CS540 Intro to AI

Linear Models in Machine Learning

Lecturer: Xiaojin Zhu *jerryzhu@cs.wisc.edu*

We briefly go over two linear models frequently used in machine learning: linear regression for, well, regression, and logistic regression for classification.

1 Linear regression

1.1 The 1D case without regularization (Ordinary Least Squares)

x: input variable (also called independent, predictor, explanatory variable)
y: output variable (also called dependent, response variable)
A scatter plot shows (x_i, y_i) for $i = 1 \dots n$ as dots.
Model assumption:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$.
On a test point x^*,

$$\mathbb{E}(Y \mid x^*) = \mathbb{E}(\beta_0 + \beta_1 x^* + \epsilon) = \beta_0 + \beta_1 x^* + \mathbb{E}(\epsilon) = \beta_0 + \beta_1 x^*.$$

$$\sigma^2_{(Y|x^*)} = \sigma^2.$$

Given data, maximum likelihood estimate = least squares estimate

$$(\hat{\beta}_0, \hat{\beta}_1) = \text{argmin}_{b_0, b_1} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)^2.$$

Convex objective.
Setting gradient to zero gives

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}.$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.$$

We used $\bar{x} = (\sum x_i)/n$ and $\bar{y} = (\sum y_i)/n$. This is known as ordinary least squares (OLS).
Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$. The residual is $y_i - \hat{y}_i$. The residual sum of squares is

$$\sum (y_i - \hat{y}_i)^2,$$

and an unbiased estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{\sum (y_i - \hat{y}_i)^2}{n - 2}.$$

The denominator is $n - 2$ because $\hat{\beta}_0, \hat{\beta}_1$ used up two degrees of freedom.
Note the best constant fit would have been $f(x) = \bar{y}$, whose residual sum of squares is

$$\sum (y_i - \bar{y})^2.$$

The coefficient of determination r^2 is a measure of how much better a non-constant line fit is:

$$r^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}.$$

$r^2 \in [0, 1]$, 1 being desirable.

1.2 Ridge regression

Input variables $\mathbf{x} = (x_0 = 1, x_1, \ldots, x_p)^\top \in \mathbb{R}^{p+1}$, output variable $y \in \mathbb{R}$.

$$y = \mathbf{x}^\top \beta + \epsilon.$$

$\epsilon \sim N(0, \sigma^2)$. Linear in β. x_i can be nonlinear functions of input. Examples:

- pth-degree polynomial regression

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p + \epsilon.$$

- second order model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1 x_2 + \beta_5 x_2^2 + \epsilon.$$

- In general

$$y = \beta_0 + \sum_j \beta_j \phi_j(\mathbf{x}) + \epsilon,$$

  where $\phi_j()$'s are arbitrary feature functions of $\mathbf{x}$.

An unbiased estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{\sum (y_i - \hat{y}_i)^2}{n - (p+1)}.$$

It will be convenient to arrange the input in a matrix $X_{n \times p}$ where each row is a feature vector, and a label vector $\mathbf{y}_{n \times 1}$. The linear regression model is

$$\mathbf{y} = X\beta + \epsilon$$

where ϵ is a vector, too.

Without regularization, the ordinary least squares (OLS) method finds parameter

$$\min_\beta \|\mathbf{y} - X\beta\|^2.$$

Rewrite the objective as

$$(\mathbf{y} - X\beta)^\top (\mathbf{y} - X\beta) = \mathbf{y}^\top \mathbf{y} - 2\beta^\top X^\top \mathbf{y} + \beta^\top X^\top X \beta.$$

Take the gradient with respect to β and set it to the zero vector:

$$-2X^\top \mathbf{y} + 2X^\top X\beta = 0$$

Assuming $X^\top X$ is invertible we obtain the solution

$$\beta = (X^\top X)^{-1} X^\top \mathbf{y}.$$

OLS is certainly not going to work, however, in the high dimension setting when $p > n$, since $X^\top X$ is rank deficient and not invertible.

More frequently used in practice for its stability, ridge regression solves a regularized optimization problem instead:

$$\min_\beta \|\mathbf{y} - X\beta\|^2 + \lambda \|\beta\|^2. \tag{1}$$

Rewrite the objective as

$$\mathbf{y}^\top \mathbf{y} - 2\beta^\top X^\top \mathbf{y} + \beta^\top X^\top X \beta + \lambda \beta^\top \beta.$$

Take the gradient with respect to β and set it to the zero vector:

$$-2X^\top \mathbf{y} + 2X^\top X\beta + 2\lambda\beta = 0$$

$$-2X^\top \mathbf{y} + 2(X^\top X + \lambda I)\beta = 0$$

The solution is

$$\beta = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}.$$

Note $(X^\top X + \lambda I)$ is always invertible. With appropriate λ ridge regression can have smaller mean squared error.

2 Logistic Regression

Consider binary classification with $y = -1, 1$, and each example is represented by a feature vector x. The intuition is to first map x to a real number, such that large positive number means that x is likely to be positive ($y = 1$), and negative number means x is negative ($y = -1$). This is done just like in linear regression:

$$\theta^\top x \tag{2}$$

The next step is to "squash" the range down to $[0, 1]$ so one can interpret it as the label probability. This is done via the *logistic* function:

$$p(y = 1|x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)}. \tag{3}$$

For binary classification with -1, 1 class encoding, one can unify the definition for $p(y = 1|x)$ and $p(y = -1|x)$ with

$$p(y|x) = \frac{1}{1 + \exp(-y\theta^\top x)}. \tag{4}$$

Logistic regression can be easily generalized to multiple classes. Let there be K classes. Each class has its own[1] parameter θ_k. The probability is defined via the *softmax* function

$$p(y = k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^K \exp(\theta_i^\top x)}. \tag{5}$$

If a predicted class label (instead of a probability) is desired, simply predict the class with the largest probability $p(y = k|x)$. Logistic regression produces linear decision boundaries between classes.

We will focus on binary classification in the rest of this note.

2.1 Training

Training (i.e., finding the parameter θ) can be done by maximizing the *conditional* log likelihood of the training data $\{(x, y)_{1:n}\}$:

$$\max_\theta \sum_{i=1}^n \log p(y_i|x_i, \theta). \tag{6}$$

However, when the training data is linearly separable, two bad things happen: 1. $\|\theta\|$ goes to infinity; 2. There are infinite number of MLE's. To see this, note any step function (sigmoid with $\|\theta\| = \infty$) that is in the gap between the two classes is an MLE.

One way to avoid this is to incorporate a prior on θ in the form of a zero-mean Gaussian with covariance $\frac{1}{\lambda}I$,

$$\theta \sim \mathcal{N}\left(0, \frac{1}{\lambda}I\right), \tag{7}$$

[1]Strictly speaking, one needs only $K - 1$ parameter vectors.

and seek the MAP estimate. This is essentially smoothing, since large θ values will be penalized more. That is, we seek the θ that maximizes

$$\log p(\theta) + \sum_{i=1}^{n} \log p(y_i|x_i, \theta) \tag{8}$$

$$= -\frac{\lambda}{2}\|\theta\|^2 + \sum_{i=1}^{n} \log p(y_i|x_i, \theta) + constant \tag{9}$$

$$= -\frac{\lambda}{2}\|\theta\|^2 - \sum_{i=1}^{n} \log\left(1 + \exp(-y_i\theta^\top x_i)\right) + constant. \tag{10}$$

Equivalently, one minimizes the ℓ_2-regularized negative log likelihood loss

$$\min_\theta \frac{\lambda}{2}\|\theta\|^2 + \sum_{i=1}^{n} \log\left(1 + \exp(-y_i\theta^\top x_i)\right). \tag{11}$$

This is a convex function so there is a unique global minimum of θ. Unfortunately there is no closed form solution. One typically solves the optimization problem with Newton-Raphson iterations, also known as *iterative reweighted least squares* for logistic regression.

3 Stochastic Gradient Descent

In anticipation of more complex non-convex learners, we present a simple training algorithm that works for both linear regression (1) and logistic regression (11). Observing that both models can be written as follows:

$$\min_\theta \sum_{i=1}^{n} \ell(x_i, y_i, \theta) + \frac{\lambda}{2}\|\theta\|^2 \tag{12}$$

where $\ell(x_i, y_i, \theta)$ is the loss function.

The gradient descent algorithm starts at an arbitrary θ_0, and repeats the following update until convergence:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta\left(\sum_{i=1}^{n} \ell(x_i, y_i, \theta) + \frac{\lambda}{2}\|\theta\|^2\right) \tag{13}$$

$$= \theta_t - \sum_{i=1}^{n} \eta\nabla_\theta\ell(x_i, y_i, \theta) - \lambda\theta \tag{14}$$

where η is a small positive number known as the step size.

Each iteration of gradient descent needs to go over all n training points to compute the gradient, which may be too expensive. Note that we can write it equivalently as

$$\theta_{t+1} = \theta_t - n\eta\mathbb{E}\nabla_\theta\ell(x, y, \theta) - \eta\lambda\theta \tag{15}$$

where the expectation is with respect to the empirical distribution on the training set. Stochastic Gradient Descent (SGD) instead performs the following:

- At each iteration draw $(\tilde{x}, \tilde{y})$ from the empirical distribution, i.e. select a training point uniformly at random.
- Perform the update

$$\theta_{t+1} = \theta_t - n\eta\nabla_\theta\ell(\tilde{x}, \tilde{y}, \theta) - \eta\lambda\theta. \tag{16}$$

The update, in expectation, agrees with (15). Note the above may differ from other text by the coefficient n. This is an artifact of us not dividing the sum of losses by n, and amounts to a rescaling of λ.

For ridge regression,

$$\ell(x, y, \theta) = (x^\top \theta - y)^2.$$

Therefore,

$$\nabla_\theta \ell(x, y, \theta) = 2(x^\top \theta - y)x.$$

For logistic regression,

$$\ell(x, y, \theta) = \log\left(1 + \exp(-y\theta^\top x)\right).$$

Therefore,

$$\nabla_\theta \ell(x, y, \theta) = \frac{1}{1 + \exp(-y\theta^\top x)} \exp(-y\theta^\top x)(-yx).$$

SGD can be further stablized by:

1. Instead of taking the last iteration's parameter vector θ_T, take the average of all parameters over time:

$$\bar{\theta} = \frac{1}{T} \sum_{t=1}^{T} \theta_t.$$

   You will see that, for reasonably large T the average parameter achieves an MSE objective that is very close to the minimum. It will still fluctuate a little as one should expect, but overall it's a lot more stable.

2. Alternatively (or in conjunction), you may try to diminish the step size η as you go. One common scheme is to let the stepwise in iteration t to be

$$\eta_t = \eta_0 / \sqrt{t}$$

   where η_0 is the original stepsize. As $t = 1, 2, \ldots$ increases, η decreases. This will also stabilize SGD.

Both methods are motivated by theoretical analysis of SGD.