Machine learning - HT 2016
5. Optimization

Varun Kanade

University of Oxford
February 05, 2016

Outline

Most machine learning problems can (ultimately) be cast as optimization problems.

- Linear Programming
- Basics: Gradients, Hessians
- Gradient Descent
- Stochastic Gradient Descent
- Constrained Optimization

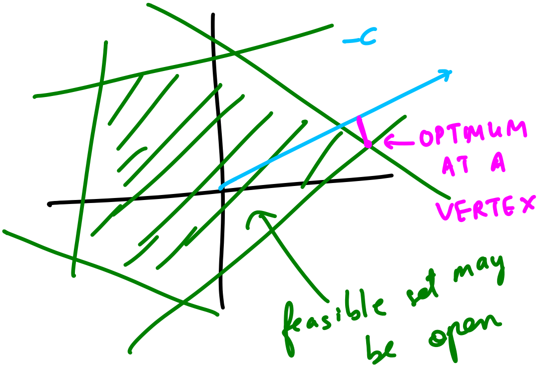
Although most software torch, octave, scikit-learn, *etc.*, will have optimization methods implemented, you will need to understand the basics of optimization to implement them effectively.

Linear Programming

Looking for solutions $\mathbf{x} \in \mathbb{R}^n$ to the following optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathbf{c}^T\mathbf{x} \\ &\text{subject to} \ \ \mathbf{a}_i^T\mathbf{x} \leq b_i, \qquad\qquad i = 1, \ldots, m \end{aligned}$$

- No analytic solution
- Efficient algorithms exist



Linear Model with Absolute Loss

Suppose we have data $\langle(\mathbf{x}_i, y_i)\rangle_{i=1}^{m}$

$L(\mathbf{w}) = \sum_{i=1}^{m} |\mathbf{x}_i^T \mathbf{w} - y_i|$

Tricks to cast problems as linear programs

Absolute Value Constraints

Constraint

$$|x| \leq a$$

Add two constraints

$$\begin{aligned} x &\leq a \\ -x &\leq a \end{aligned}$$

Max Constraints

Constraint

$$\max(x_1, x_2) \leq a$$

Add two constraints

$$\begin{aligned} x_1 &\leq a \\ x_2 &\leq a \end{aligned}$$

Linear Model with Absolute Loss

Suppose we have data $\langle(\mathbf{x}_i, y_i)\rangle_{i=1}^m$

$L(\mathbf{w}) = \sum_{i=1}^m |\mathbf{x}_i^T \mathbf{w} - y_i|$

Add auxilliary variables $\alpha_1, \dots, \alpha_m$

minimise $\sum_{i=1}^m \alpha_i$

$$\left.\begin{aligned} x_i^T \cdot w - y_i &\le \alpha_i \\ -x_i^T w + y_i &\le \alpha_i \end{aligned}\right\} \Rightarrow |x_i^T w - y_i| \le \alpha$$

(if (α^*, w^*) is optimum solution)

it must be the case that

$$\alpha_i = |x_i^T \cdot w^* - y_i|$$

Linear Model with Lasso Regularization

$$L(\mathbf{w}) = \sum_{i=1}^{m} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \sum_{i=1}^{n} |w_i|$$

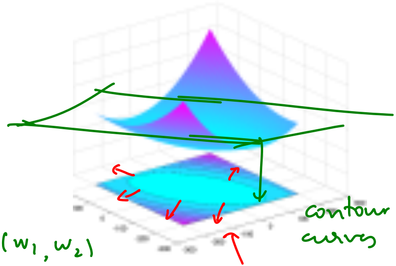
- Quadratic loss---can't frame as linear programming
- Lasso regularization does not allow for closed form solutions

Need to use general optimization methods

Calculus Background: Gradients

$$z = f(w_1, w_2) = \frac{w_1^2}{a^2} + \frac{w_2^2}{b^2}$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{pmatrix} = \begin{pmatrix} \frac{2w_1}{a^2} \\ \frac{2w_2}{b^2} \end{pmatrix}$$



contour curves

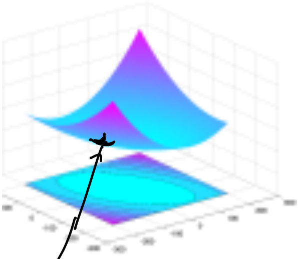
gradient points in the direction of steepest increase.

$$\frac{\partial f}{\partial w_1} = \lim_{\delta w_1 \to 0} \frac{f(w_1 + \delta w_1, w_2) - f(w_1, w_2)}{\delta w_1}$$

Calculus Background: Hessians

$$z = f(w_1, w_2) = \frac{w_1^2}{a^2} + \frac{w_2^2}{b^2}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{bmatrix} = \begin{bmatrix} \frac{2w_1}{a^2} \\ \frac{2w_2}{b^2} \end{bmatrix}$$



$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} \end{pmatrix} = \begin{pmatrix} \frac{2}{a^2} & 0 \\ 0 & 2/b^2 \end{pmatrix}$$

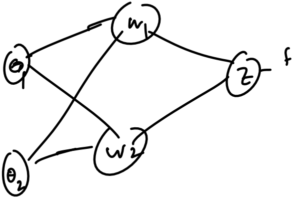
Hessian defines "curvature".

How fast is the gradient changing

If f is "nice", e.g. all second derivatives are continuous H is symmetric.

Calculus Background: Chain Rule

$$z = f(w_1(\theta_1, \theta_2), w_2(\theta_1, \theta_2))$$



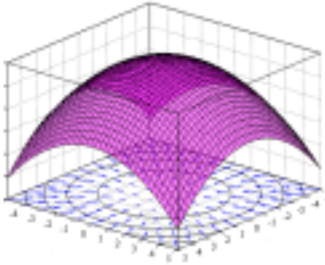
$$\frac{\partial f}{\partial \theta_1} = \frac{\partial f}{\partial w_1} \cdot \frac{\partial w_1}{\partial \theta_1} + \frac{\partial f}{\partial w_2} \cdot \frac{\partial w_2}{\partial \theta_1}$$

Use this a lot, when we do backpropagation in NN.

Gradient Vector

Suppose $\mathbf{w} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$.

$$\nabla_w f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$



Hessian matrix of f contains all second order partial derivatives.

$$\mathbf{H} = \nabla_w^2 f(w) = \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{bmatrix}$$

Gradient Descent Algorithm

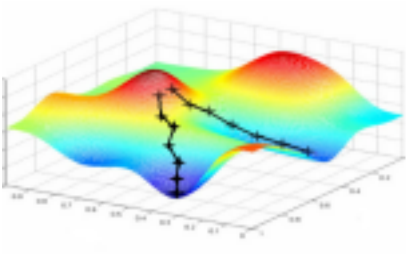
Gradient descent is one of the simplest, but very general algorithm for optimization

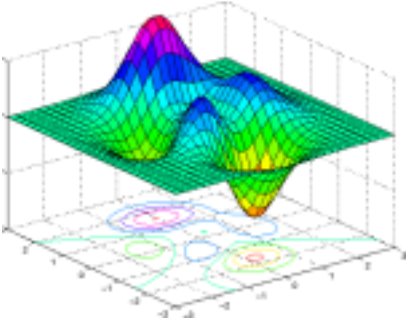
It is an iterative algorithm, producing a new $\mathbf{w}_{t+1}$ at each iteration as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t)$$

We will denote the gradients by $\mathbf{g}_t$

$\eta_t > 0$ is the learning rate or step size





Gradient Descent for Least Squares Regression

When would you want to use gradient descent to solve least squares?

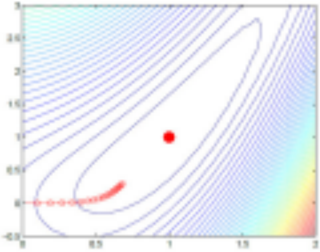
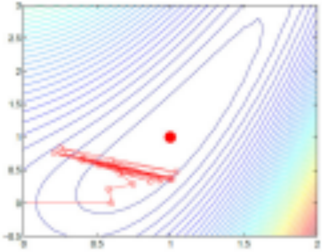
$$L(\mathbf{w}) = (\mathbf{X}\mathbf{w} - y)^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = \sum_{i=1}^{m}(\mathbf{x}_i^T\mathbf{w} - y_i)^2$$

$$\nabla_w L = 2(X^T X w - X^T y)$$

If n and m are both large, inverting $(X^T X)$ to get exact solution may be too expensive.

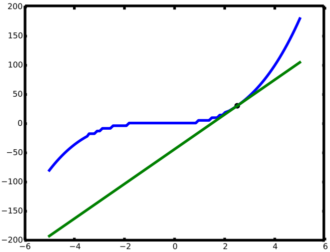
In such cases we might want to use gradient descent

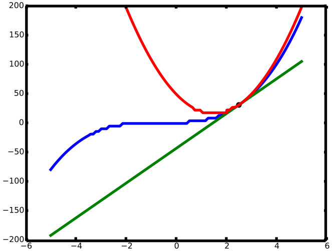
Choosing a step size

- If step size is too large, algorithm may never converge
- If step size is too small, convergence may be very slow
- May want a time-varying step size

Second Order Methods





- Gradient descent uses only the first derivative
- Local linear approximation

- Newton's method uses second derivatives
- Degree 2 Taylor approximation around current point

Newton's Method in High Dimensions

The updates depend on the gradient and the Hessian

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{H}_t^{-1}\mathbf{g}_t$$

Approximate f around $\mathbf{w}_t$ using second order Taylor approximation

$$f_{\text{quad}}(\mathbf{w}) = f(\mathbf{w}_t) + \mathbf{g}_t^T(\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T\mathbf{H}_t(\mathbf{w} - \mathbf{w}_t)$$

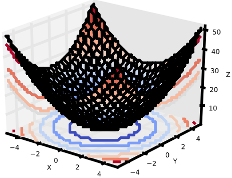
We move directly to the stationary point of f_{quad}

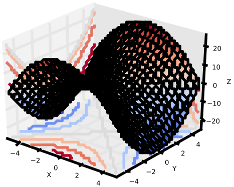
$$\nabla_w f_{quad} = g_t + H_t(w - w_t)$$

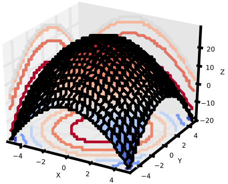
setting $\nabla_w f_{quad} = 0$ to get w_{t+1} we have

$$w_{t+1} = w_t - H_t^{-1} g_t .$$

Newton’s Method gives Stationary Points







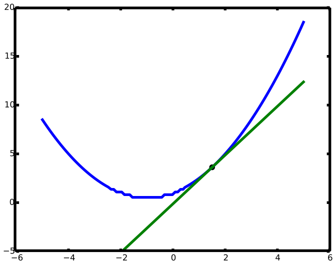
Hessian will tell you which kind of stationary point is found

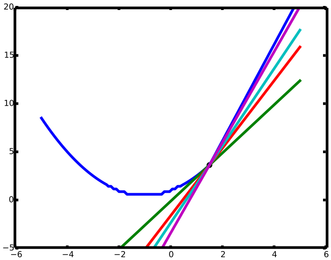
Computationally expensive--computing Hessian and inverting it at each iteration

Sub-gradient Descent

Focus on the case when f is convex,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \qquad \text{for all } x, y, \alpha \in [0,1]$$





$$f(x) \geq f(x_0) + g(x - x_0) \quad \text{where } g \text{ is a sub-derivative}$$

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \mathbf{g}(\mathbf{x} - \mathbf{x}_0) \quad \text{where } \mathbf{g} \text{ is a sub-gradient}$$

Any $\mathbf{g}$ satisfying the above inequality will be called a sub-gradient at $\mathbf{x}_0$

Sub-gradient Descent

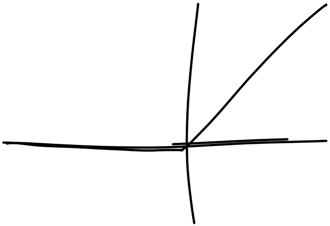
$f(\mathbf{w}) = |w_1| + |w_2| + |w_3| + |w_4|$ for $\mathbf{w} \in \mathbb{R}^4$

What is a sub-gradient at the point $(2, -3, 0, 1)$?

$$\bar{g} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

this could be any value between $[-1, 1]$

one possible subgradient



$f(x) = \max(0, x)$

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$

At 0 $[0, 1]$ is the subderivative

Learning as Optimization

Offline Learning

We have data $\mathcal{D} = \langle(\mathbf{x}_i, y_i)\rangle_{i=1}^m$. We are minimizing a loss,

$$L(\mathbf{w}) = L(\mathbf{w}, \mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + R(\mathbf{w})$$

Thus the gradient of the loss function is:

$$\nabla_{\mathbf{w}} L = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + \nabla_{\mathbf{w}} R(\mathbf{w})$$

For ridge linear regression we have

$$L(\mathbf{w}) = L(\mathbf{w}, \mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m} (\mathbf{x}_i^T \mathbf{w} - y_i)^2 + \lambda \mathbf{w}^T \mathbf{w}$$

Stochastic Gradient Descent

For learning we take the gradient of the loss function

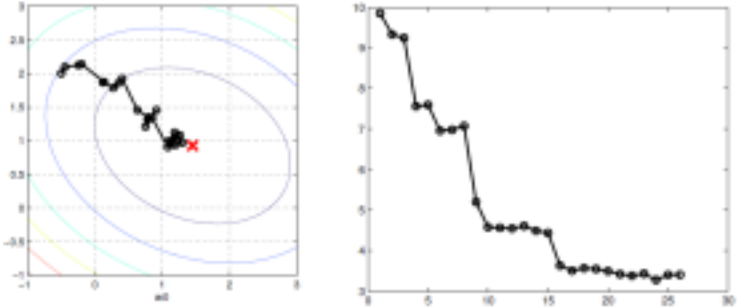
$$\nabla_{\mathbf{w}} L = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + R(\mathbf{w})$$

Suppose I pick a random point $(\mathbf{x}_i, y_i)$ and evaluate $\mathbf{g}_i = \nabla_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{x}_i, y_i)$

What is $\mathbb{E}[\mathbf{g}_i]$?

$$\mathbb{E}[g_i) = \frac{1}{m} \sum_{i=1}^{m} \nabla_w \ell(w; x_i, y_i)$$

Online Learning: Stochastic Gradient Descent



Can choose the learning rate η by cross-validation

Online Learning with mini-batches

Batch Learning

$$w_{t+1} = w_t - \frac{\eta}{m}\sum_{i=1}^{m} \nabla_w \ell(\mathbf{w}; \mathbf{x}_i, y_i) - \lambda \nabla_w R(\mathbf{w})$$

Online Learning

$$w_{t+1} = w_t - \eta \nabla_w \ell(\mathbf{w}; \mathbf{x}_i, y_i) - \lambda \nabla_w R(\mathbf{w})$$

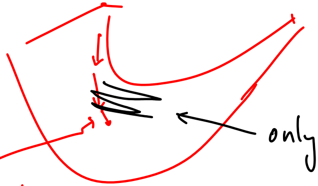
Minibatch Online Learning

$$w_{t+1} = w_t - \frac{\eta}{b}\sum_{i=1}^{b} \nabla_w \ell(\mathbf{w}; \mathbf{x}_i, y_i) - \lambda \nabla_w R(\mathbf{w})$$

Momentum

Movement is in a direction that is a combination of previous move and the gradient

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \alpha(\mathbf{w}_t - \mathbf{w}_{t-1}) + (1-\alpha)(-\mathbf{g}_t)$$



Adagrad

Text Data

Among the potential sticking points were Mr Cameron's proposals on changing the EU rules to make it easier for member states to band together to block EU laws - and plans to protect non-eurozone countries.

y	x_1	x_2	x_3	x_4
1	1	0	0	1
-1	1	1	0	0
-1	1	1	1	0
1	1	1	0	0
1	1	0	0	0
-1	1	1	1	0
1	1	1	0	0
1	1	1	0	1
1	1	1	0	0

Rarely occurring features can be highly predictive

Adagrad Update

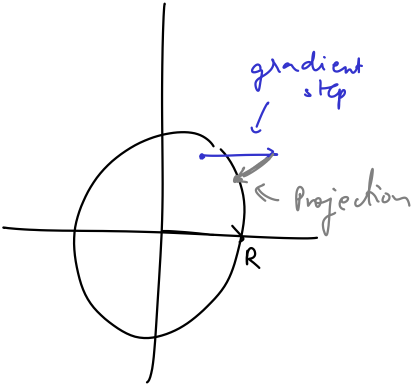
In case such as text, vectors are typically sparse. "Rare words" are most useful

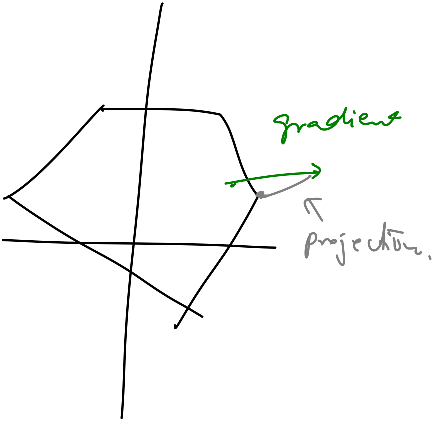
$$w_{t+1,i} \leftarrow w_{t,i} - \frac{\eta}{\sqrt{\sum_{s=1}^{t} g_{s,i}^2}} g_{t,i}$$

Constrained Convex Optimization

What if we want to look for a solution in a constrained set (not all of $\mathbb{R}^n$)?

Minimize $(\mathbf{X}\mathbf{w}-\mathbf{y})^T(\mathbf{X}\mathbf{w}-\mathbf{y})$ in the set $\mathbf{w}^T\mathbf{w} < R^2$





Summary

Convex Optimization

- Convex Optimization is (typically) efficient
- Try to cast learning problem as a convex optimization problem
- Books: Boyd and Vandenberghe, Nesterov's Book

Non-Convex Optimization

- Encountered frequently in deep learning
- Stochastic Gradient Descent gives (local) minima
- Nonlinear Programming - Dimitri Bertsekas