Optimization Models in Machine Learning: Introduction and Examples

University of Washington

July 30, 2020

Outline

1. Intro to Optimization + Notebook 1
2. Linear Regression and Regularization + Notebook 2
3. Logistic Regression + Notebook 3
4. Outlier Removal + Notebook 4

Optimization: Overview

A general optimization problem has the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{array}$$

with components

- $x = (x_1, \ldots, x_n)$ - optimization variable
- $f_0 : \mathbf{R}^n \to \mathbf{R}$ - objective function
- $f_i : \mathbf{R}^n \to \mathbf{R}$ - constraint functions; b_i - constraint bounds

Optimization: Overview

A general optimization problem has the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{array}$$

with components

- $x = (x_1, \ldots, x_n)$ - optimization variable
- $f_0 : \mathbf{R}^n \to \mathbf{R}$ - objective function
- $f_i : \mathbf{R}^n \to \mathbf{R}$ - constraint functions; b_i - constraint bounds

Many applications:

- Data fitting and regression
- Classification
- Image processing
- Portfolio optimization
- Recommender systems
- Optimal control
- Medical treatment planning

Optimization: Problem Classes

There are different classes of optimization problems, which can determine a problem's difficulty and solution method:

- Constrained vs. Unconstrained
- Smooth vs. Nonsmooth
- Convex vs. Nonconvex

[1] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

Optimization: Problem Classes

There are different classes of optimization problems, which can determine a problem's difficulty and solution method:

- Constrained vs. Unconstrained
- Smooth vs. Nonsmooth
- Convex vs. Nonconvex

An important class: **convex optimization** problems

> "With only a bit of exaggeration, we can say that if you formulate a practical problem as a convex optimization problem, then you have solved the original problem."[1]

[1] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

Optimization: Problem Classes

A convex optimization problem has objective and constraint functions that satisfy the inequality

$$f_i\Big(\lambda x + (1-\lambda)y\Big) \leq \lambda f_i(x) + (1-\lambda) f_i(y)$$

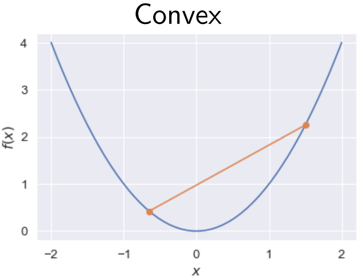
for all $x, y \in \mathbf{R}^n$ and all $0 \leq \lambda \leq 1$.

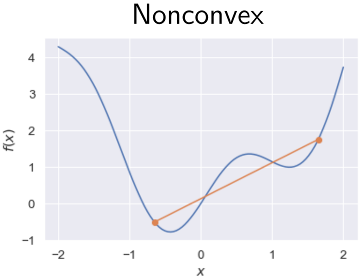
Optimization: Problem Classes

A convex optimization problem has objective and constraint functions that satisfy the inequality

$$f_i\Big(\lambda x + (1-\lambda)y\Big) \leq \lambda f_i(x) + (1-\lambda) f_i(y)$$

for all $x, y \in \mathbf{R}^n$ and all $0 \leq \lambda \leq 1$.





Important consequence: in a convex problem, no "local minima"

Optimization: Solution Methods

Very few optimization problems have a closed-form solution (e.g., least-squares); most problems are solved using iterative methods.

One important iterative method is gradient descent:

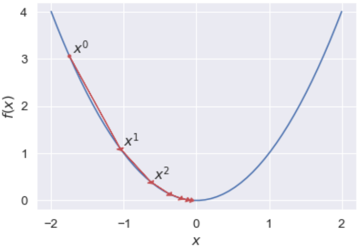
$$x^{k+1} = x^k - \alpha \nabla f\left(x^k\right)$$

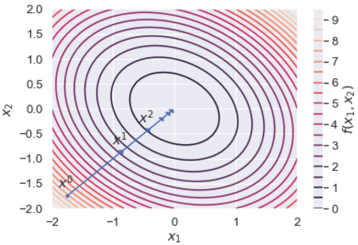
Optimization: Solution Methods

Very few optimization problems have a closed-form solution (e.g., least-squares); most problems are solved using iterative methods.

One important iterative method is gradient descent:

$$x^{k+1} = x^k - \alpha \nabla f\left(x^k\right)$$





Optimization: Machine Learning Models

Many problems in machine learning seek to build a model

$$g(a; x) \approx y$$

given a data set

$$\Big\{(a_1, y_1), \ldots, (a_m, y_m)\Big\},$$

with components

- $a_i = (a_{i1}, \ldots, a_{in})$ - data features
- $y_i \in \mathbf{R}$ or $\{0, 1\}$ - data value or label/class
- $g : \mathbf{R}^n \to \mathbf{R}$ or $\{0, 1\}$ - prediction function
- $x = (x_1, \ldots, x_n)$ - model parameters
- m - number of data points
- n - number of data features

Optimization: Machine Learning Models

We can fit a model to the given data by solving an optimization problem of the form

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{m} f_i\Big(g(a_i; x), y_i\Big) + r(x),$$

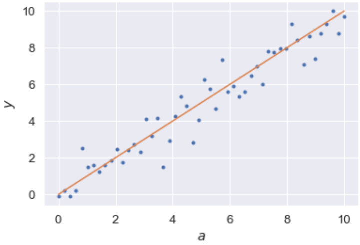
with components

- $x = (x_1, \ldots, x_n)$ - model parameters we want to learn
- $f_i : \mathbf{R}^n \to \mathbf{R}$ - "loss" functions: measure how well the model fits the data for given parameters; e.g., $(g(a_i; x) - y_i)^2$
- $r(x) : \mathbf{R}^n \to \mathbf{R}$ - regularization function

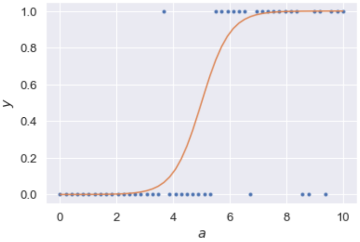
Optimization: Machine Learning Models

We focus on two common problems in machine learning:

Linear Regression



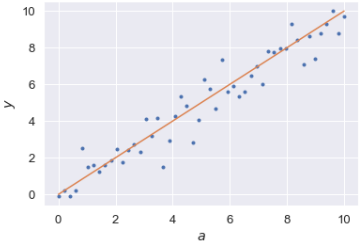
Logistic Regression (Classification)



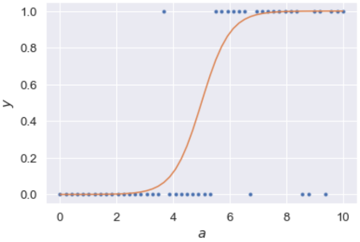
Optimization: Machine Learning Models

We focus on two common problems in machine learning:

Linear Regression



Logistic Regression (Classification)



- Data: Continuous features $\{a_i\}$ and outputs $\{y_i\}$
- Goal: Find linear predictor

$$x_0 + x_1 a_i \approx y_i$$

- Approach: Assume a statistical model for errors and develop a maximum likelihood formulation

Linear Regression: Derivation

Assuming the errors in our data come from a normal distribution,

$$y_i = x_0 + x_1 a_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \quad \text{independent},$$

Linear Regression: Derivation

Assuming the errors in our data come from a normal distribution,

$$y_i = x_0 + x_1 a_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \quad \text{independent},$$

the probability of observation (a_i, y_i) given the parameters is

$$P\Big((a_i, y_i); x_0, x_1\Big) \propto \exp\left(\frac{-(y_i - x_0 - x_1 a_i)^2}{2\sigma^2}\right).$$

Linear Regression: Derivation

Assuming the errors in our data come from a normal distribution,

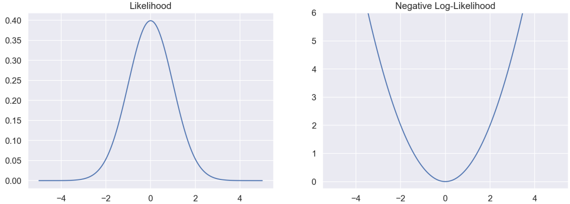
$$y_i = x_0 + x_1 a_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2) \quad \text{independent},$$

the probability of observation (a_i, y_i) given the parameters is

$$P\Big((a_i, y_i); x_0, x_1\Big) \propto \exp\left(\frac{-(y_i - x_0 - x_1 a_i)^2}{2\sigma^2}\right).$$

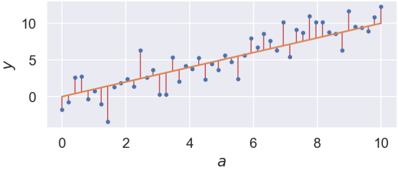
We can fit model parameters by maximizing the likelihood (minimizing the negative log-likelihood):

$$-\log \prod_{i=1}^{m} \exp\left(-\frac{(y_i - x_0 - x_1 a_i)^2}{2\sigma^2}\right) \propto \sum_{i=1}^{m} (y_i - x_0 - x_1 a_i)^2$$



Linear Regression: Intuition and Properties

$$\min_{x_0, x_1} \sum_{i=1}^{m} (y_i - x_0 - x_1 a_i)^2$$



- Minimize the least-squares distance between observations y_i and predictions $x_0 + x_1 a_i$.
- The problem is convex, smooth, and easy to solve.
- Linear regression actually has a closed-form solution, but it is often found more efficiently by iterative algorithms

Regularization: Overview

Many problems in machine learning add a regularization term $r(x)$ to the objective function to

- incorporate prior knowledge about *structure* in x, e.g., sparsity or smoothness
- help avoid overfitting,
- get more robust (to data perturbations) solutions, or
- improve the stability of the solution process.

Two popular forms of regularized linear regression:

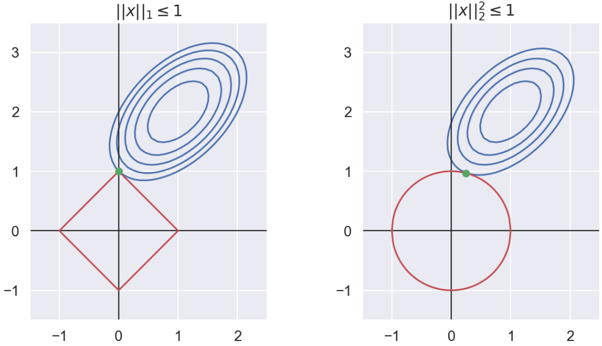
- Lasso - $\min_x f(x) + \lambda\|x\|_1$, where $\|x\|_1 = \sum_{i=1}^n |x_i|$
- Ridge - $\min_x f(x) + \lambda\|x\|_2^2$, where $\|x\|_2^2 = \sum_{i=1}^n x_i^2$

Regularization: Geometric Interpretation

Consider the constrained least-squares problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|Ax - y\|_2^2 \\ \text{subject to} & \|x\|_p \leq t. \end{array}$$

Choice of norm influences properties of solution x: with $p = 1$, solutions tend to occur on the vertices, where many $x_i = 0$.



Regularization: Relaxed Constraints

We can move the norm from a constraint into the objective function to get

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_p,$$

where regularization parameter λ balances model error with how much we regularize.

The Lasso ($p = 1$) is often used to find sparse solutions. Ridge regression ($p = 2$) is often used for ill-conditioned problems.

More generally: regularizers can promote other structures:
For example, if the parameters form a matrix X, a low-rank matrix is often desired (e.g., the 'matrix completion problem' for recommender systems).

Logistic Regression: Overview

- Data: Continuous features $\{a_i\}$ and discrete labels $y_i \in \{0, 1\}$

- Goal: Find linear predictor

$$x_0 + x_1 a_i = \begin{cases} \text{positive} & \Rightarrow \quad y_i = 1 \\ \text{negative} & \Rightarrow \quad y_i = 0 \end{cases}$$

- Approach: Combine Bernoulli model with a linear predictor

- Examples: Hours studied vs. Pass/Fail, measurements vs. disease

Logistic Regression: Derivation

Rewriting the Bernoulli model in standard form,

$$P\Big((a_i, y_i); p_i\Big) = p_i^{y_i}(1-p_i)^{1-y_i}$$
$$= \exp\left(y_i \log\left(\frac{p_i}{1-p_i}\right) + \log(1-p_i)\right),$$

we can model the term multiplying y_i using our linear predictor,

$$\log\left(\frac{p_i}{1-p_i}\right) = x_0 + x_1 a_i,$$

which gives us,

$$\log(1-p_i) = -\log\left(1 + \exp(x_0 + x_1 a_i)\right).$$

Combining the above expressions results in the likelihood function

$$\mathcal{L}\Big(x_0, x_1; (a, y)\Big) = \prod_{i=1}^{m} \exp\Big(y_i(x_0+x_1 a_i) - \log\left(1+\exp(x_0+x_1 a_i)\right)\Big).$$

Logistic Regression: Derivation

We can fit our model parameters to the given data by maximizing the likelihood, or by minimizing the negative log-likelihood:

$$-\log \mathcal{L}\Big(x_0, x_1; (a, y)\Big) = \sum_{i=1}^{m} \log\left(1 + \exp(x_0 + x_1 a_i)\right) - y_i(x_0 + x_1 a_i)$$

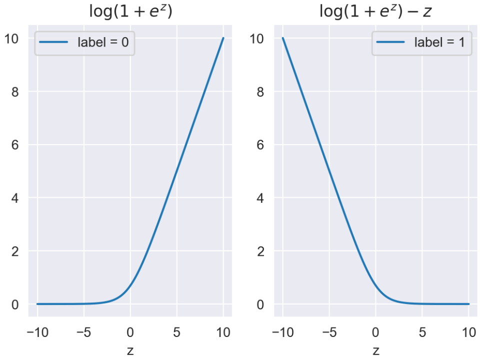
Explicitly, we solve the following problem

$$\min_{x_0, x_1} \sum_{i=1}^{m} \log(1 + \exp(x_0 + x_1 a_i)) - y_i(x_0 + x_1 a_i)$$

Logistic Regression: Intuition and Properties

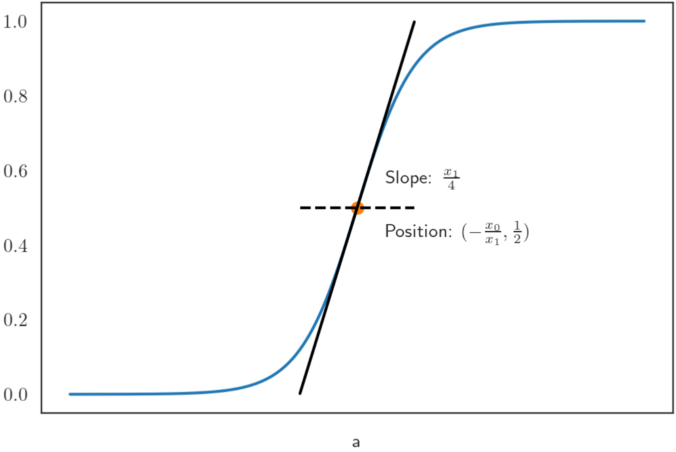
$$\min_{x_0, x_1} \sum_{i=1}^{m} \log(1 + \exp(x_0 + x_1 a_i)) - y_i(x_0 + x_1 a_i)$$

- If the label is 0, we want to make $\log(1 + \exp(x_0 + x_1 a_i))$ as small as possible, equivalent to making $x_0 + x_1 a_i \ll 0$
- If the label is 1, can show objective decreases with respect to $x_0 + x_1 a_i$, so we want $x_0 + x_1 a_i \gg 0$



Logistic Regression: Intuition and Properties

- We look for intercept x_0 and slope x_1 that do the best job for all the data in the set.



Logistic Regression: Intuition and Properties

- The problem is convex and smooth, and 'nice' to solve.
- For a future data points with feature a, $p = \frac{\exp(x_0 + x_1 a)}{1 + \exp(x_0 + x_1 a)}$
- Other methods can also be used, e.g. support vector machines.

Trimming: Outlier Removal

There are many classic ways to remove outliers:

Trimming: Outlier Removal

There are many classic ways to remove outliers:

- Fit, remove outliers, refit
  - Upside: easy to do
  - Downside: outliers can affect the initial fit
  - Downside: when to stop?

Trimming: Outlier Removal

There are many classic ways to remove outliers:

- Fit, remove outliers, refit
  - Upside: easy to do
  - Downside: outliers can affect the initial fit
  - Downside: when to stop?
- Regression: replace least squares loss with 'robust' penalty
  - Upside: relatively easy to do
  - Downside: how to pick shape?
  - Downside: how to extend to non-additive errors?

Trimming: Outlier Removal

There are many classic ways to remove outliers:

- Fit, remove outliers, refit
  - Upside: easy to do
  - Downside: outliers can affect the initial fit
  - Downside: when to stop?
- Regression: replace least squares loss with ‘robust’ penalty
  - Upside: relatively easy to do
  - Downside: how to pick shape?
  - Downside: how to extend to non-additive errors?
- Our focus: trimming
  - Upside: works for any model
  - Upside: transparent assumptions
  - Downside: nonconvex model
  - Upside: doesn’t seem to matter in practice

Trimming: Overview

Trimming uses auxiliary weights to detect outliers:

$$\min_{x,w} \sum_{i=1}^{m} w_i f_i(x) \quad \text{s.t.} \quad w_i \in [0,1], \quad \sum_{i=1}^{m} w_i = h$$

- For fixed x, minimal h residuals have $w_i = 1$, rest are 0
- Minimal h residuals are thus classified as ‘inliers’
- Remaining $m - h$ points are by default 'outliers'
- As x varies, we are looking to only fit inliers.

Problem is theoretically hard, but practically works very well.

Trimming: Overview

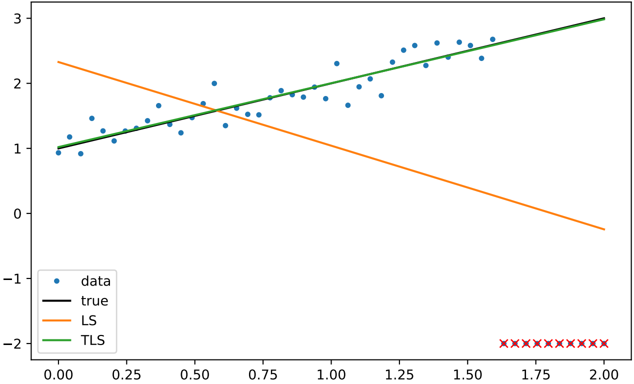
The general idea extends to any learning model:

$$\min_{x,w} \sum_{i=1}^{m} w_i f_i(x) \quad \text{s.t.} \quad w_i \in [0,1], \quad \sum_{i=1}^{m} w_i = h$$

- Least squares: $f_i(x) = \frac{1}{2}(y_i - x_0 - x_1 a_i)^2$
- Logistic: $f_i(x) = \log\big(1 + \exp(x_0 + x_1 a_i)\big) - y_i(x_0 + x_1 a_i)$
- Neural net: $f_i(x) =$ soft max for a labeled data point

Trimming: Least Squares Example

$$\min_{x,w} \sum_{i=1}^{m} \frac{w_i}{2}(y_i - x_0 - x_1 a_i)^2 \quad \text{s.t.} \quad w_i \in [0,1], \quad \sum_{i=1}^{m} w_i = h$$



Trimming: CNN Example

Here we see the results of a convolutional neural network (CNN) classifier that predicts cats and birds, with inliers (top row) and outliers (bottom row) identified using trimming.