Optimization for Data Science

F. Rinaldi[1]

1 Università degli Studi di Padova | Dipartimento Matematica

Padova
2020

Outline

Optimization for Data Science

How to Exploit Second Order Information

Second order expansion

If the function f is twice continuously differentiable and x_k is a given point, we can write:

$$f(x_k + d) = f(x_k) + \nabla f(x_k)^\top d + \frac{1}{2} d^\top \nabla^2 f(x_k) d + \beta_2(x_k, d),$$

with

$$\lim_{\|d\| \to 0} \frac{\beta_2(x_k, d)}{\|d\|^2} = 0.$$

- Newton approach minimizes at each iteration a *quadratic approximation* of f, that is

$$\eta_k(d) := f(x_k) + \nabla f(x_k)^\top d + \frac{1}{2} d^\top \nabla^2 f(x_k) d.$$

- This η_k can be considered a good approximation of $f(x_k + d)$.

How to Exploit Second Order Information

Second order expansion

If the function f is twice continuously differentiable and x_k is a given point, we can write:

$$f(x_k + d) = f(x_k) + \nabla f(x_k)^\top d + \frac{1}{2} d^\top \nabla^2 f(x_k) d + \beta_2(x_k, d),$$

with

$$\lim_{\|d\| \to 0} \frac{\beta_2(x_k, d)}{\|d\|^2} = 0.$$

- Newton approach minimizes at each iteration a *quadratic approximation* of f, that is

$$\eta_k(d) := f(x_k) + \nabla f(x_k)^\top d + \frac{1}{2} d^\top \nabla^2 f(x_k) d.$$

- This η_k can be considered a good approximation of $f(x_k + d)$.

Details

Newton method

Starting from x_1, we use Newton method updating rule:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

where $d_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$, is the so-called *Newton direction*.

- At each iteration we get a minimizer of $\eta_k(d)$;
- We assume $\nabla^2 f(x_k)$ is positive definite.

Details

Newton method

Starting from x_1, we use Newton method updating rule:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

where $d_k = - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$, is the so-called *Newton direction*.

- At each iteration we get a minimizer of $\eta_k(d)$;
- We assume $\nabla^2 f(x_k)$ is positive definite.

Scheme

Algorithm 1 `Newton method`

```
Choose a point x_1 ∈ R^n
For k = 1, ...
        If x_k satisfies some specific condition, then STOP
        Set x_{k+1} = x_k − [∇²f(x_k)]^{−1} ∇f(x_k)
End for
```

Choose a point $x_1 \in \mathbb{R}^n$

For $k = 1, \ldots$

  If x_k satisfies some specific condition, then STOP

  Set $x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$

End for

Comments

- Using first and second order information usually speeds up the method.
- When we are sufficiently close to the solution x^*, the iterates move towards the solution with higher rate than gradient.
- Some problems can arise when building up the direction (e.g., $\nabla^2 f(x_k)$ might be singular, hence direction might not be defined in x_k).
- In order to overcome these issues, modify the direction with suitable criteria and use specific line search.

Comments

- Using first and second order information usually speeds up the method.
- When we are sufficiently close to the solution x^*, the iterates move towards the solution with higher rate than gradient.
- Some problems can arise when building up the direction (e.g., $\nabla^2 f(x_k)$ might be singular, hence direction might not be defined in x_k).
- In order to overcome these issues, modify the direction with suitable criteria and use specific line search.

Second Order Methods in Data Science

- Use of Newton method in data science applications is limited by the significant computational burden it imposes.
- Handling the Hessian matrix at each iteration is not possible for huge-scale problems.
- Consider variants that avoid using the matrix (like, e.g., Quasi-Newton methods).

Machine Learning

Roughly speaking

Machine learning studies computer algorithms for learning to do stuff (we might, for instance, be interested in learning to complete a task, or to make accurate predictions).

- Learning usually related to some sort of observations or data
- Observations can be:
  - examples (the most common case in this course);
  - direct experience;
  - instructions.
- Machine learning is about learning to do better in the future based on what was experienced in the past.
- Machine learning paradigm is "programming by example".

Machine Learning

Roughly speaking

Machine learning studies computer algorithms for learning to do stuff (we might, for instance, be interested in learning to complete a task, or to make accurate predictions).

- Learning usually related to some sort of observations or data
- Observations can be:
  - examples (the most common case in this course);
  - direct experience;
  - instructions.
- Machine learning is about learning to do better in the future based on what was experienced in the past.
- Machine learning paradigm is “programming by example”.

Some Useful Info

- We will mainly consider automatic methods here.
- We will try to devise learning algorithms that perform a given task without human intervention or assistance.
- Machine learning is directly connected to Artificial Intelligence (AI).
- Although a subarea of AI, machine learning also intersects other fields (especially statistics and optimization).

Keep in mind that

if we will ever build some intelligent system, learning will be the main tool we will use to get there.

Some Useful Info

- We will mainly consider automatic methods here.
- We will try to devise learning algorithms that perform a given task without human intervention or assistance.
- Machine learning is directly connected to Artificial Intelligence (AI).
- Although a subarea of AI, machine learning also intersects other fields (especially statistics and optimization).

Keep in mind that

if we will ever build some intelligent system, learning will be the main tool we will use to get there.

Supervised Learning and Classification Problems

Supervised Learning

- Consider a functional dependency $g : X \to Y$.
- In *supervised learning*, the goal is extracting an estimate $\hat{g}$ of g from a given finite set of training data pairs (*training set*):

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in Y \ \text{ and } i = 1, \ldots, m\}\ .$$

Classification Problems

- Input space is divided into k subsets $X_1, \ldots, X_k \in X$ such that

$$X_i \cap X_j = \emptyset \qquad i, j = 1, \ldots, k, \qquad i \neq j$$

- GOAL: assigning a given input vector x to the subset it belongs to.

Supervised Learning and Classification Problems

Supervised Learning

- Consider a functional dependency $g : X \to Y$.
- In *supervised learning*, the goal is extracting an estimate $\hat{g}$ of g from a given finite set of training data pairs (*training set*):

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in Y \ \text{ and } i = 1, \dots, m\}\ .$$

Classification Problems

- Input space is divided into k subsets $X_1, \dots, X_k \in X$ such that

$$X_i \cap X_j = \emptyset \qquad i, j = 1, \dots, k, \qquad i \neq j$$

- GOAL: assigning a given input vector x to the subset it belongs to.

Binary Classification

- We only consider *binary classification* problems.
- We have two sets $X_1, X_2 \in X$, such that $X_1 \cap X_2 = \emptyset$.
- We want to determine whether an input vector $x \in X$ belongs to X_1 or X_2.
- The training set for binary classification is

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in \{\pm 1\} \text{ and } i = 1, \dots, m\}.$$

- Two classes X_1 and X_2 labelled by $+1$ and -1, respectively.
- The functional dependency $g : X \to \{\pm 1\}$, assumes the following form:

$$g(x) = \begin{cases} +1 & \text{if } x \in X_1 \\ -1 & \text{if } x \in X_2\ . \end{cases} \tag{1}$$

Binary Classification

- We only consider *binary classification* problems.
- We have two sets $X_1, X_2 \in X$, such that $X_1 \cap X_2 = \emptyset$.
- We want to determine whether an input vector $x \in X$ belongs to X_1 or X_2.
- The training set for binary classification is

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in \{\pm 1\} \text{ and } i = 1, \ldots, m\}.$$

- Two classes X_1 and X_2 labelled by $+1$ and -1, respectively.
- The functional dependency $g : X \to \{\pm 1\}$, assumes the following form:

$$g(x) = \begin{cases} +1 & \text{if } x \in X_1 \\ -1 & \text{if } x \in X_2\ . \end{cases} \tag{1}$$

Binary Classification

- We only consider *binary classification* problems.
- We have two sets $X_1, X_2 \in X$, such that $X_1 \cap X_2 = \emptyset$.
- We want to determine whether an input vector $x \in X$ belongs to X_1 or X_2.
- The training set for binary classification is

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in \{\pm 1\} \ \text{and}\ i = 1, \ldots, m\}.$$

- Two classes X_1 and X_2 labelled by $+1$ and -1, respectively.
- The functional dependency $g : X \to \{\pm 1\}$, assumes the following form:

$$g(x) = \begin{cases} +1 & \text{if } x \in X_1 \\ -1 & \text{if } x \in X_2 \ . \end{cases} \tag{1}$$

Classes of Learning Machines

Perceptron

Basic calculation unit in learning machines.

MultiLayer Perceptron Networks (MLPN)

A multilayer network typically consists of

- an *input layer*, which is basically a set of source nodes;
- one or more *hidden layers*, composed by various computational nodes;
- an *output layer* of computational nodes.

We can construct (*train*) the desired network $\hat{g}$ in a supervised manner by using a popular algorithm known as the *backpropagation algorithm*.

Classes of Learning Machines

Perceptron

Basic calculation unit in learning machines.

MultiLayer Perceptron Networks (MLPN)

A multilayer network typically consists of

- an *input layer*, which is basically a set of source nodes;
- one or more *hidden layers*, composed by various computational nodes;
- an *output layer* of computational nodes.

We can construct (*train*) the desired network $\hat{g}$ in a supervised manner by using a popular algorithm known as the *backpropagation algorithm*.

Classes of Learning Machines II

Radial Basis Function Networks (RBFN)

A radial basis function network has three layers:

- First layer is the input layer, a set of source nodes used to connect the network to its environment.
- The second layer (the only hidden layer) maps the input vector x into a hidden space of high dimensionality.
- The last layer (output layer) gives the response of the network to a given input vector x.

Design a neural network as a curve-fitting problem in a high-dimensional space by means of radial basis functions.

Classes of Learning Machines III

Support Vector Machines (SVM)

- Support vector machines, introduced by Vapnik, represent another efficient tool for classification.
- This class of learning machines implements in an approximate way the method of *structural risk minimization*.

As we will see, training a support vector machine requires the solution of a large dense quadratic programming problem.

Generalization

- The hope is that the estimate $\hat{g}$ of g will *generalize*.
- A learning machine is said to generalize well when computes correctly the input-output mapping for *test data* not included in the training set.
- Generalization strictly connected with complexity of the machine.
- A complex estimate $\hat{g}$ usually approximates g poorly on points not in the training set (*overfitting*).
- A very simple model not preferred as it gives too poor a fit to the training data.

Optimal complexity by *Occam's Razor* (named after William of Occam (1285-1349))

Choose simplest model possible that still grants good performance on the training data.

Generalization

- The hope is that the estimate $\hat{g}$ of g will *generalize*.
- A learning machine is said to generalize well when computes correctly the input-output mapping for *test data* not included in the training set.
- Generalization strictly connected with complexity of the machine.
- A complex estimate $\hat{g}$ usually approximates g poorly on points not in the training set (*overfitting*).
- A very simple model not preferred as it gives too poor a fit to the training data.

Optimal complexity by *Occam's Razor* (named after William of Occam (1285-1349))

Choose simplest model possible that still grants good performance on the training data.

Generalization

- The hope is that the estimate $\hat{g}$ of g will *generalize*.
- A learning machine is said to generalize well when computes correctly the input-output mapping for *test data* not included in the training set.
- Generalization strictly connected with complexity of the machine.
- A complex estimate $\hat{g}$ usually approximates g poorly on points not in the training set (*overfitting*).
- A very simple model not preferred as it gives too poor a fit to the training data.

Optimal complexity by *Occam's Razor* (named after William of Occam (1285-1349))

Choose simplest model possible that still grants good performance on the training data.

Evaluate Generalization

Cross-Validation

In order to evaluate the generalization ability of a learning machine, we can use the procedure of *cross-validation*:

- Split the training set T into k distinct segments $T_1, \ldots, T_k$.
- Construct the function $\hat{g}$ using data from $k-1$ segments (test performance using the remaining segment).
- Repeat for each of the k possible choices, and consider average over k.
- When k is equal to the number of training data we obtain the *leave-one-out method*.

Remark

What about No Free-Lunch Theorem?

Evaluate Generalization

Cross-Validation

In order to evaluate the generalization ability of a learning machine, we can use the procedure of *cross-validation*:

- Split the training set T into k distinct segments $T_1, \ldots, T_k$.
- Construct the function $\hat{g}$ using data from $k-1$ segments (test performance using the remaining segment).
- Repeat for each of the k possible choices, and consider average over k.
- When k is equal to the number of training data we obtain the *leave-one-out method*.

Remark

What about No Free-Lunch Theorem?