Recitation 24

Thursday December 8, 2022

1 Recap - Gradient Descent

Algorithm 1: Gradient Descent

```
Input: initial guess $x_0$, step size $\gamma > 0$;
while $\nabla f(x_k) \neq 0$ do
    $x_{k+1} = x_k - \gamma \nabla f(x_k)$
end
return $x_k$;
```

1.1 Stochastic GD

In Vanilla Gradient Descent (**Batch Gradient Descent**), we compute the gradient using all the data points in each iteration. This can be slow and redundant for large data sets. **Stochastic gradient descent** (SGD) in contrast performs gradient update for one randomly chosen training example at each iteration. Moreover, a method called **mini-batch gradient descent** combines both of them and computes the gradient based on a small batch of data points at each iteration.

1.2 Logistic Regression

Gradient descent is used for solving many machine learning problems, and one example is logistic regression. In the logistic regression model, we first apply a linear transformation on the input points x: $z = wx + b$. Then, we use the sigmoid function (aka logistic function) to classify the transformed points: $\sigma(z) = \frac{1}{1+e^{-z}}$.

1. Sigmoid
   The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real-valued number and maps it to the range [0,1], as if we are assigning a probability.

2. Decision Boundary
   We model $\sigma(z)$ as the probability that the corresponding label $y = 1$, and $1 - \sigma(z)$ as the probability that $y = 0$. Then we predict $\hat{y} = 1$ if $\sigma(z) > 0.5$, and $\hat{y} = 0$ otherwise.

3. Loss function
   Because of this relationship with probability, we choose a loss function called the cross entropy, or negative log likelihood:

   $$L(x, y; w) = -\sum y \log(\sigma(wx + b)) + (1 - y) \log(1 - \sigma(wx + b)).$$

   This loss function is conveniently convex, and we can find the global minimum using gradient descent.

1.3 Perceptron

Algorithm 2: Perceptron

```
Input: initialize $w_0 = 0 \in \mathbb{R}^d$, step size $\eta \in [0, 1]$;
for $i \in [N]$ do
    Predict: $y'_i = sgn(\langle w_t, x_i \rangle)$
    If $y'_i \neq y_i$: $w_{t+1} = w_t + \eta y_i x_i$
end
return $w_N$;
```

Margin Assumption: Suppose there is a unit vector u that satisfies

1. for all x_i labeled $+1$ we have $\langle u, x_i \rangle \geq \gamma$
2. For all x_i labeled -1 we have $\langle u, x_i \rangle \leq -\gamma$

Given the margin assumption the following theorem holds

Theorem: Suppose the margin assumption holds and furthermore each example x_i satisfies $\|x_i\| \leq R$. Setting $\eta = 1$ we have the number of mistakes where $y_i \neq y'_i$ is upper bounded by $(\frac{R}{\gamma})^2$.

2 Exercises

1. For a dataset $\{(x_i, y_i)\}_{i\in[N]}$ for $x_i \in \mathbb{R}^d$, there exists $z \in \mathbb{R}^d$ such that $sgn(\langle u, x_i - z\rangle) = y_i$. How can we modify the input to the perceptron to learn u and z?

2. Consider the dataset of (x, y) pairs for $x \in \mathbb{R}$ and $y \in \{\pm 1\}$ as follows $\{(-4, +1), (-3, +1), (-2, -1), (-1, -1), (0, -1), (1, -1), (2, -1), (3, +1), (4, +1)\}$ The data is not linearly separable. How does applying the mapping $\phi(x) = (x, x^2)$ change this?

3 Solutions

1. Take input covariates $\{x_i\}_{i\in[N]}$ and pad them by a scalar -1 so that the covariates become $\{(x_i, -1)\}_{i\in[N]}$. The perceptron on the modified dataset learns (u, b) for some scalar b such that $\langle u, z\rangle = b$. Given u and b we can solve for z.

2. After applying the mapping $\phi(x) = (x, x^2) = (\phi(x)_1, \phi(x)_2)$ the data is linearly separable across the line $\phi(x)_2 = 4$ in 2 dimensions.