OPTIMIZATION METHODS FOR MACHINE LEARNING

BETHANY LUSCH
Asst. Computer Scientist
Argonne National Lab
Leadership Computing Facility
blusch@anl.gov

August 9, 2019
ATPESC
WHAT IS OPTIMIZATION?

**minimize** \[ f(x) \]

\[ \text{subject to } \ldots \text{constraints} \ldots \]

“objective” or “loss” function

- “best route” (Minimize cost of delivery subject to all mail is delivered)
- “best product” (Minimize -profit subject to safety)
- “best prediction model” (Minimize prediction error)
MACHINE LEARNING

minimize \quad f(x)

subject to \quad \ldots \text{constraints}\ldots

• “best prediction model” (Minimize prediction error)
• “best recommendation” (Minimize # who don’t buy anything)
• “best clusters” (Minimize distance within clusters while maximizing distance between clusters)

Underneath most ML problems is an optimization problem
TYPES OF OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2^{nd} derivs
- Have gradients
- General

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \quad i = 1, \ldots, m.
\end{align*}
\]
TYPES OF OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Roughly…

- More time on formulating problem to fit these categories
- More time on optimization algorithm

Desperation
LINEAR PROGRAMMING

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

\[ \begin{aligned}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \quad i = 1, \ldots, m. \\
\end{aligned} \]

Linear
\[ c^T x \]

Convex polytope
\[ Ax \leq b \\
x \geq 0. \]

Mature polynomial-time algorithms
Local minima are global optima

Picture source: Ylloh at wikipedia.org/wiki/Linear_programming
QUADRATIC PROGRAMMING

- Linear
- Quadratic
- Convex
- Have 2nd derivs
- Have gradients
- General

Convex polytope

\[ \frac{1}{2} x^T Q x + c^T x \]

minimize \( f_0(x) \)

subject to \( f_i(x) \leq b_i, \ i = 1, \ldots, m. \)

If \( Q \) is positive definite:
Weakly polynomial-time algorithms
Local minima are global optima

Desperation

OPTIMIZATION - McCormick

Picture source: Jph425 at optimization.mccormick.northwestern.edu
CONVEX OPTIMIZATION

- Linear
- Quadratic
- Convex
  - Have 2nd derivs
  - Have gradients
  - General

Convex function
Convex set

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \ i = 1, \ldots, m.
\end{align*}
\]

In some cases: poly-time
Local minima are global optima
Reliable methods
DIFFERENTIABLE OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Differentiable function

\[ \min_{x} f_0(x) \]
subject to \[ f_i(x) \leq b_i, \ i = 1, \ldots, m. \]

Generally NP-hard
Local minima problematic
Can use gradients and ideally Hessians in algorithm

Picture source: Robert Johansson in “Optimization”
GENERAL OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \ i = 1, \ldots, m.
\end{align*}
\]

Generally NP-hard
Hopefully know \textit{some} structure!
DISCRETE OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2nd derivs
- Have gradients
- General

continuous optimization

discrete optimization

Picture source: wallpaperflare.com
CLASSIFICATION EXAMPLE

- Problem: label each document \( x \) as related to politics or not (1 or -1).
- Hard to come up with rules by hand, so ML helps: learn function \( h(x) \)
- Really want to minimize expected risk of misclassification:
  \[
  \minimize_h R(h) = \mathbb{P}[h(x) \neq y] = \mathbb{E}[\mathbb{1}[h(x) \neq y]]
  \]

- How do we pick family of functions to optimize over?
- How do we know which one is optimal?
REALITIES

- Really want to minimize expected risk of misclassification:

$$\min_h R(h) = \mathbb{P}[h(x) \neq y] = \mathbb{E} [\mathbbm{1}[h(x) \neq y]]$$

- Don’t know probability distribution, so minimize empirical risk:

$$\min_h R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq y_i]$$

- Easier if smooth loss and parameterized $h$:

$$\min_w R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i)$$
CHOOSING FUNCTION FAMILY

- Possibility of low empirical risk on training data
- Expected risk and empirical risk don’t have large gap
- Can efficiently solve optimization
- (convenient representation, smoothness, …)

Major themes of machine learning!
BIAS VS. VARIANCE

High variance
- **overfitting**

High bias
- **underfitting**

Low bias, low variance
- **balanced**

Picture similar to: Seema Singh, "Understanding the Bias-Variance Tradeoff"
LINEAR REGRESSION (LEAST-SQUARES)

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

\[
\begin{align*}
\text{minimize} & \quad \|Ax - b\|^2_2 \\
\text{subject to} & \quad x \in \mathbb{R}^n
\end{align*}
\]

if multiply out:

\[
\begin{align*}
\text{minimize} & \quad x^T A^T A x - 2b^T A x + b^T b \\
\end{align*}
\]

... quadratic program!
SUPPORT VECTOR MACHINE

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Find linear classifier with maximum margin

\[
\begin{align*}
\text{minimize} & \quad \|a\|_2 + \gamma (1^T u + 1^T v) \\
\text{subject to} & \quad a^T x_i - b \geq 1 - u_i, \quad i = 1, \ldots, N \\
& \quad a^T y_i - b \leq -(1 - v_i), \quad i = 1, \ldots, M \\
& \quad u \geq 0, v \geq 0
\end{align*}
\]

... quadratic program!

SUPPORT VECTOR MACHINE

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Find linear classifier with maximum margin

Kernel SVM can do nonlinear classification while remaining a quadratic program

K-MEANS CLUSTERING

- Linear
- Quadratic
- Convex
- Have 2nd derivs
- Have gradients
- General

Find clustering that minimizes distances within clusters

\[
\text{minimize } \sum_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu \|^2_2
\]

Desperation

NP-hard discrete problem, so use approx. algorithm

DEEP LEARNING

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Getting desperate, plus want to be scale well, so use gradient descent

$$\min_{\theta} \| x - f(x) \|^2_2$$
RECALL: TYPES OF OPTIMIZATION

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Roughly…

- More time on formulating problem to fit these categories
- More time on optimization algorithm

Desperation
ANALOGOUSLY...

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Roughly...

- More time on formulating problem (choosing features) so that these (biased) methods are suitable
- More time on optimization algorithm
REMINDER

- Possibility of low empirical risk on training data
- Expected risk and empirical risk don’t have large gap
- Can efficiently solve optimization

- Big neural networks can be very expressive (low bias)
- So don’t need to be as clever about input features
- But then easy to overfit…
- Optimization is tricky: optimization stalls, plus local minima or saddle points

Bias vs. variance
Overfitting vs. underfitting
GRADIENT DESCENT

\[
\begin{align*}
\text{minimize } & \quad f(w) \\
w_k+1 & \leftarrow w_k - \alpha_k \nabla f(w_k)
\end{align*}
\]

Picture source: Divakar Kapil in “Stochastic vs Batch Gradient Descent”
TYPES OF GRADIENT DESCENT

i.e. for empirical risk, explicitly summing over data points

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

$$w_{k+1} \leftarrow w_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_{i,k}(w_k)$$

Batch GD: use all points every step

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla f_{i,k}(w_k)$$

Stochastic GD: use one point per step

$$w_{k+1} \leftarrow w_k - \frac{\alpha_k}{|S_k|} \sum_{i \in S_k} \nabla f_{i,k}(w_k)$$

Mini-batch GD: use a subset each step
TYPES OF GRADIENT DESCENT

Batch GD: use all points every step  
Each step is accurate but expensive

Stochastic GD: use one point per step  
Each step is noisy but fast

Mini-batch GD: use a subset each step  
Happy medium?

Very common in deep learning, but often call it SGD
GRADIENT DESCENT CONSIDERATIONS

\[ \text{minimize} \quad f(w) \]

\[ w_{k+1} \leftarrow w_k - \alpha_k \nabla f(w_k) \]

• Step size \( \alpha_k \):
  • Too big: overshoot
  • Too small: very slow
  • (But can be good to escape local minima)

• Initialization
• Can you make the problem easier?

Li, et al. “Visualizing the Loss Landscape of Neural Nets” NeurIPS 2018
VARIANT: ADAM

Popular improvement on GD: Adam optimizer
• Separate learning rate for each weight
• Momentum: uses moving average of the gradient
• Also incorporates squared gradients

Cool exploration/visualization of momentum: https://distill.pub/2017/momentum/

(For those familiar: combines the best properties of AdaGrad, momentum, and RMSProp)
REGULARIZATION

- Common way to avoid overfitting: regularization
- Most common: L2 regularization

\[
\text{minimize } \frac{1}{n} \sum_{i=1}^{n} (h(x_i; w) - y_i)^2 + \lambda \|w\|^2_2
\]

Roughly: big coefficients/weights correspond to large variation
OVERFITTING CAUTION

• Can you generalize outside of your training set to validation/testing set?
• What about interpolating to data you haven’t collected?
• Extrapolation extra unlikely to work
SUMMARY

- Linear
- Quadratic
- Convex
- Have 2\textsuperscript{nd} derivs
- Have gradients
- General

Major themes in machine learning:
- Overfitting vs. underfitting
- Ability to efficiently solve optimization problem

For more, see:

Optimization Methods for Large-Scale Machine Learning*

Léon Bottou†
Frank E. Curtis†
Jorge Nocedal§
ANY QUESTIONS?

Thinking ahead to next talk: how would you parallelize gradient descent?