First-Order Optimization Algorithms for Machine Learning

Coordinate Optimization

Mark Schmidt

University of British Columbia

Summer 2020

Last Time: Structured Regularization

- We discussed total variation regularization,

$$\underset{w}{\text{argmin}}\, f(w) + \sum_{(i,j)\in E} \lambda_{ij}\|w_i - w_j\|,$$

  if we want w_i values to be similar across nodes in a graph.

- We discussed structured sparsity,

$$\underset{w}{\text{argmin}}\, f(w) + \sum_{g\in\mathcal{G}} \lambda_g\|w_g\|,$$

  where overlapping groups can enforce patterns of sparsity.

- Unfortunately, these regularizers are not "simple".
  - But we can efficiently approximate the proximal operator in all these cases.

Inexact Proximal-Gradient Methods

- For total-variation and overlapping group-L1, we can use Dykstra's algorithm
  - Iterative method that computes proximal operator for sum of "simple" functions.
- For nuclear-norm regularization, methods approximate top singular vectors.
  - Krylov subspace methods, randomized SVD approximations.
- Inexact proximal-gradient methods:
  - Proximal-gradient methods with an approximation to the proximal operator.
  - If approximation error decreases fast enough, same convergence rate:
    - To get $O(\rho^t)$ rate, error must be in $o(\rho^t)$.
- A related approach is the "proximal average" for sum of "simple":
  - Replace proximal operator of sum with average of proximal operators for each term.

Alternating Direction Method of Multipliers

- ADMM is also popular for structured sparsity problems

- Alternating direction method of multipliers (ADMM) solves:

$$\min_{Aw+Bv=c} f(w) + r(v).$$

- Alternates between proximal operators with respect to f and r.
  - We usually introduce new variables and constraints to convert to this form.
- We can apply ADMM to L1-regularization with an easy prox for f using

$$\min_w \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1 \quad \Leftrightarrow \quad \min_{v=Xw} \frac{1}{2}\|v - y\|^2 + \lambda\|w\|_1,$$

- For total-variation and structured sparsity we can use

$$\min_w f(w) + \|Aw\|_1 \quad \Leftrightarrow \quad \min_{v=Aw} f(w) + \|v\|_1.$$

- If prox can not be computed exactly: linearized ADMM.
  - But ADMM rate depends on tuning parameter(s) and iterations aren't sparse.

Conditional Gradient Method ("Frank-Wolfe")

- In some cases the projected-gradient step

$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k} \|v - w^k\|^2 \right\},$$

  may be hard to compute.

- Frank-Wolfe step is sometimes cheaper:

$$w^{k+\frac{1}{2}} = \underset{v \in \mathcal{C}}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) \right\},$$

  requires bounded $\mathcal{C}$, algorithm takes convex combination of w^k and $w^{k+\frac{1}{2}}$.

https://www.youtube.com/watch?v=24e08AX9Eww

- $O(1/t)$ rate for convex objectives, some linear results for strongly-convex.
  - Like Newton, iterations are affine-invariant (don't change with affine transformation).
  - Tends to be slower than projected-gradient in cases where they have similar costs.

Mirror Descent

- One generalization of the projected-gradient step

$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k} \|v - w^k\|^2 \right\},$$

  is the mirror descent step:

$$w^{k+1} = \underset{v \in \mathcal{C}}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{\alpha_k} D(v, w^k) \right\},$$

  where D is a Bregman divergence (generalization of squared Euclidean norm).
- Special cases:
  - Gradient descent: $D(v, w) = \frac{1}{2}\|v - w\|^2$.
  - Newton: $D(v, w) = \frac{1}{2}(v - w)^\top \nabla^2 f(w)(v - w)$.
  - Exponentiated gradient: $D(v, w) = KL(v \,||\, w)$.

$UV^\top$ Parameterization for Matrix Problems

- We discussed nuclear norm regularization problems,

$$\underset{W \in \mathbb{R}^{d \times k}}{\text{argmin}} \; f(W) + \lambda \|W\|_*,$$

  which gives a solution with a low rank representation $W = UV^\top$.
- But standard algorithms are too costly in many applications.
  - We often can't store W.

- Many recent approaches directly minimize under $UV^\top$ parameterization,

$$\underset{U \in \mathbb{R}^{d \times R}, V \in \mathbb{R}^{k \times R}}{\text{argmin}} \; f(UV^\top) + \lambda_U \|U\|_F^2 + \lambda_V \|V\|_F^2,$$

  and just regularize U and V (here we're using the Frobenius matrix norm).

$UV^\top$ Parameterization for Matrix Problems

- We used this approach in 340 for latent-factor models,

$$f(W,Z) = \frac{1}{2}\|ZW - X\|_F^2 + \frac{\lambda_1}{2}\|Z\|_F^2 + \frac{\lambda_2}{2}\|W\|_F^2.$$

- We can sometimes prove these non-convex re-formulation give a global solution.
  - Includes PCA.

- In other cases, people are working hard on finding assumptions where this is true.
  - These assumptions are typically unrealistically strong.
  - But it works well enough in practice that practitioners don't seem to care.

End of Part 1: Key Ideas

- Typical ML problems are written as optimization problem

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) + \lambda r(w).$$

- Gradient descent:
  - Applies when F is differentiable, yields iteration cost that is linear in d.
  - Needs $O(1/\epsilon)$ iterations in general, only $O(\log(1/\epsilon))$ for PL functions.
  - Faster versions like Nesterov's and Newton-like methods exist.
- Proximal gradient:
  - Applies when f_i is differentiable and r is "simple" (like L1-regularization).
  - Similar convergence properties to gradient descent, even for non-smooth r.
    - Faster than subgradient method for such problems.
  - Special case is projected gradient, which allows "simple" constraints.
  - Can be used for "structured" regularization, like group L1-regularization.

Outline

Transductive Learning

- Our usual supervised learning framework:

$$X = \begin{bmatrix} 0 & 0.7 & 0 & 0.3 & 0 & 0 \\ 0.3 & 0.7 & 0 & 0.6 & 0 & 0.01 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0.3 & 0.7 & 1.2 & 0 & 0.10 & 0.01 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

- In transductive learning, we also have unlabeled examples,

$$\bar{X} = \begin{bmatrix} 0.3 & 0 & 1.2 & 0.3 & 0.10 & 0.01 \\ 0.6 & 0.7 & 0 & 0.3 & 0 & 0.01 \\ 0 & 0.7 & 0 & 0.6 & 0 & 0 \\ 0.3 & 0.7 & 0 & 0 & 0.20 & 0.01 \end{bmatrix},$$

and our goal is only to label these particular examples.

- We don't worry about performance on other potential test examples.

Transductive Learning

- Transductive learning framework:
  1. We have n labeled examples (x^i, y^i).
  2. We have t unlabeled examples $\bar{x}^i$ that we want to label.

- This arises a lot:
  - Usually getting unlabeled data is easy but getting labeled data is hard $(t >> n)$.
  - Typically situation: small number of labeled and huge number of unlabeled.

- Sometimes classifying the data is an intermediate step:
  - Goal is to ulimately use labeled examples to do something else.
  - "I can label a small number of examples, if it helps labeling them all".

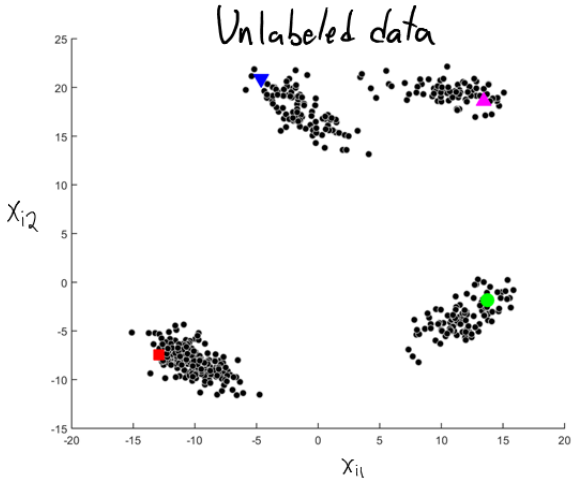
- Sometimes it's not possible to obtain labels for any x^i.
  - Predicting gene functions is limited by what we can measure.

Transductive Learning vs. (Semi-)Supervised Learning

- Transductive learning is a special case semi-supervised learning (SSL).
  - Learning with labeled and unlabeled examples.
- But transductive SSL has an unusual measure of performance:
  - We don't worry about "test error" (performance on all possible examples).
  - We only care about error for our "test" examples $\bar{x}^i$.
- Any supervised or semi-supervised method can be used for transduction.
  - Fit model, then apply it to unlabeled examples.
- But in transductive learning, we don't need a model that can predict on new $\tilde{x}^i$.
  - Some methods don't fit a generic model for mapping from x^i to y^i.

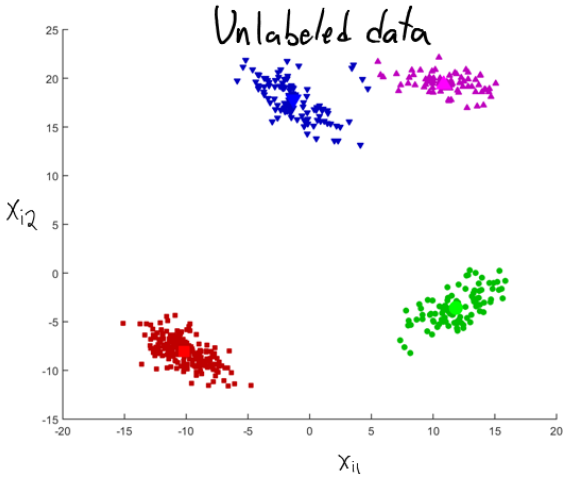
Transductive Learning

- Why should unlabeled data tell us anything about labels?
  - Usually, we assume that similar features $\rightarrow$ similar labels.



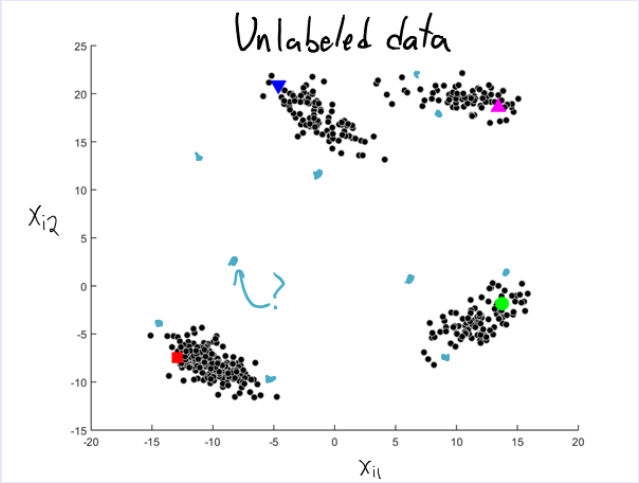
Transductive Learning

- Why should unlabeled data tell us anything about labels?
  - Usually, we assume that similar features $\rightarrow$ similar labels.



Digression: Transductive vs. Inductive SSL

- In transductive learning we don't need to be able to predict on new examples.
  - In inductive semi-supervised learning goal is to predict well on new examples.



Label Propagation (Graph-Based SSL)

- A weird idea: treat the $\bar{y}^i$ as variables that we can optimize.
  - Now optimize the $\bar{y}^i$ to encourage that "similar features have similar labels".

- Label propagation ("graph-based SSL") method:
  - Define weights w_{ij} saying how similar labeled example i is to unlabled example j.
    - Usually w_{ij} will be large if x^i and $\bar{x}^j$ are similar.
  - Define weights $\bar{w}_{ij}$ saying how similar unlabeled example i is to unlabeled example j.
  - Find labels $\bar{y}^i$ minimizing a measure of total variation on the label space:

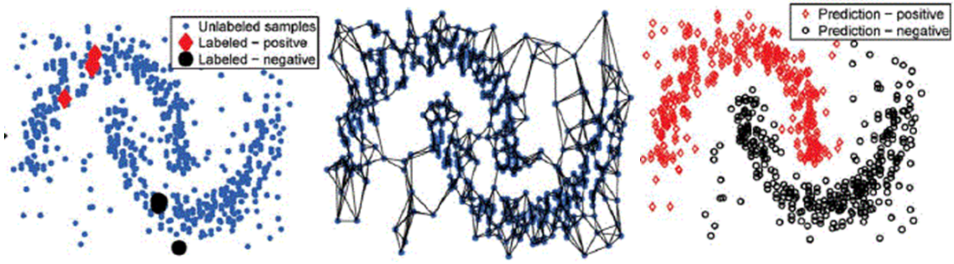
$$\underset{\bar{y} \in \mathbb{R}^t}{\text{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{t} w_{ij}(y^i - \bar{y}^j)^2 + \frac{1}{2} \sum_{i=1}^{t} \sum_{j=1}^{t} \bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2.$$

  - First term: unlabeled example should get similar labels to "close" labeled examples.
    - "If x^i and $\bar{x}^j$ are similar, then $\bar{y}^j$ should be similar to y^i."
  - Second term: similar unlabeled examples should have similar labels.
    - "Label information 'propagates' through the graph of $\bar{y}^i$ values".

Label Propagation (Graph-Based SSL)

- Label propagation is often surprisingly effective (even with few labeled examples).

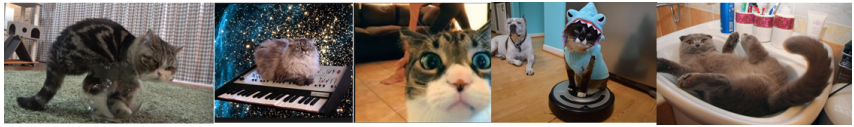
- A common choice of the weights (many variations exist):
  - Find the k-nearest neighbours of each example (among labeled and unlabeled).
  - Set w_{ij} and $\bar{w}_{ij}$ to 0 if nodes i and j aren't neighbours.
  - Otherwise, set these to some measure of similarity between features.



http://www.ee.columbia.edu/ln/dvmm/pubs/publications.html

Label Propagation for YouTube Tagging and Bioinformatics

- Label propagation doesn't necessarily need features.
  - Consider assigning "tags" to YouTube vidoes (e.g., "cat").



www.youtube.com

  - Construct a graph based on sequence of videos that people watch.
    - Give high weight if video 'A' is often followed/preceded by video 'B'.
  - Use label propagation to tag all videos.

- Becoming popular in bioinformatics:
  - Label a subset of genes using manual experiments.
  - Find out which genes interact using cheaper manual experiments.
  - Predict function/location/etc. of genes using label propagation.

Label Propagation Variations

- Many variations on label propagation exist:
  - Different ways to choose the graph/weights.
  - Multi-class versions,

$$\underset{\bar{Y}\in\mathbb{R}^{t\times k}}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}\|y^i - \bar{y}^j\|^2 + \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \bar{w}_{ij}\|\bar{y}^i - \bar{y}^j\|^2.$$

  - Other measures of similarity/distance,

$$\underset{\bar{y}\in\mathbb{R}^{t}}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{t} f_{ij}(y^i, \bar{y}^j) + \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} f_{ij}(\bar{y}^i, \bar{y}^j).$$

  - Variants where the given labels y^i are also variables (as they might be wrong).
    - Weight gives how much you trust original label.
  - Variants where the unlabeled $\bar{y}^i$ are regularized towards a default value.
    - Can reflect that example is really far from any labeled examples.

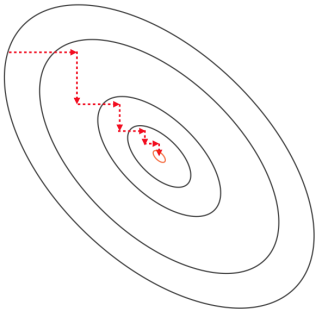
Outline

Beyond Gradient Descent

- For high-dimensional problems we often prefer gradient descent over Newton.
  - Gradient descent requires far more iterations.
  - But iteration cost is only linear in d.
- For very large datasets, even gradient descent iterations can be too slow.
  - If iteration cost is $O(nd)$, we may only be able to do a small number of iterations.
- Two common strategies for yielding even cheaper iterations:
  - Coordinate optimization (today).
  - Stochastic gradient (next time).

Coordinate Optimization

- Each iteration of coordinate optimization only updates on variable:



- For example, on iteration k we select a variable j_k and set

$$w_{j_k}^{k+1} = w_{j_k}^k - \alpha_k \nabla_{j_k} f(w^k),$$

a gradient descent step on coordinate j_k (other w_j stay the same).

  - This variation is called coordinate descent (many variations exist).

Why use Coordinate Descent?

- Theoretically, coordinate descent is a provably bad algorithm:
  - The convergence rate is slower than gradient descent.
  - The iteration cost can be similar to gradient descent.
    - Computing 1 partial derivative may have same cost as computing gradient.
- But it is widely-used in practice:
  - Nothing works better for certain problems.
  - Certain fields think it is the "ultimate" algorithm.
- ??
- Renewed theoretical interest began with a paper by Nesterov in 2010:
  - Showed global convergence rate for randomized coordinate selection.
  - Coordinate descent is faster than gradient descent if iterations are d times cheaper.

Problems Suitable for Coordinate Optimization

- For what functions is coordinate descent d times faster than gradient descent?
- The simplest example is separable functions,

$$f(w) = \sum_{j=1}^{d} f_j(w_j),$$

- Here f is the sum of an f_j applied to each w_j, like $f(w) = \frac{\lambda}{2}\|w\|^2 = \sum_{j=1}^{d} \underbrace{\lambda w_j^2}_{f_j(w_j)}$.
- Cost of gradient descent vs. coordinate descent:
  - Gradient descent costs $O(d)$ to compute each $f'(w_j^k)$.
  - Coordinate descent costs $O(1)$ to compute the *one* $f'_{j_k}(w_{j_k}^k)$.
- In fact, for separable functions you should only use coordinate optimization.
  - The variables w_j have "separate" effects, so can be minimized independently.

Problems Suitable for Coordinate Optimization

- A more interesting example is pairwise-separable functions,

$$f(w) = \sum_{i=1}^{d} \sum_{j=1}^{d} f_{ij}(w_i, w_j),$$

  which depend on a function of each pair of variables.

- An example is label propagation.
  - Also includes any quadratic function.

- Cost of gradient descent vs. coordinate descent:
  - Gradient descent costs $O(d^2)$ to compute each f'_{ij}.
  - Coordinate descent costs $O(d)$ to compute d values of f'_{ij}.

Problems Suitable for Coordinate Optimization

- Our label propagation example looked a bit more like this:

$$f(w) = \sum_{j=1}^{d} f_j(w_j) + \sum_{(i,j)\in E} f_{ij}(w_i, w_j),$$

  where E is a set of (i, j) pairs ("edges" in a graph).

- Adding a separable function doesn't change costs.
  - We could just combine the f_j with one f_{ij}.

- Restricting (i, j) to E makes gradient descent cheaper:
  - Now costs $O(|E|)$ to compute gradient.
  - Coordinate descent could also cost $O(|E|)$ if degree of j_k is $O(|E|)$.

- Coordinate descent is still d times faster in expectation if you randomly pick j_k.
  - Each f'_{ij} is needed with probability $2/d$.
  - So expected cost of $O(|E|/d)$ to compute one partial derivative.

Label Propagation with Coordinate Optimization

- For the binary label propagation objective,

$$\underset{\bar{y}\in\mathbb{R}^t}{\text{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}(y^i - \bar{y}^j)^2 + \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2,$$

  we can exactly optimize one coordinate given the others.

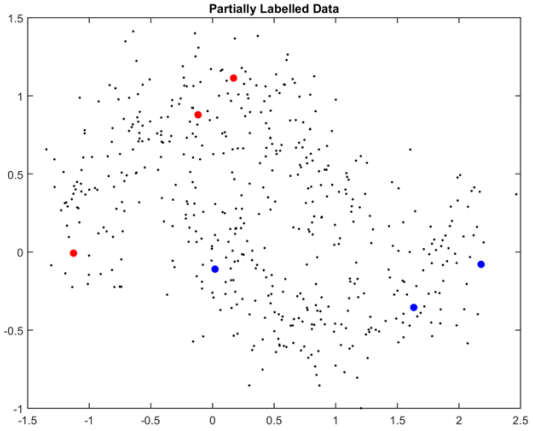
- Taking the derivative and setting it to 0 gives:

$$\bar{y}^i = \frac{\sum_{j=1}^{n} w_{ij}y^j + \sum_{j\neq i} \bar{w}_{ij}\bar{y}^j}{\sum_{j=1}^{n} w_{ij} + \sum_{j\neq i} \bar{w}_{ij}},$$

  where I'm assuming $\bar{w}_{ij} = \bar{w}_{ji}$ (otherwise, you replace both by their average).
- So coordinate optimization takes weighted average of neighbours.

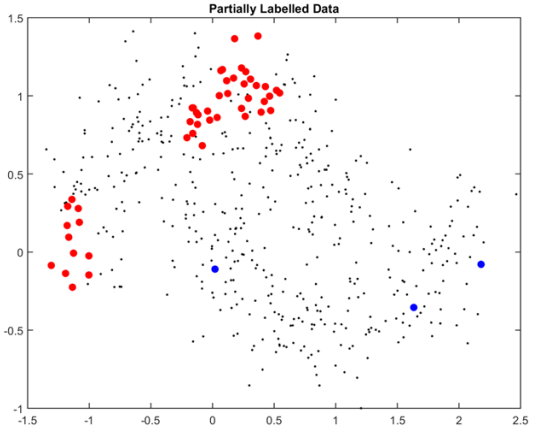
Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



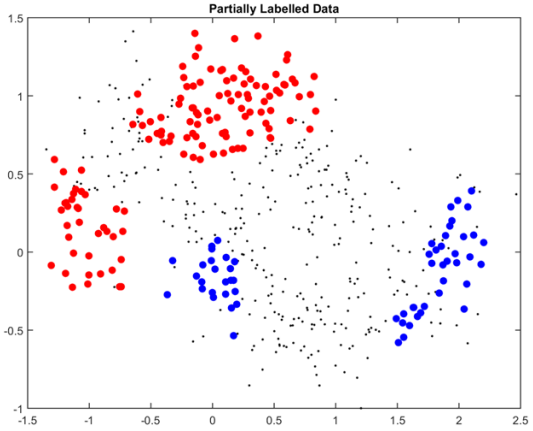
Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



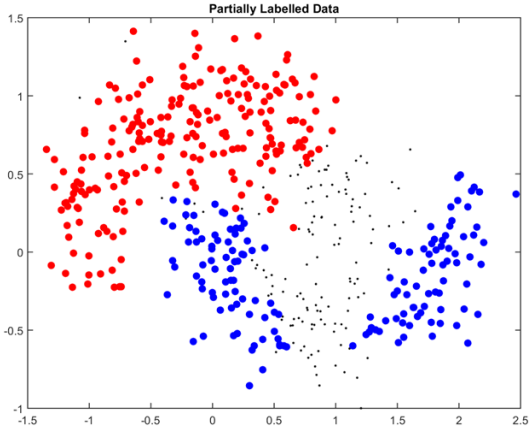
Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



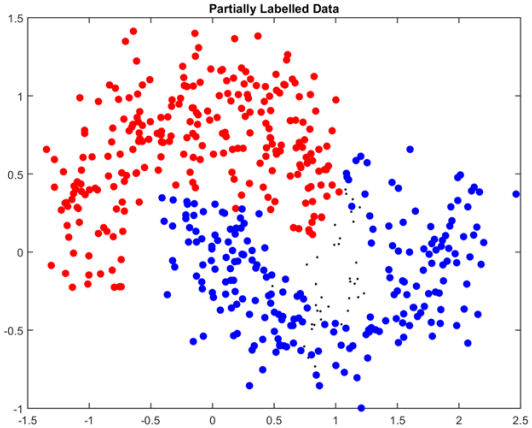
Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



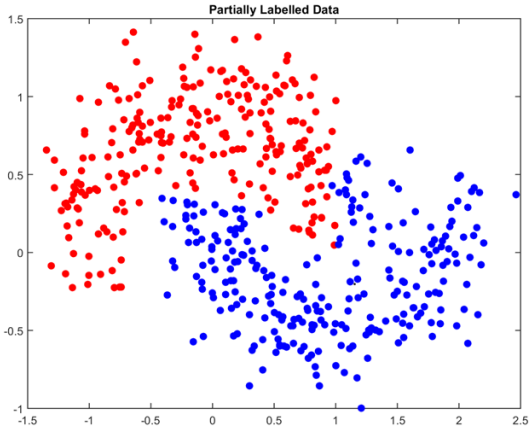
Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



Label Propagation with Coordinate Optimization

- Label propagation with coordinate optimization (rounding to nearest label):



Summary

- Inexact proximal-gradient can be used for structured sparsity.
- Transductive learning:
  - Given labeled and unlabeled examples, label the unlabeled examples.
- Label propagation:
  - Transductive learning method minimizing variation in the label space.
- Coordinate optimization: updating one variable at a time.
  - Efficient if updates are d-times cheaper than gradient descent.

- Next time: the most important algorithm in machine learning.