Stanford University

CS 329P : Practical Machine Learning (2021 Fall)

3.3 Linear Methods

Qingqing Huang, Mu Li, Alex Smola

https://c.d2l.ai/stanford-cs329p

Linear Regression

- A simple house price prediction
  - Assume 3 features: $x_1 = $ #beds, $x_2 = $ #baths, $x_3 = $ living sqft
  - The predicted price is $\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$
  - Weights w_1, w_2, w_3 and bias b will be learnt from training data
- In general, given data $\mathbf{x} = [x_1, x_2, \ldots, x_p]$, linear regression predicts

$$\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_p x_p + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

- Code:

```
# weight w has shape (p, 1)
# bias b is a scalar
# data x has shape (p, 1)
y_hat = (x*w).sum() + b
```

$_2, \ldots, x_p]$ and bias b

Objective Function

- Collect n training examples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times p}$ with labels $\mathbf{y} = [y_1, \ldots, y_n]^T \in \mathbb{R}^n$
  - E.g. house listings and sold prices
- Objective: minimize the mean square error (MSE)

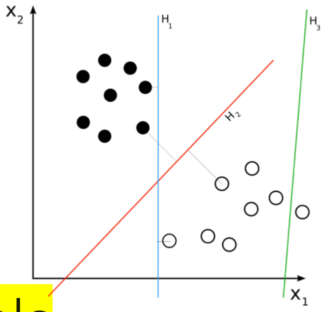
$$\mathbf{w}^*, \mathbf{b}^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\, \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

$$= \underset{\mathbf{w}, b}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b)^2$$

- Exercise: write the closed form solution

Use linear regression for classification problem

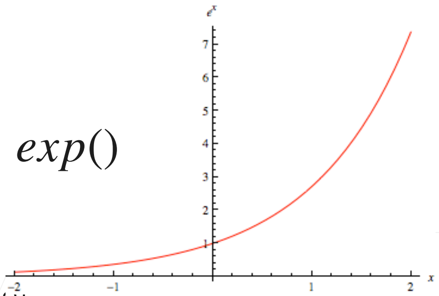
- Regression: continuous output in $\mathbb{R}$
- Multi-class classification:
  - One-hot label $\mathbf{y} = [y_1, y_2, \ldots, y_m]$ where $y_i = 1$ if $i = y$ otherwise 0
  - $\hat{\mathbf{y}} = \mathbf{o}$, where i-th output o_i is the confidence score for class i
  - Learn a liner model for each class $o_i = \langle \mathbf{x}, \mathbf{w}_i \rangle + b_i$
  - Minimize MSE loss $\frac{1}{m}\|\mathbf{o} - \mathbf{y}\|_2^2$
  - Predict label $\operatorname{argmax}_i \{o_i\}_{i=1}^m$
- Waste model capacity on pushing o_i near 0 for off labels



Softmax Regression

- One-hot label $\mathbf{y} = [y_1, y_2, \ldots, y_m]$ where $y_i = 1$ if $i = y$ otherwise 0
- $$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \text{ where } \hat{y}_i = \frac{\exp(o_i)}{\sum_{k=1}^{m} \exp(o_k)}$$
  - Turns confidence scores into probabilities (non-negative, sum to 1)
  - Ideally we want $\hat{\mathbf{y}} = \text{one-hot}(\mathrm{argmax}_i o_i)$, softmax is a continuous approximate to that
  - Still a linear model, decision made on linear transformation of the input, as $\mathrm{argmax}_i\ \hat{y}_i = \mathrm{argmax}_i\ o_i$

exp()



- Cross-entropy loss between two distributions $\hat{\mathbf{y}}$ and $\mathbf{y}$

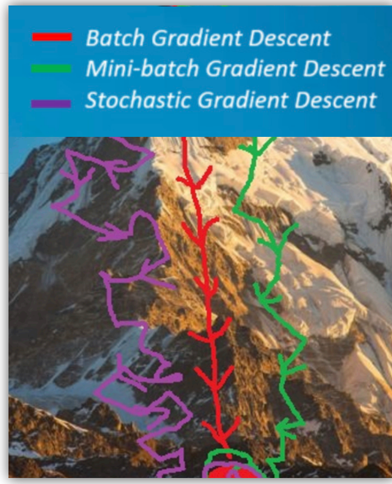
$$H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_i -y_i \log(\hat{y}_i) = -\log \hat{y}_y$$

  - When label class is i, assigns less penalty on o_j as long as $o_j \ll o_i$

- Exercise: think about how to handle examples with multi labels?

Mini-batch Stochastic gradient descent (SGD)

- Train by mini-batch SGD (by various other ways as well)
  - $\mathbf{w}$ model param, b batch size, η_t learning rate at time t
  - Randomly initialize $\mathbf{w}_1$
  - Repeat $t = 1,2,\ldots$ until converge
    - Randomly samples $I_t \subset \{1,\ldots,n\}$ with $|I_t| = b$
    - Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}_t} \ell(\mathbf{X}_{I_t}, \mathbf{y}_{I_t}, \mathbf{w}_t)$
- Pros: solve all objectives in this course except for trees
- Cons: sensitive to hyper-parameters b and η_t



Code

- Train a linear regression model with min-batch SGD
- Hyperparameters
  - batch_size
  - learning_rate
  - num_epochs

Full code at: http://d2l.ai/chapter_linear-networks/linear-regression-scratch.html

```python
# `features` shape is (n, p), `labels` shape is (n, 1)
def data_iter(batch_size, features, labels):
    num_examples = len(features)
    indices = list(range(num_examples))
    random.shuffle(indices)  # read examples at random
    for i in range(0, num_examples, batch_size):
        batch_indices = torch.tensor(
            indices[i:min(i + batch_size, num_examples)])
        yield features[batch_indices], labels[batch_indices]

w = torch.normal(0, 0.01, size=(p, 1), requires_grad=True)
b = torch.zeros(1, requires_grad=True)

for epoch in range(num_epochs):
    for X, y in data_iter(batch_size, features, labels):
        y_hat = X @ w + b
        loss = ((y_hat - y)**2 / 2).mean()
        loss.backward()
        for param in [w, b]:
            param -= learning_rate * param.grad
            param.grad.zero_()
```

Summary

- Linear methods linearly combine inputs to obtain predictions
- Linear regression uses MSE as the loss function
- Softmax regression is used for multiclass classification
  - Turn predictions into probabilities and use cross-entropy as loss
  - Cross entropy loss between two probability distribution
- Mini-batch SGD can learn both models (and later neural networks as well)