Mathematics of Machine Learning

Optimization strategies for supervised learning


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

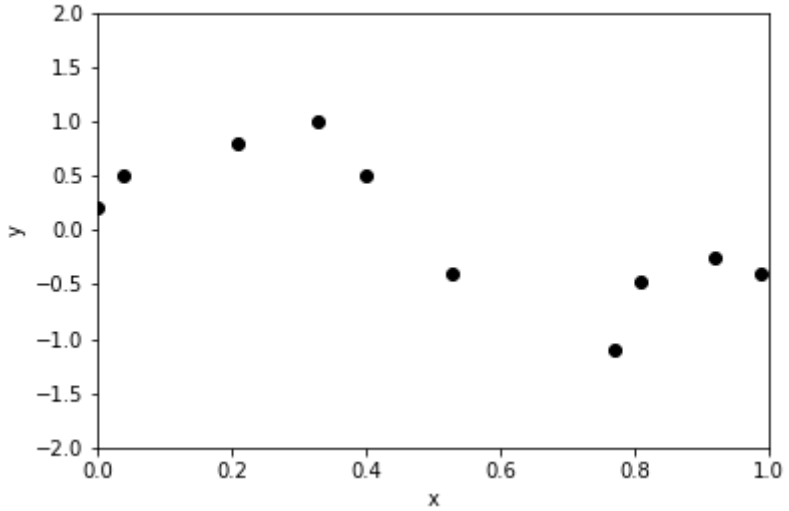
Overview

- From learning to Optimization
- Introduction to Optimization
- Current trends

Supervised learning

Given $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ find a function f such that $f(x^{(i)}) \approx y^{(i)}$

Fit a line through a set of points



Fit a line through a set of points

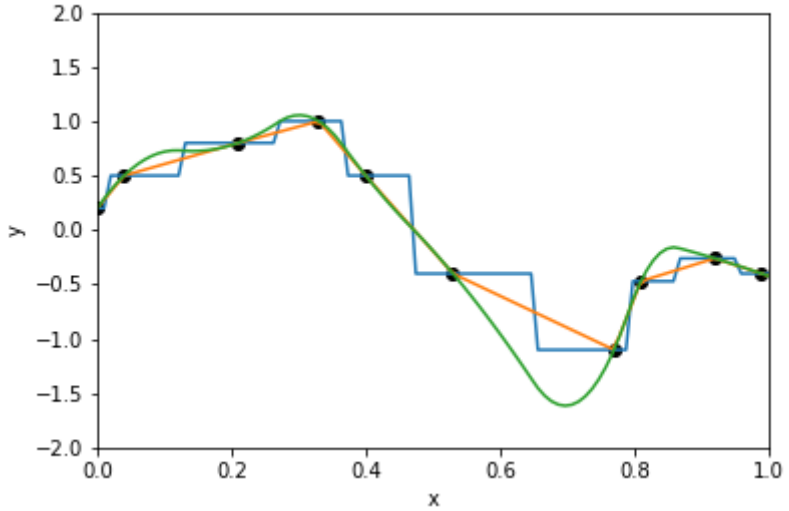


Figure out which papays are yummy

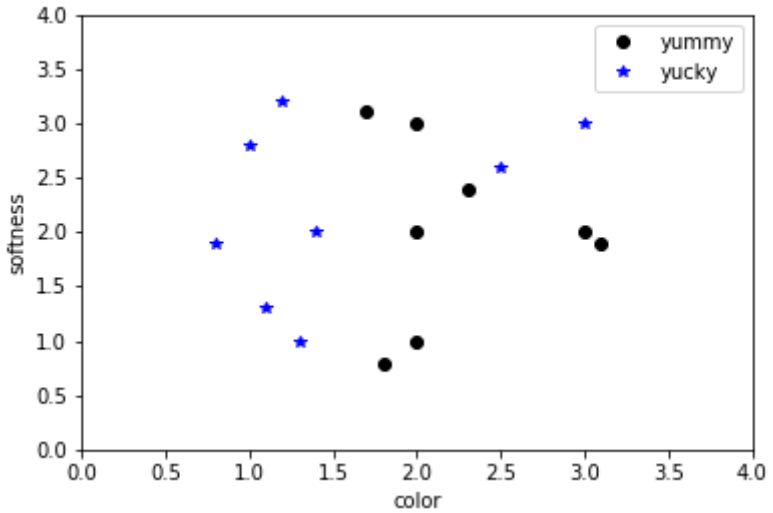
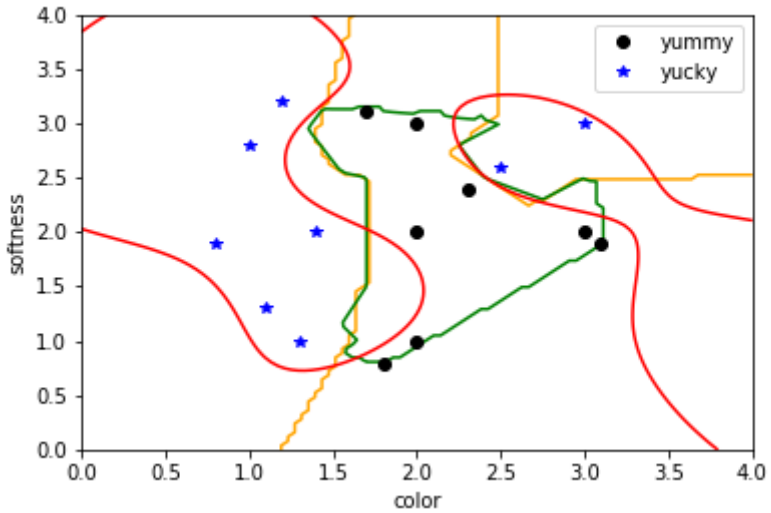


Figure out which papaya's are yummy



The supervised learning zoo

- Regression
- Neural networks
- Support Vector Machines
- ....

Linear regression

$$f(x) = w_0 + \sum_{k=1}^{n} w_i x_i,$$

where the parameters are determined by solving

$$\min_{w} \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

Neural networks

$$f(x) = f_n \circ f_{n-1} \circ \ldots \circ f_1(x),$$

with

$$f_k(x) = \sigma(W_k x + b_k).$$

- The architecture is determined by the structure of the matrices W_k.
- The optimal weights are found by *training*.

Support vector machines

$$f(x) = \sum_{i=1}^{m} w_i k(x_i, x),$$

- The kernel $k(x, x')$ determines the properties of the function
- The weights are determined by solving

  $$\min_{w} w^T K w + L(w),$$

  where $k_{ij} = k(x_i, x_j)$.

Optimization

Find a minimizer of a *cost function*

$$C(w) = \sum_{i=1}^{m} L(f_i(w), y_i) + R(w),$$

where

- $w \in \mathbb{R}^n$ parametrizes the function
- $f_i(w)$ is the predicted label for x_i
- $L(f(x), y)$ is the *loss function*, and
- $R(w)$ is the *regularization term*.

Example: linear regression

Let

- $f_i(w) = w_0 + \sum_{k=1}^{n} w_i x_i$
- $L(f_i, y_i) = (f_i - y_i)^2$
- $R(w) =\tilde{} \ \|w\|_2^2$

This leads to a least-squares problem

$$C(w) = \|Xw - y\|_2^2 +\tilde{} \ \|w\|_2^2,$$

with a closed-form solution

$$\widehat{w} = \left(X^T X +\tilde{} \ I\right)^{-1} X^T y.$$

Optimization

The optimization problem

$$\min_{w} \sum_{i=1}^{m} L(f_i(w), y_i) + R(w),$$

- Often does not have a closed-form solution
- May not have a *unique* minimizer
- Has too many variables to allow for sampling-based methods
- Does have a very particular *structure*

Furthermore:

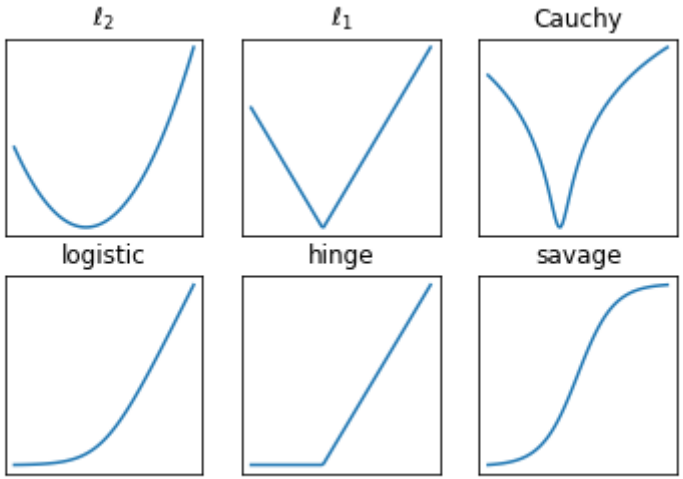
- Evaluation of the cost function may be computationally expensive
- Evaluating the gradient may be difficult
- A good initial guess of the parameters may not be available

Optimization

- Structure of the cost function
- Optimality conditions
- Basic iterative algorithms

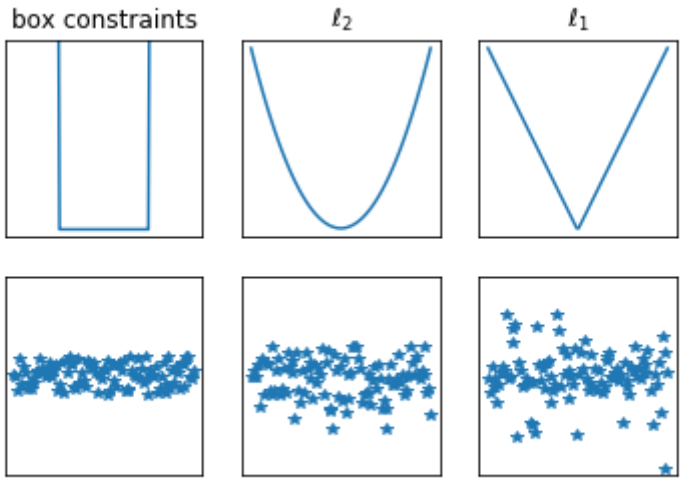
Loss functions

Measure the difference between the prediction f_i and the label y_i:



Regularizers

Promote certain regularity of the coefficients:



Defining properties

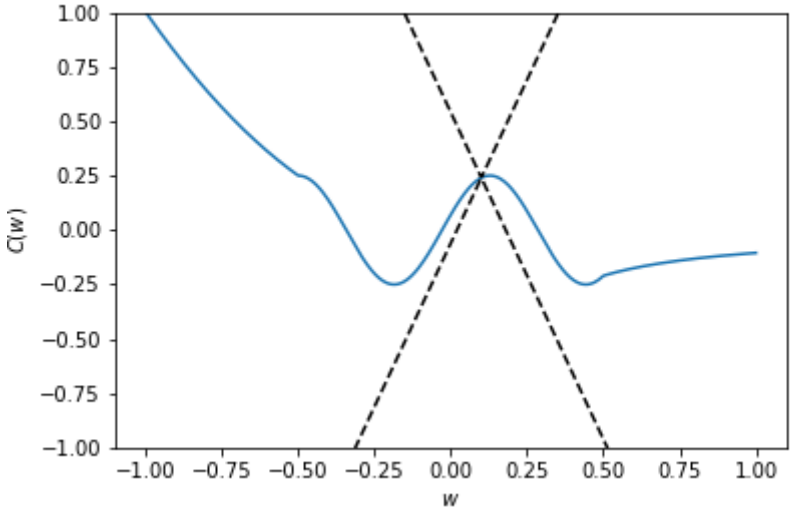
Three properties come up frequently in optimization:

- Continuity
- Convexity
- Smoothness

Defining properties - Lipschitz continuity

A function $C : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant λ if $\forall\, w, w' \in \mathbb{R}^n$

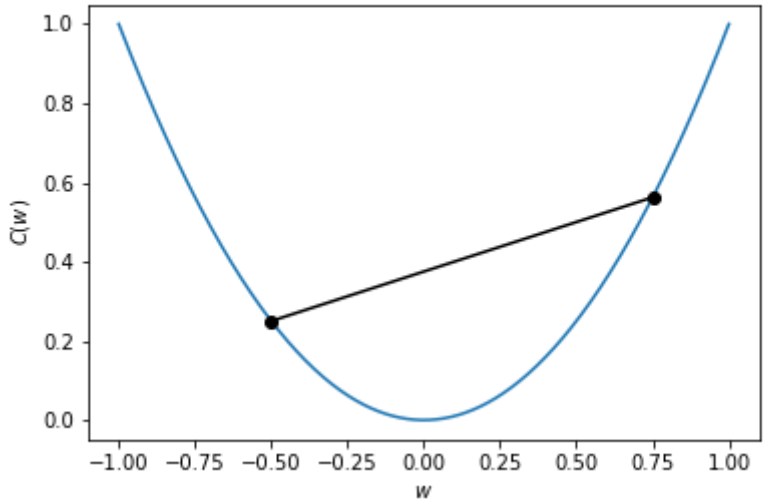
$$|C(w) - C(w')| \leq \lambda \|w - w'\|_2.$$



Definining property - convexity

A function $C : \mathbb{R}^n \to \mathbb{R}$ is convex if $\forall w, w' \in \mathbb{R}^n$ and $\forall t \in [0, 1]$ we have

$$C(tw + (1 - t)w') \leq tC(w) + (1 - t)C(w').$$



Defining property - smoothness

A function $C : \mathbb{R}^n \to \mathbb{R}$ is k-smooth if all partial derivatives or order $\leq k$ exist and are continuous.

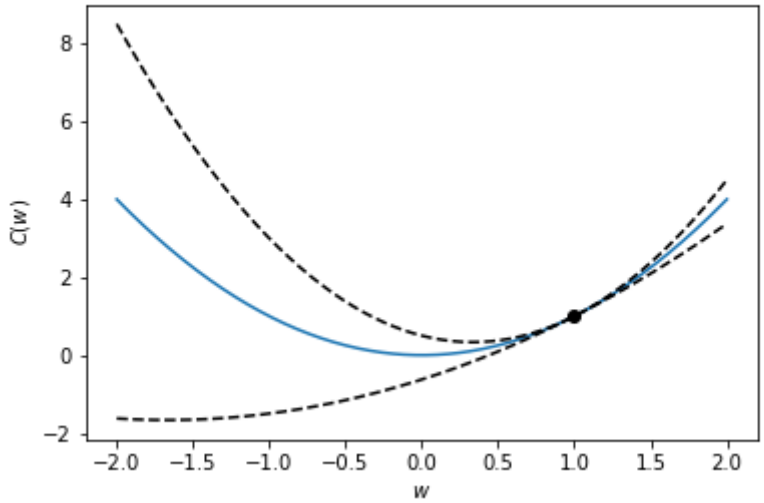
For $k \geq 1$:

- The Lipschitz constant can be found in terms of the gradient $\max_w \|\nabla C(w)\|$
- The Lipschitz constant of the gradient, $\varkappa$, is of interest.
- The function is γ*-strongly convex* if
  $C(w) \geq C(w') + \langle \nabla C(w'), (w - w') \rangle + \frac{\gamma}{2} \|w - w'\|_2^2$.
- The constant $\varkappa / \gamma$ is called the *condition number* of C.

For $k \geq 2$:

- The Lipschitz constant of the *gradient* is the largest eigenvalue of the Hessian
- If the Hessian is positive semidefinite, the function is convex
- If the Hessian is positive definite, the function is *strongly* convex (with constant γ the smallest eigenvalue of the Hessian)

Defining property - strong convexity



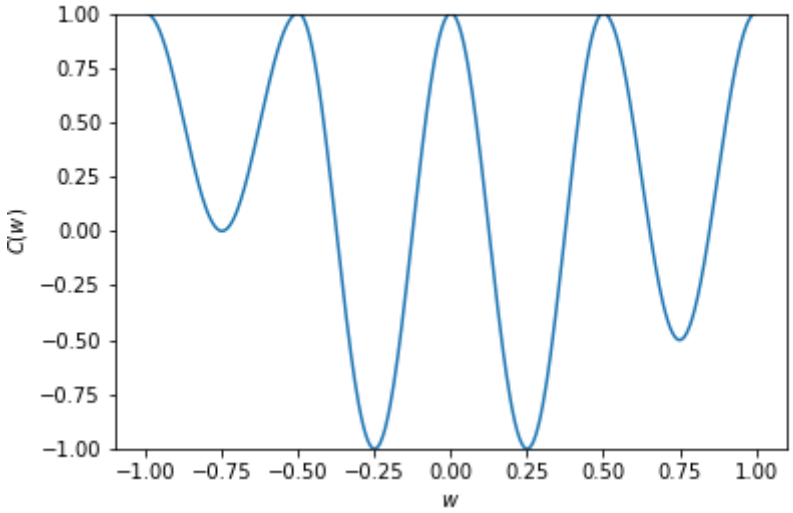
Defining properties

Note:

- It is often not possible to find the (global) bounds $(\varkappa, \quad)$ a-priori
- The objective may not be globally convex/strongly convex, but is usefull to think of these as local properties that hold close to a minimizer

Local and global minima

- For convex functions, a local minimum is also a global minimum
- For strongly convex functions, the minimum is unique



Optimality conditions (for local minima)

Smooth function:

- Gradient is zero, Hessian is positive (semi-)definite

Convex function:

- Subgradient contains zero

Algorithms

Basic template (for smooth cost functions)

$$w_{k+1} = w_k - \grave{\alpha}_k d_k,$$

where $\grave{\alpha}_k$ is the *stepsize* and the *search direction* is constructed from the gradient:

- $d_k = \nabla C(w_k)$ (steepest descent)
- $d_k = \nabla C(w_k) + \tilde{\beta}_k d_{k-1}$ (conjugate gradients)
- $d_k = B_k \nabla C(w_k)$ (quasi-Newton)
- $d_k = \left(\nabla^2 C(w_k)\right)^{-1} \nabla C(w_k)$ (Newton's method)

Note that the search direction needs to be descent direction: $\langle \nabla C(w_k), d_k \rangle < 0$.

Algorithms - steepest descent

Steepest descent with a fixed stepsize:

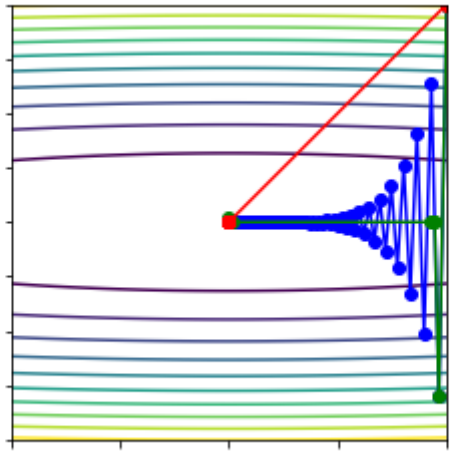
- The iterates converge to a *stationary point* from *any* initial point (with suitable choice of α_k)
- For convex functions with Lipschitz-continuous gradient, the iterates converge to a minimum at a *sub-linear* rate of $\mathcal{O}(1/k)$ (with $\alpha_n \in (0, 2/L)$)
- For strongly convex functions, the iterates converge to the minimum at a *linear* rate $\mathcal{O}(\rho^k)$ (with $\alpha_n \in (0, 2/L)$)

To reach a point w_k for which $|C(w_k) - C(w_*)| \leq \epsilon$ we need

- $\mathcal{O}(\epsilon^{-1})$ iterations with a sub-linear rate
- $\mathcal{O}(\log \epsilon^{-1})$ iterations with a linear rate

Algorithms - comparison

- **Steepest descent:** at best *linear convergence* to stationary point from *any* initial point
- **Conjugate gradient / Quasi-newton:** Faster convergence in practice (*superlinear*), less robust
- **Full Newton:** *Quadratic convergence* to local minimum when starting nearby



Algorithms - comparison

In practice we need to trade-off convergence rates, robustness and computational cost:

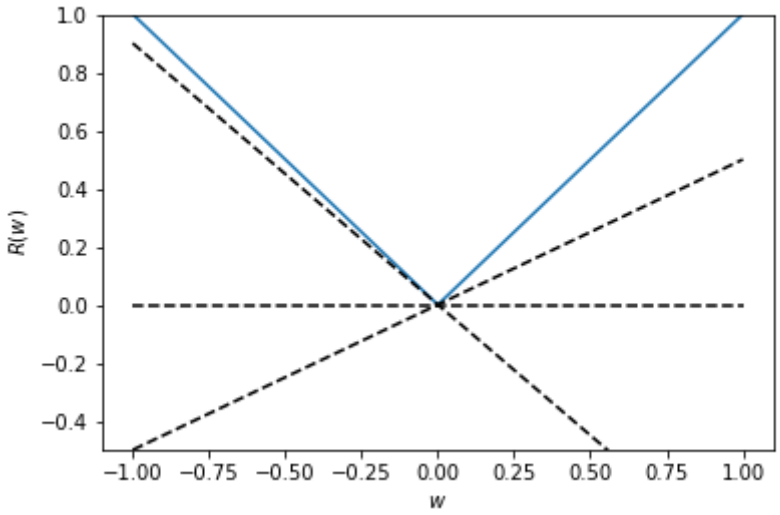
- Steepest descent has cheap iterations
- Quasi-Newton typically requires some storage to build up curvature information
- Full Newton requires the solution of a large system of linear equations at each iteration

Algorithms for non-smooth regularization

We can use *sub-gradients*:

$$w_{k+1} = w_k + \eta_k \left(\nabla L(w_k) + \partial R(w_k)\right),$$

and get a sub-linear convergence rate

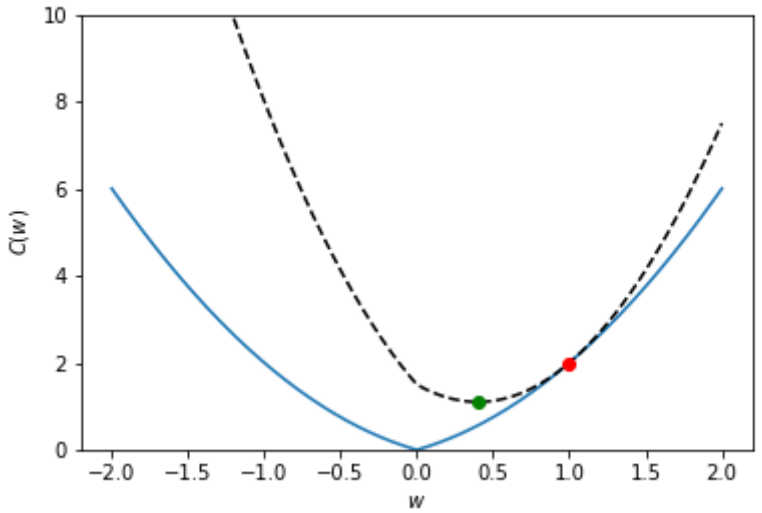


Algorithms for non-smooth regularization

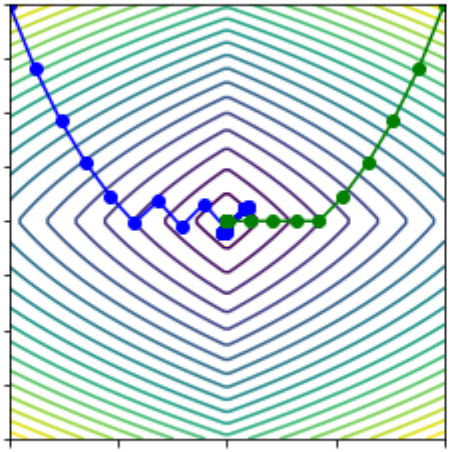
...or modify the basic iteration to retain a linear convergence rate when L is strongly convex

$$w_{k+1} = \mathrm{prox}_{\eta_k R}(w_k - \eta_k \nabla \mathrm{L}(w_k)),$$

where $\mathrm{prox}_{R}(z) = \mathrm{argmin}_x \, \mathrm{R}(x) + \frac{1}{2}\|z - x\|_2^2$.



Algorithms for non-smooth regularization



Algorithms for non-smooth regularization

Note that when R is the indicator of a convex set, the proximal operator projects onto the convex set.

- Box-constraints $\mathrm{w} \in [-1, 1]^{\mathrm{n}}$: $\mathrm{prox}(\mathrm{y}) = \mathrm{sign}(\mathrm{y}) \cdot \min\{|\mathrm{y}|, 1\}$
- 2-norm ball $\|\mathrm{w}\|_2 \leq$ ': $\mathrm{prox}(\mathrm{y}) =$ ' $\cdot\, \mathrm{y}/\|\mathrm{y}\|_2$
- 1-norm ball $\|\mathrm{w}\|_1 \leq$ ': $\mathrm{prox}(\mathrm{y}) = \mathrm{sign}(\mathrm{y}) \cdot \max\{|\mathrm{y}| -$ ', $0\}$

Other tricks of the trade

- Splitting techniques:
$$\min_{w,v} C(w) + R(v) \quad \text{s.t.} \quad w = v,$$
- Smoothing
$$\underline{R}(w) = \min_{v} R(v) + \frac{1}{2}\|w - v\|_2^2$$
- Relaxation
$$\min_{w} C(w) + \check{} \ \|Aw - b\|_2^2$$
- Partial minimization
$$\min_{w_1, w_2} C(w_1, w_2) + R_1(w_1) + R_2(w_2)$$

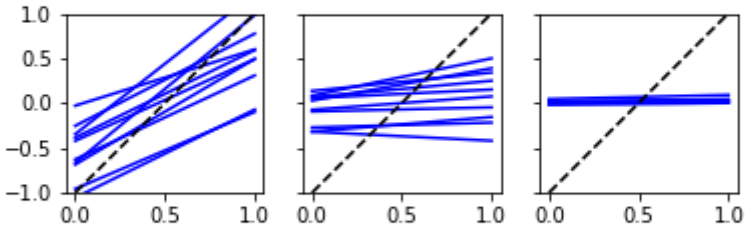
Some practical issues

- Bias-variance trade-off
- Vanishing/exploding gradient in RNN's

Bias-variance trade-off

Regularized linear regression:

$$\min_{w} \|Xw - y\|_2^2 + \tilde{} \ \|w\|_2^2.$$



Choosing an appropriate regularization parameter is its own optimization problem!

Vanishing / exploding gradients in RNN's

The function has a recursive structure:

$$f(x_0) = x_n, \qquad x_k = \sigma\left(W_{k-1}\, x_{k-1} + b_{k-1}\right).$$

Computing the gradient (*back-propagation*) requires evaluating the adjoint of the *tangent linear model*

$$\Delta x_k = \sigma'\left(W_{k-1}\, x_{k-1} + b_{k-1}\right) \cdot (W_{k-1}\, \Delta x_{k-1}).$$

This dynamical system may not be stable, even if the network is!

Some current trends

- Stochastic optimization and acceleration
- Visualization/insight

Stochastic optimization

Minimize *expected loss*:

$$\min_{w} \mathbb{E}C(w; \cdot).$$

We need to evaluate the objective at only one sample at a time:

$$w_{k+1} = w_k - \cdot_k \nabla C(w; x_k),$$

or use a *batch*

$$w_{k+1} = w_k - \frac{\cdot_k}{|\cdot_k|} \sum_{i \in \cdot_k} \nabla C(w; x_k)$$

We can interpret this algorithm as gradient-descent on the *full objective* with a noisy gradient: $\nabla C(w) + E_k$.

Stochastic optimization

Main assumptions:

- Gradient direction is correct on average: $\mathbb{E}(E_k) = 0$ (e.g., choose $\sqrt{2}$ uniformly at random from $\{1, 2, \ldots, m\}$)
- Variance of the error is bounded: $\mathbb{E}(\|E_k\|_2^2) \leq$ ` ^{2}.

Basic iteration produces iterates for which

$$\mathbb{E}(C(w_{k+1}) - C(w_*)) \leq \left(1 - 2\hat{} \; _k + ' \hat{} \; \tfrac{2}{k} L\right) (C(w_k) - C(w_*)) + \hat{} \; \frac{\tfrac{2}{k} \; ^2 L}{2}.$$

Stochastic optimization

We find

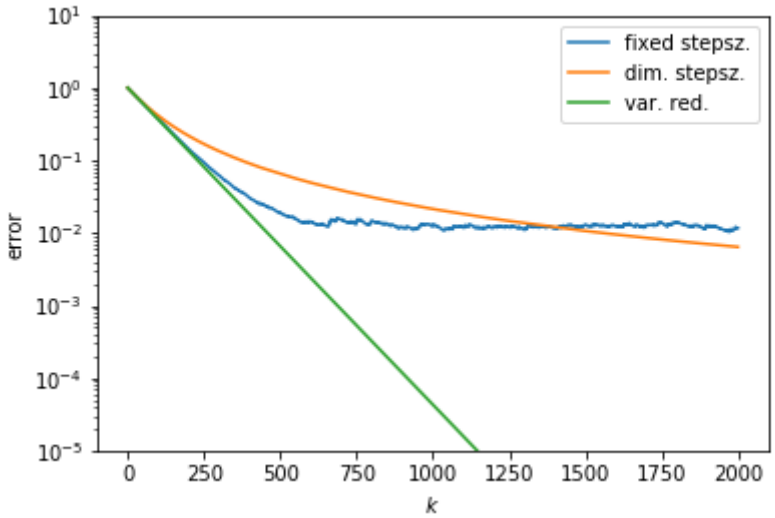
- linear convergence to a point close to a minimizer with fixed stepsize
- sublinear convergence to the minimizer when using a diminishing stepsize $\alpha_k = 1/k$.

Linear convergence to a minimizer can be retained by *variance reduction*:

- Increase *batchsize*
- Use previous gradients (SVRG, SAGA)

Constants can be improved by gradient-scaling.

Stochastic optimization

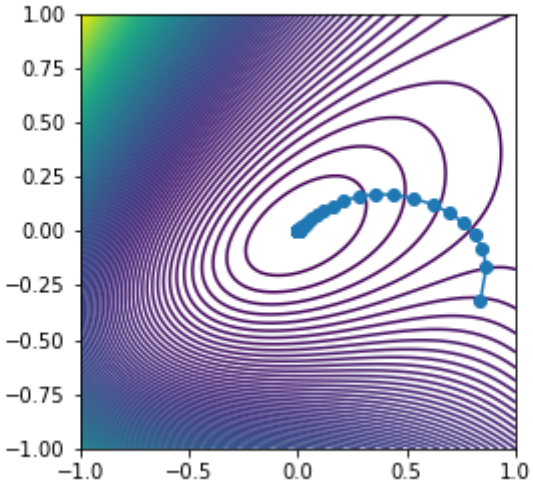


Visualization

Project iterates onto principal components of

$$\mathrm{M} = \left(\mathrm{x}_0 - \mathrm{x}_\mathrm{n}, \mathrm{x}_1 - \mathrm{x}_\mathrm{n}, \ldots, \mathrm{x}_{\mathrm{n}-1} - \mathrm{x}_\mathrm{n}\right),$$

and plot objective along dominant directions.



Summary

- Supervised learning leads to interesting and structured optimization problems
- Many specialized algorithms are available to take advantage of this structure
- Stochastic methods can be usefully analyzed as gradient-descent-with-errors

Further reading

- Extensive overview of optimization methods (http://ruder.io/optimizing-gradient-descen
- Another extensive overview (https://www.researchgate.net/profile/Frank_E_Curtis/publication/303992986_Optimiz Scale_Machine_Learning/links/5866da5908aebf17d39aeba7/Optimization-Methods-fo Learning.pdf)
- The effect of regularization in classification (https://thomas-tanay.github.io/post--L2-reg
- Well-posed network architectures (https://arxiv.org/pdf/1705.03341.pdf)
- Visualizing the loss (https://arxiv.org/pdf/1712.09913.pdf)
- Partial minimization (https://arxiv.org/abs/1601.05011)