Linear model training 1 / 2

Data distribution

We denote $p(\mathbf{x}, \mathbf{y})$ the data distribution where:

- $\mathbf{x}$: random variables over inputs
- $\mathbf{y}$: random variables over outputs

Linear model training 1 / 2

Data distribution

We denote $p(\mathbf{x}, \mathbf{y})$ the data distribution where:

- $\mathbf{x}$: random variables over inputs
- $\mathbf{y}$: random variables over outputs

Training problem

Find the model parameters that minimize the expected loss of the data distribution:

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\ \ell(\mathbf{y}, s_{\boldsymbol{\theta}}(\mathbf{x}))\ ] \quad + \quad \beta r(\theta)$$

- ℓ: loss function
- r: regularization function, usually not applied to all parameters in $\boldsymbol{\theta}$ (i.e. not applied to the bias/intercept term)
- $\beta \geq 0$: regularization weight

Linear model training 2 / 2

Monte-Carlo estimation

We approximate the true expected loss using samples from the data distribution:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\ \ell(\mathbf{y}, s_\theta(\mathbf{x}))\ ] \quad \simeq \quad \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} \ell(\boldsymbol{y}, s_\theta(\boldsymbol{x}))$$

where the training dataset D contains $|D|$ samples from the data distribution.

Linear model training 2 / 2

Monte-Carlo estimation

We approximate the true expected loss using samples from the data distribution:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\ \ell(\mathbf{y}, s_\theta(\mathbf{x}))\ ] \quad \simeq \quad \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \ell(\boldsymbol{y}, s_\theta(\boldsymbol{x}))$$

where the training dataset D contains $|D|$ samples from the data distribution.

Convexity

If

- the scoring function is linear
- the loss is convex
- the regularization function is convex

then the training problem object is convex.

Generic optimization problem

Reweighting

Sometimes it is easier to absord the $\frac{1}{|D|}$ factor in the regularization weight:

$$
\begin{aligned}
& \arg\min_{\boldsymbol{\theta}} \quad \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} \ell(\boldsymbol{y}, s_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad + \quad \beta r(\theta) \\
= \quad & \arg\min_{\boldsymbol{\theta}} \quad \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} \ell(\boldsymbol{y}, s_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad + \quad \underbrace{|D|\beta}_{\substack{\text{new reg.} \\ \text{weight}}} \; r(\theta)
\end{aligned}
$$

Generic optimization problem

Reweighting

Sometimes it is easier to absord the $\frac{1}{|D|}$ factor in the regularization weight:

$$\begin{aligned} & \arg\min_{\boldsymbol{\theta}} \quad \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \ell(\boldsymbol{y}, s_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad + \quad \beta r(\theta) \\ = \quad & \arg\min_{\boldsymbol{\theta}} \quad \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \ell(\boldsymbol{y}, s_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad + \quad \underbrace{|D|\beta}_{\substack{\text{new reg.} \\ \text{weight}}} \; r(\theta) \end{aligned}$$

Generic problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be two convex functions.

$$\min_{\boldsymbol{u}\in \mathbf{dom}\, f} f(\boldsymbol{u}) \qquad \text{or} \qquad \min_{\boldsymbol{u}\in \mathbf{dom}\, f \cap \mathbf{dom}\, h} f(\boldsymbol{u}) + h(\boldsymbol{u}) \qquad \text{or} \qquad \min_{\boldsymbol{u}\in \mathbf{dom}\, f \cap \mathbf{dom}\, h} f(\boldsymbol{M}\boldsymbol{u}) + h(\boldsymbol{u})$$

Gradient descent

Generic optimization problem

Let's consider the following optimization problem:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} f(\boldsymbol{u})$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed and convex function.

Gradient descent algorithm

Assume f is differentiable everywhere in its domain. The gradient descent algorithm is an iterative optimization algorithm that searches for the minimum of f by considering a sequence of points as follows:

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} - \epsilon^{(t)} \nabla f(\boldsymbol{u}^{(t)})$$

- $\epsilon^{(t)}$ is the stepsize at time step t
- initial point $\boldsymbol{u}^{(0)} \in \textbf{dom}\, f$ can be chosen randomly

Why does it work?

Theorem: Descent direction

Let $\boldsymbol{u}$ be a non optimal point, i.e. $\nabla f(\boldsymbol{u}) \neq 0$.
Then, there exist ϵ such that:

$$f(\boldsymbol{u} - \epsilon \nabla f(\boldsymbol{u})) < f(\boldsymbol{u})$$

We say that $-\nabla f(\boldsymbol{u})$ is a descent direction.
Proof: See [Boyd et al., 2004, Sections 9.2 and 9.3] and [Beck, Lemma 5.7]

Why does it work?

Theorem: Descent direction

Let $\boldsymbol{u}$ be a non optimal point, i.e. $\nabla f(\boldsymbol{u}) \neq 0$.
Then, there exist ϵ such that:

$$f(\boldsymbol{u} - \epsilon \nabla f(\boldsymbol{u})) < f(\boldsymbol{u})$$

We say that $-\nabla f(\boldsymbol{u})$ is a descent direction.
Proof: See [Boyd et al., 2004, Sections 9.2 and 9.3] and [Beck, Lemma 5.7]

Stepsize

How to choose the stepsize?

- line search: (approximately) search for the best stepsize, i.e. solve $\epsilon^{(t)} = \arg\min_{\epsilon > 0} f(\boldsymbol{x}^{(t)} - \epsilon \nabla f(\boldsymbol{x}^{(t)}))$
- constant stepsize
- diminishing stepsize: start with a given stepsize and decrease its value each t steps or according to the function evaluation / dev data evaluation

Stochastic gradient descent 1 / 3

Let's consider the following optimization problem:

$$\min_{\boldsymbol{u}\in\mathbb{R}^n} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{u})$$

where $\forall i \in \{1...n\}, f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a set of proper closed convex functions, we assume the intersection of their domain is a non-empty convex set.

In stochastic gradient descent, at each step the gradient is approximated using a subset of the functions f_i:

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} - \frac{\epsilon^{(t)}}{|\mathrm{I}(t)|}\sum_{i\in\mathrm{I}(t)} \nabla f_i(\boldsymbol{u})$$

where $\mathrm{I}(t) \subseteq \{1...n\}$ is the subset of indices used at step t.
$\implies$ the subset of should consist of uniformly sampled indices!

Stochastic gradient descent 2 / 3

Machine learning application

We call $\mathrm{I}(t)$ a mini-batch and it consists of a subset of the training data.

$$\min_{\boldsymbol{\theta}} \quad \underbrace{\frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} \ell(\boldsymbol{y}, s_{\boldsymbol{\theta}}(\boldsymbol{x}))}_{\substack{\text{Approximate this term} \\ \text{using a subset of datapoints}}} \quad + \quad \alpha r(\theta)$$

Two approaches

- Sampling with replacement: at each step, randomly choose a subset of datapoints
- Sampling without replacement: optimization is based on a sequence of epochs
  - randomly choose of subset of datapoints that you did not see in the current epoch yet
  - an epoch is over when you saw all datapoints

$\Longrightarrow$ Sampling without replacement is standard in ML

Stochastic gradient descent 3 / 3

```
# Loop over epoch
for epoch in range(num_epochs):
    random.shuffle(training_data)

    # Loop over minibatches
    for i in range(0, len(training_data), minibatch_size):
        minibatch = training_data[i : i + minibatch_size]

        optimization_step(minibatch)

    # Evaluate on dev data
    evaluate_on_dev()
```

Other tricks:

- Save the model that obtain the best results on dev
- Control stepsize thanks to dev results

Coordinate descent

Coordinate descent

Motivations

All these algorithms require a stepsize, which may be difficult to tune.

Is there any method that does not depend on a stepsize?

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a proper, closed, convex and differentiable function. Assume a problem of the form:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} f(\boldsymbol{u})$$

The coordinate descent algorithm is an iterative optimization algorithm that searches for the minimum of f by considering a sequence of points as follows:

$$u_1^{(t+1)} \quad \in \quad \arg\min_{u_1 \in \mathbb{R}} f(\ [\ u_1, u_2^{(t)}, u_3^{(t)}, ..., u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ )$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a proper, closed, convex and differentiable function. Assume a problem of the form:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} f(\boldsymbol{u})$$

The coordinate descent algorithm is an iterative optimization algorithm that searches for the minimum of f by considering a sequence of points as follows:

$$\begin{aligned}
u_1^{(t+1)} &\in \arg\min_{u_1 \in \mathbb{R}} f(\ [\ u_1, u_2^{(t)}, u_3^{(t)}, \ldots, u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
u_2^{(t+1)} &\in \arg\min_{u_2 \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2, u_3^{(t)}, \ldots, u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ )
\end{aligned}$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a proper, closed, convex and differentiable function. Assume a problem of the form:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} f(\boldsymbol{u})$$

The coordinate descent algorithm is an iterative optimization algorithm that searches for the minimum of f by considering a sequence of points as follows:

$$\begin{aligned}
u_1^{(t+1)} &\in \arg\min_{u_1 \in \mathbb{R}} f(\ [\ u_1, u_2^{(t)}, u_3^{(t)}, ..., u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
u_2^{(t+1)} &\in \arg\min_{u_2 \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2, u_3^{(t)}, ..., u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
u_3^{(t+1)} &\in \arg\min_{u_3 \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2^{(t+1)}, u_3, ..., u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ )
\end{aligned}$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a proper, closed, convex and differentiable function. Assume a problem of the form:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} f(\boldsymbol{u})$$

The coordinate descent algorithm is an iterative optimization algorithm that searches for the minimum of f by considering a sequence of points as follows:

$$\begin{aligned}
u_1^{(t+1)} &\in \arg\min_{u_1 \in \mathbb{R}} f(\ [\ u_1, u_2^{(t)}, u_3^{(t)}, \ldots, u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
u_2^{(t+1)} &\in \arg\min_{u_2 \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2, u_3^{(t)}, \ldots, u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
u_3^{(t+1)} &\in \arg\min_{u_3 \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2^{(t+1)}, u_3, \ldots, u_{n-1}^{(t)}, u_n^{(t)}\ ]^\top\ ) \\
&\ldots \\
u_{n-1}^{(t+1)} &\in \arg\min_{u_{n-1} \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2^{(t+1)}, u_3^{(t+1)}, \ldots, u_{n-1}, u_n^{(t)}\ ]^\top\ ) \\
u_n^{(t+1)} &\in \arg\min_{u_n \in \mathbb{R}} f(\ [\ u_1^{(t+1)}, u_2^{(t+1)}, u_3^{(t+1)}, \ldots, u_{n-1}^{(t+1)}, u_n\ ]^\top\ )
\end{aligned}$$

Or any other order, as long as you directly use the new value for the next coordinate.