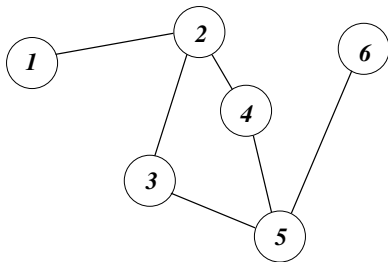
Do nodes i and j influence each other directly, or only indirectly via other nodes?

Obtain X from the regularized formulation:

$$\min_X \langle S, X \rangle - \log \det(X) + \lambda \|X\|_1, \quad \text{where } \|X\|_1 = \sum_{i,j} |X_{ij}|.$$

[d'Aspremont et al., 2008, Friedman et al., 2008].

Reveals Network Structure. Example with $p = 6$.



$$X = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 \\ 0 & * & * & 0 & * & 0 \\ 0 & * & 0 & * & * & 0 \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

Application V: Sparse Principal Components (PCA)

Seek **sparse** approximations to the leading eigenvectors of the sample covariance matrix S .

For the leading sparse principal component, solve

$$\max_{v \in \mathbb{R}^n} v^T S v = \langle S, v v^T \rangle \quad \text{s.t. } \|v\|_2 = 1, \|v\|_0 \leq k,$$

for some given $k \in \{1, 2, \dots, n\}$. Convex relaxation replaces $v v^T$ by an $n \times n$ positive semidefinite proxy M :

$$\max_{M \in \mathbb{S}^n} \langle S, M \rangle \quad \text{s.t. } M \succeq 0, \langle I, M \rangle = 1, \|M\|_1 \leq R,$$

where $\|\cdot\|_1$ is the sum of absolute values [d'Aspremont et al., 2007].

Adjust the parameter R to obtain desired sparsity.

Sparse PCA (rank r)

For sparse leading rank- r eigenspace, seek $V \in \mathbb{R}^{n \times r}$ with **orthonormal columns** such that $\langle S, VV^T \rangle$ is maximized, and V has at most k nonzero rows. Convex relaxation:

$$\max_{M \in \mathbb{S}^{\mathbb{R}^{n \times n}}} \langle S, M \rangle \quad \text{s.t. } 0 \preceq M \preceq I, \langle I, M \rangle \leq r, \|M\|_1 \leq R.$$

Explicit low-rank formulation is

$$\max_{F \in \mathbb{R}^{n \times r}} \langle S, FF^T \rangle \quad \text{s.t. } \|F\|_2 \leq 1, \|F\|_{2,1} \leq \bar{R},$$

where $\|F\|_{2,1} := \sum_{i=1}^n \|F_{i \cdot}\|_2$.

[Chen and Wainwright, 2015]

Application VI: Sparse + Low-Rank

Given $Y \in \mathbb{R}^{m \times n}$, seek low-rank M and sparse S such that $M + S \approx Y$.

Applications:

- Robust PCA: Sparse S represents “outlier” observations.
- Foreground-Background separation in video processing.
 - ▶ Each column of Y is one frame of video, each row is a single pixel evolving in time.
 - ▶ Low-rank part M represents background, sparse part S represents foreground.

Convex formulation:

$$\min_{M,S} \|M\|_* + \lambda \|S\|_1 \quad \text{s.t. } Y = M + S.$$

[Candès et al., 2011, Chandrasekaran et al., 2011]

Sparse + Low-Rank: Compact Formulation

Compact formulation: Variables $L \in \mathbb{R}^{n \times r}$, $R \in \mathbb{R}^{m \times r}$, $S \in \mathbb{R}^{m \times n}$ sparse.

$$\min_{L,R,S} \frac{1}{2} \|LR^T + S - Y\|_F^2 + \lambda \|S\|_1 \quad (\text{fully observed})$$

$$\min_{L,R,S} \frac{1}{2} \|P_\Phi(LR^T + S - Y)\|_F^2 + \lambda \|S\|_1 \quad (\text{partially observed}),$$

where Φ represents the locations of the observed entries.

[Chen and Wainwright, 2015, Yi et al., 2016].

(For well-posedness, need to assume that the “true” L , R , S satisfy certain incoherence properties.)

Application VII: Subspace Identification

Given vectors $a_j \in \mathbb{R}^n$ with **missing entries**, find a subspace of \mathbb{R}^n such that all “completed” vectors a_j lie approximately in this subspace.

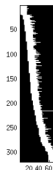
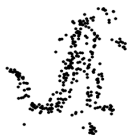
If $\Omega_j \subset \{1, 2, \dots, n\}$ is the set of observed elements in a_j , seek $X \in \mathbb{R}^{n \times d}$ such that

$$[a_j - Xs_j]_{\Omega_j} \approx 0,$$

for some $s_j \in \mathbb{R}^d$ and all $j = 1, 2, \dots$

[Balzano et al., 2010, Balzano and Wright, 2014].

Application: Structure from motion. Reconstruct opaque object from planar projections of surface reference points.



Application VIII: Linear Support Vector Machines

Each item of data belongs to one of two classes: $y_j = +1$ and $y_j = -1$.

Seek (x, β) such that

$$a_j^T x - \beta \geq 1 \quad \text{when } y_j = +1;$$

$$a_j^T x - \beta \leq -1 \quad \text{when } y_j = -1.$$

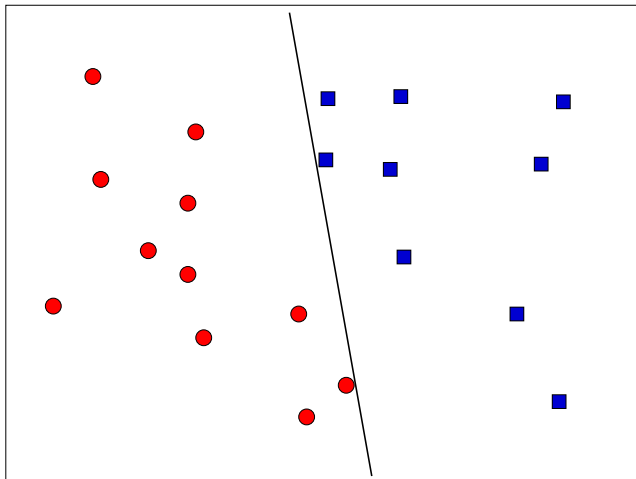
The mapping is $\phi(a_j) = \text{sign}(a_j^T x - \beta)$.

Design an objective so that the j th loss term is zero when $\phi(a_j) = y_j$, positive otherwise. A popular one is **hinge loss**:

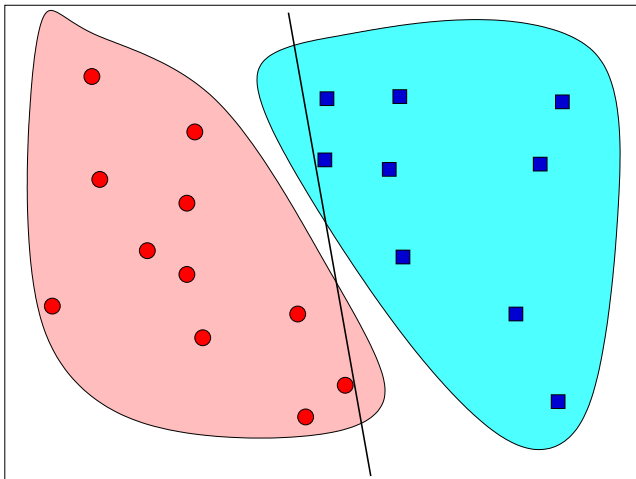
$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^m \max(1 - y_j(a_j^T x - \beta), 0).$$

Add a **regularization term** $(\lambda/2)\|x\|_2^2$ for some $\lambda > 0$ to maximize the margin between the classes.

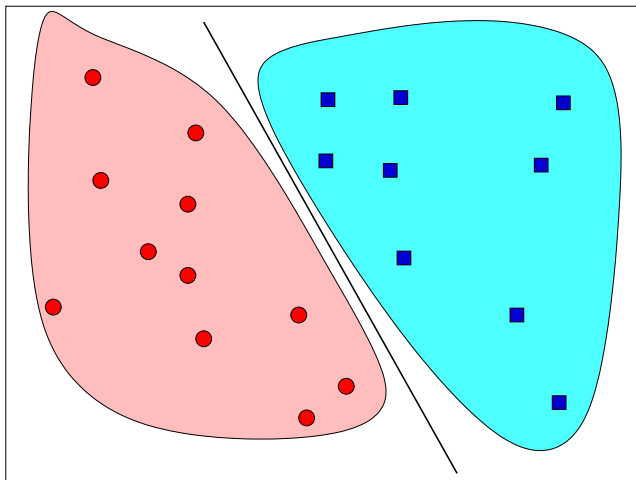
Regularize for Generalizability



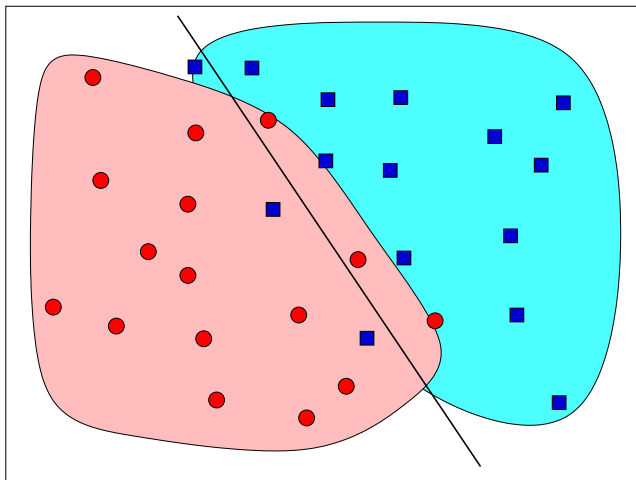
Regularize for Generalizability



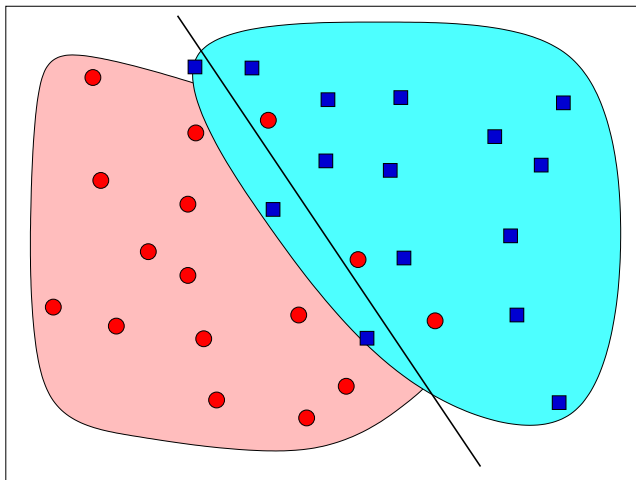
Regularize for Generalizability



Regularize for Generalizability



Regularize for Generalizability



Application IX: Nonlinear SVM

Data a_j , $j = 1, 2, \dots, m$ may not be *separable* neatly into two classes $y_j = +1$ and $y_j = -1$. Apply a nonlinear transformation $a_j \rightarrow \psi(a_j)$ (“lifting”) to make separation more effective. Seek (x, β) such that

$$\psi(a_j)^T x - \beta \geq 1 \quad \text{when } y_j = +1;$$

$$\psi(a_j)^T x - \beta \leq -1 \quad \text{when } y_j = -1.$$

Leads to the formulation:

$$\min_x \frac{1}{m} \sum_{j=1}^m \max(1 - y_j(\psi(a_j)^T x - \beta), 0) + \frac{1}{2} \lambda \|x\|_2^2.$$

Can avoid defining ψ explicitly by using instead the **dual** of this QP.

Nonlinear SVM: Dual

Dual is a quadratic program in m variables, with simple constraints:

$$\min_{\alpha \in \mathbb{R}^m} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \quad \text{s.t.} \quad 0 \leq \alpha \leq (1/\lambda)e, \quad y^T \alpha = 0.$$

where $Q_{k\ell} = y_k y_\ell \psi(a_k)^T \psi(a_\ell)$, $y = (y_1, y_2, \dots, y_m)^T$, $e = (1, 1, \dots, 1)^T$.

No need to choose $\psi(\cdot)$ explicitly. Instead choose a kernel K , such that

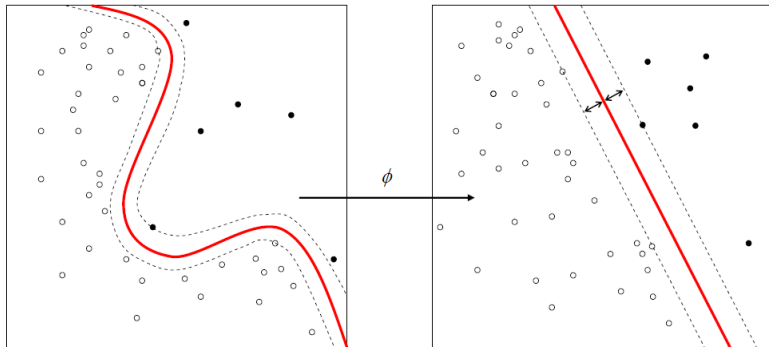
$$K(a_k, a_\ell) \sim \psi(a_k)^T \psi(a_\ell).$$

[Boser et al., 1992, Cortes and Vapnik, 1995]. “Kernel trick.”

Gaussian kernels are popular:

$$K(a_k, a_\ell) = \exp(-\|a_k - a_\ell\|^2 / (2\sigma)), \quad \text{for some } \sigma > 0.$$

Nonlinear SVM



Application X: Logistic Regression

Binary logistic regression is similar to binary SVM, except that we seek a function p that gives **odds** of data vector a being in class 1 or class -1 , rather than making a simple prediction.

Seek odds function p parametrized by $x \in \mathbb{R}^n$:

$$p(a; x) := (1 + e^{a^T x})^{-1}.$$

Choose x so that $p(a_j; x) \approx 1$ when $y_j = 1$ and $p(a_j; x) \approx 0$ when $y_j = -1$.

Choose x to minimize a negative log likelihood function:

$$\mathcal{L}(x) = -\frac{1}{m} \left[\sum_{y_j=-1} \log(1 - p(a_j; x)) + \sum_{y_j=1} \log p(a_j; x) \right]$$

Sparse solutions x are interesting because they indicate which components of a_j are critical to classification. Can solve: $\min_z \mathcal{L}(z) + \lambda \|z\|_1$.

Multiclass Logistic Regression

Have M classes instead of just 2. M can be large e.g. identify phonemes in speech, identify line outages in a power grid.

Labels $y_{j\ell} = 1$ if data point j is in class ℓ ; $y_{j\ell} = 0$ otherwise; $\ell = 1, \dots, M$.

Find subvectors $x_{[\ell]}$, $\ell = 1, 2, \dots, M$ such that if a_j is in class k we have

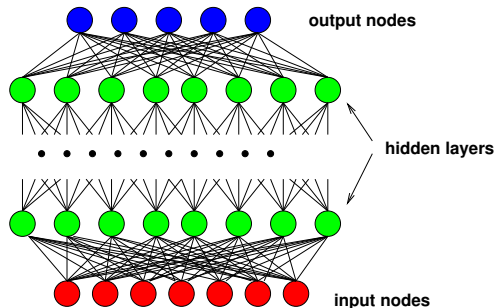
$$a_j^T x_{[k]} \gg a_j^T x_{[\ell]} \quad \text{for all } \ell \neq k.$$

Find $x_{[\ell]}$, $\ell = 1, 2, \dots, M$ by minimizing a negative log-likelihood function:

$$f(x) = -\frac{1}{m} \sum_{j=1}^m \left[\sum_{\ell=1}^M y_{j\ell} (a_j^T x_{[\ell]}) - \log \left(\sum_{\ell=1}^M \exp(a_j^T x_{[\ell]}) \right) \right]$$

Can use **group LASSO** regularization terms to select important features from the vectors a_j , by imposing a common sparsity pattern on all $x_{[\ell]}$.

Application XI: Deep Learning



Inputs are the vectors a_j , outputs are **odds** of a_j belonging to each class (as in multiclass logistic regression).

At each layer, inputs are converted to outputs by a **linear transformation** composed with an **element-wise function**:

$$a^{\ell+1} = \sigma(W^\ell a^\ell + g^\ell),$$

where a^ℓ is node values at layer ℓ , (W^ℓ, g^ℓ) are *parameters* in the network, σ is the element-wise function.

Deep Learning

The element-wise function σ makes transformations to scalar input:

- Logistic function: $t \rightarrow 1/(1 + e^{-t})$;
- Hinge: $t \rightarrow \max(t, 0)$: “ReLU”;
- Bernoulli: random! $t \rightarrow 1$ with probability $1/(1 + e^{-t})$ and $t \rightarrow 0$ otherwise (inspired by neuron behavior).

The example depicted shows a completely connected network — but more typically networks are engineered to the application (speech processing, object recognition, ...).

- local aggregation of inputs: **pooling**;
- restricted connectivity + constraints on weights (elements of W^ℓ matrices): **convolutions**.
- connections that skip a layer: **ResNet**. Each layer fits the “residual” of the fit from the layer below.

Training Deep Learning Networks

The network contains many **parameters** — (W^ℓ, g^ℓ) , $\ell = 1, 2, \dots, L$ in the notation above — that must be selected by **training** on the data (a_j, y_j) , $j = 1, 2, \dots, m$. Objective has the form:

$$\sum_{j=1}^m h(x; a_j, y_j)$$

where $x = (W^1, g^1, W^2, g^2, \dots)$ are the parameters in the model and h measures the mismatch between observed output y_j and the outputs produced by the model (as in multiclass logistic regression).

Number of parameters (elements in x) is often vastly greater than the number of data points — yet “overfitting” is not necessarily a problem!

Nonlinear, Nonconvex, usually **Nonsmooth**.

Many software packages available for training: Caffe, PyTorch, Tensor Flow, Theano,... Many run on GPUs.

How Does a Neural Network Make The Problem Easier?

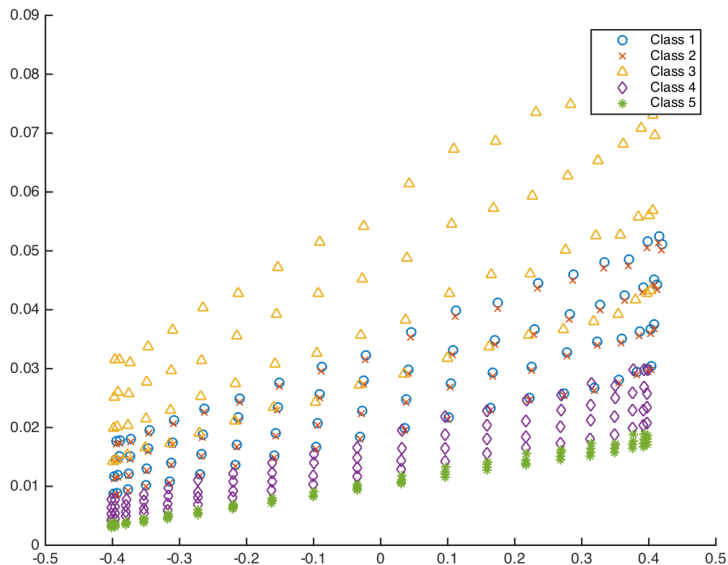
Can think of the neural network as transforming the raw data in a way that makes the ultimate task (regression, classification) easier.

We consider a multiclass classification application in power systems. The *raw data* is PMU measurements at different points in a power grid, under different operating conditions. The goal is to use this data to detect line outages. Each class corresponds to outage of a particular line.

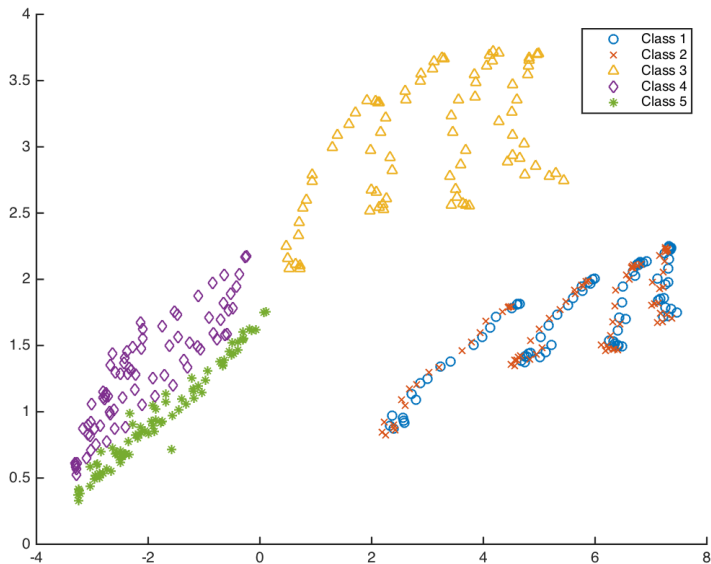
High-dimensional. Can illustrate by doing a singular value decomposition of the data matrix and plotting pairs of principal components on a 2-d graph.

Do this **before** and **after** transformation. One hidden layer with 200 nodes.

Raw Data (Before Transformation)



After Transformation by One Layer



Summary

Optimization provides powerful frameworks for formulating and solving problems in data analysis and machine learning.

BUT it's usually not enough to just formulate these problems and use off-the-shelf optimization technology to solve them. The algorithms need to be customized to the problem structure (in particular, large amount of data) and the context.

Research in this area has exploded over the past decade and is still going strong, with a great many unanswered questions. (Many of them in deep learning.)

References I



Balzano, L., Nowak, R., and Recht, B. (2010).
Online identification and tracking of subspaces from highly incomplete information.
In 48th Annual Allerton Conference on Communication, Control, and Computing, pages 704–711.
<http://arxiv.org/abs/1006.4046>.



Balzano, L. and Wright, S. J. (2014).
Local convergence of an algorithm for subspace identification from partial data.
Foundations of Computational Mathematics, 14:1–36.
DOI: 10.1007/s10208-014-9227-7.



Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016).
Global optimality of local search for low-rank matrix recovery.
Technical Report arXiv:1605.07221, Toyota Technological Institute.





Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992).
A training algorithm for optimal margin classifiers.
In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pages 144–152.





Candès, E., Li, X., and Soltanolkotabi, M. (2014).
Phase retrieval via a Wirtinger flow.
Technical Report arXiv:1407.1065, Stanford University.


References II


 Candès, E. and Recht, B. (2009).
Exact matrix completion via convex optimization.
Foundations of Computational Mathematics, 9:717–772.

 Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011).
Robust principal component analysis?
Journal of the ACM, 58.3:11.

 Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. (2011).
Rank-sparsity incoherence for matrix decomposition.
SIAM Journal on Optimization, 21(2):572–596.

 Chen, Y. and Wainwright, M. J. (2015).
Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees.
Technical Report arXiv:1509.03025, University of California-Berkeley.

 Cortes, C. and Vapnik, V. N. (1995).
Support-vector networks.
Machine Learning, 20:273–297.

 d'Aspremont, A., Banerjee, O., and El Ghaoui, L. (2008).
First-order methods for sparse covariance selection.
SIAM Journal on Matrix Analysis and Applications, 30:56–66.

References III



d'Aspremont, A., El Ghaoui, L., Jordan, M. I., and Lanckriet, G. (2007).
A direct formulation for sparse PCA using semidefinite programming.
SIAM Review, 49(3):434–448.



Dasu, T. and Johnson, T. (2003).
Exploratory Data Mining and Data Cleaning.
John Wiley & Sons.



Friedman, J., Hastie, T., and Tibshirani, R. (2008).
Sparse inverse covariance estimation with the graphical lasso.
Biostatistics, 9(3):432–441.



Keshavan, R. H., Montanari, A., and Oh, S. (2010).
Matrix completion from a few entries.
IEEE Transactions on Information Theory, 56(6):2980–2998.



Recht, B., Fazel, M., and Parrilo, P. (2010).
Guaranteed minimum-rank solutions to linear matrix equations via nuclear norm minimization.
SIAM Review, 52(3):471–501.



Stigler, S. M. (1981).
Gauss and the invention of least squares.
Annals of Statistics, 9(3):465–474.

References IV



Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015).

Going deeper with convolutions.

In *CVPR*.



Yi, X., Park, D., Chen, Y., and Caramanis, C. (2016).

Fast algorithms for robust pca via gradient descent.

Technical Report arXiv:1605.07784, University of Texas-Austin.



Zheng, Q. and Lafferty, J. (2015).

A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements.

Technical Report arXiv:1506.06081, Statistics Department, University of Chicago.