Machine learning Notes

MinSeok Song

This personal note is based on the lecture note of Stanford CS229 taught by Andrew Ng. The objective is to summarize the note in a condensed manner. I also try to work out any nontrivial result whose proof is omitted in the note or record my intuition to understand certain concept.

Least square mean algorithm

- Gradient descent algorithm has two kinds; batch gradient descent($\theta_j \to \theta_j + \alpha \sum_i (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$) and stochastic gradient descent($\theta_j \to \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$) for $i = 1$ to $i = m$.
- When m is large, single step is costly for batch gradient descent, which is why stochastic gradient descent(also known as incremental gradient descent) is preferred.
- (page 9) Some facts about matrix derivatives.
  - $\nabla_A trABA^TC = \nabla_X trXBA^TC + \nabla_X trABX^TC = CAB + C^TAB^T$
  -$\nabla_A|A| = |A|(A^{-1})^T$

Proof. This is equivalent to saying that $\nabla_A|A|_{ij} = |A|(A^{-1})^T_{ij}$ Note that $A^{-1} = \frac{1}{det(A)} \cdot adj(A)$, so $A^{-T}_{ij} = A^{-1}_{ji} = \frac{1}{det(A)} \cdot adj(A)_{ji} = \frac{1}{det(A)} \cdot cof(A, i, j)$. Now use the definition of determinant. □

The rest section just derives the closed form for the least square covariate estimates by using derivatives. This result is same as maximizing likelihood function of θ. Further, we can use so called locally weighted linear regression algorithm. This is to add weight for each error squares, i.e.,

$$\sum_i w^{(i)}(y^{(i)} - \theta^T x^{(i)})^2$$

. One standard choice is to use

$$w^{(i)} = exp(-\frac{(x^i - x)^2}{2\tau^2})$$

This gives higher weight on the training examples close to the query point. This is non-parametric algorithm because we must have entire training set in order to perform prediction.

- Related fact: if $Y \ N(X\beta, \Sigma)$ where $\Sigma = diag(\sigma_1^2, \dots, \sigma_n^2)$, WLS with $w_i \propto 1/\sigma_i^2$ is the best linear unbiased estimator (BLUE) of β.

Classification and logistic regression

When $y \in \{0, 1\}$ (discrete), we call it a classification problem. We use $h_\theta(x) = g(\theta^T x)$ for a predictor, and the task is to find an appropriate θ. For logistic regression problem, we assume that $E(y \mid x; \theta) = h_\theta(x)$

Definition 1. $g(z) = \frac{1}{1+e^{-z}}$ *is called the logistic function or the sigmoid function.*

In order to find θ, we use gradient ascent method to find the maximum value of likelihood function. The likelihood function is given by

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

and the gradient ascent rule is given by

$$\theta_j \to \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

for one training example case.

Newton's method: $\theta \to \theta - \frac{l'(\theta)}{l''(\theta)}$ For multidimensional case, $\theta \to \theta - H^{-1}\nabla_\theta l(\theta)$, where H is a Hessian of likelihood function.

If we let g to be a threshold function and use the same update rule for θ, then we have the **perceptron learning algorithm**.

Newton's method is generally faster than gradient descent method when n is not too large, since even though each iteration is more expensive, it converges much faster. In the context of logistic regression, it is also called **Fisher scoring**.

Definition 2. *Exponential family is any density function expressed as*

$$p(y; \eta) = b(y) \exp \eta^T T(y) - a(\eta)$$

where η is called the natural parameter, $T(y)$ is called sufficient statistic, and $a(\eta)$ is called log partition function (this ensures that p is density function).

- More generalized version is given by

$$p(y; \eta, \tau) = b(a, \tau) \exp((\eta^T T(y) - a(\eta))/c(\tau))$$

We have a "recipe" for constructing models in which y(given x and θ) comes from exponential family distribution.

We first list three assumptions for GLM.
1. $y \mid x; \theta$ follows exponential family distribution of η.
2. $E(T(y) \mid x; \theta) = h_\theta(x)(= g(\eta))$.
3. $\eta = \theta^T x$.

- OLS
$$h_\theta(x) = E(y \mid x;\theta) = \mu = \eta = \theta^T x$$
- LR
$$h_\theta(x) = E(y \mid x;\theta) = \phi = 1/(1+e^{-\eta}) = 1/(1+e^{-\theta^T x})$$

  This explains why logistic function is "natural" because it arises from Bernoulli. Note how η (used in exponential family distribution) is a bridge between ϕ (parameter of density) and θ, x (parameter in regression). This gives the idea of the following terminology.

Definition 3. *$g(\eta)$ is called* ***canonical response function*** *and g^{-1} is called* ***canonical link function****.*

(page 28) Identifying $T(y)$ is the key. In this case, $T(y) \in \mathbb{R}^{k-1}$ is given by a vector whose y'th element is one and otherwise zero.

We then get a link function $\phi \mapsto \{\log \frac{\phi_i}{\phi_k}\}_{i=1,\ldots,k}$ and a response function $\eta \mapsto \frac{e^{\eta_i}}{\sum_{j=1}^k e^{\eta_j}}$ (called **softmax function**).

(careful!) η is $n-1$ dimensional vector space. However, we may consider $\eta_k = 0$ because $\log(\phi_k/\phi_k) = 0$. This allows us to get a nice response function using upto η_k. We also define $\theta_k = 0 \in \mathbb{R}^{n+1}$.

We have $p(y = i \mid x;\theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^k e^{\theta_j^T x}}$. This model including θ and x as parameters is called **softmax regression**, a generalization of logistic regression (this is multinomial distribution. Clearly, when $k=2$, this reduces to Bernoulli).

We can now formulate log likelihood, and can maximize by using such methods as gradient ascent or Newton's method.

Generative Learning algorithms

Different modeling strategy for $p(y \mid x;\theta)$.

Definition 4. *An algorithm that tries to learn directly from $p(x \mid y)$ is called* ***discriminative learning algorithm****. An algorithm that learns from $p(x \mid y)$ and $p(y)$ is called* ***generative learning algorithm****.*

Think of it as distribution of animal's features (animal fixed) vs. predictive distribution of animals based on features. For generative learning algorithm, we can use Bayes rule given by

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}$$

when we calculate $\arg\max_y p(y \mid x)$ in order to predict y value.

(p4) Diagonal elements of the covariance matrix is indicated by the angle of contour (whether it be more or less than 45 degrees), mean value is indicated by the measurement of the center, and off-diagonal elements is indicated by the orientation (sign) of contour. (it would be illuminating to think how the variance of x/y direction does not change when we move two points in a certain way)

- GDA model:

  $y \sim Bernoulli(\phi)$
  $x \mid y = 0 \sim N(\mu_0, \Sigma)$
  $x \mid y = 1 \sim N(\mu_1, \Sigma)$

■ Relationship between GDA and logistic regression. If $p(x \mid y)$ is a multivariate Gaussian density function with shared Σ, $p(y \mid x)$ follows a logistic function with respect to some parameter.

  The converse does not hold. (page 7)

  Logistic regression is more robust.

  GDA is asymptotically better (in the sense that there is no model to predict y better) if the model is correct. For incorrect modeling assumptions, logistic regression is generally better.

- (p10) Naive Bayes assumption: x_i's are conditionally independent given y. This is a very strong asssumption but works well in most problems. We use it when we have discrete values for x. When x is continuous, we can discretize it. We may use this mthod over GDA.

- Laplace smoothing: for binomial case, take $\phi_j = p(z) = \frac{\sum_{i=1}^m 1\{z^{(i)} = j\}}{m} \rightarrow \frac{\sum_{i=1}^m 1\{z^{(i)} = j\} + 1}{m + k}$ where z takes values in $1, \ldots, k$. since we do not want probability being zero by looking at the finite amount of training sets in general.

- Taking $p(x_i = 0 \text{ or } 1 \mid y)$ is called **multi-variate Bernoulli event model**. When x_i takes values from 1 to $|V|$ (index of i'th word in the dictionary, instead of whether a particular word in the dictionary is included in the email or not), we call this **multinomial event model**. This is even better and easy to implement.

SVM

We used logistic function for g in logistic regression. For SVM, we don't use a intermediate step and set the range of g as $\{-1, 1\}$.

- **functional margin of (w,b) with respect to a training example** is $\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$

  Positive margin indicates correctness of the prediction and the scale represents confidence.

For a linear classifier: $g(w^Tx+b) = g(2w^Tx+2b)$ so it's probably a good idea to normalize by $|w|$

Definition 5. *function margin of (w,b) with respect to $S = \{(x^{(i)}, y^{(i)}); i = 1, \dots, m)\}$ is the smallest functional margins of the individual training examples.*

Question. Why is w orthogonal to the decision boundary?

If we glide through the boundary, b is fixed, so we have to have a fixed w^Tx as x changes. This w has to be the direction to $+1$ because the sign of w^Tx needs to be invariant under scaling of x.

- **Geometric margin** with respect to a training example is given by

  $$\gamma^{(i)} = y^{(i)}((\frac{w}{\|w\|})^T x^{(i)} + \frac{b}{\|w\|})$$

  Note that w and b are invariant under scaling.

- **The optimal margin classifier**

  we solve

  $$\max_{\gamma,w,b} \frac{\hat{\gamma}}{\|w\|} \text{ s.t. } y^{(i)}(w^Tx^{(i)} + b) \geqslant \hat{\gamma}$$

  that is,

  $$\min_{\gamma,w,b} \frac{1}{2}\|w\|^2 \text{ s.t. } y^{(i)}(w^Tx^{(i)} + b) \geqslant 1, i = 1, \dots, m$$

  This satisfies KKT condition due to the comment below.

- Question. why do we normalize in terms of w? why can't we just use functional margins?

  Answer: well, we need to restrict our coefficient in some capacity. Indeed, we can scale w and b at the same time however we want. There is no contradiction between functional and geometric margins because we are essentially fixing w and b at the same time so scaling them does not change our confidence of a training set in a way.

- Define $\theta_P(w) = \max_{\alpha,\beta:\alpha_i \geqslant 0} f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$. $\min_w \theta_P(w)$ has the same solution as the above primal problem. Note that θ finite $\rightarrow$ constraint is satisfied $\rightarrow \theta_P(w) = f(w)$ and on the other hand, constraint is satisfied $\rightarrow \theta_P(w) = f(w)$.

- Dual problem is when minimum and maximum are exchanged.

  Question. why max min$\leqslant$min max?

  This is because max min$\leqslant$ max for every w.

- **primal** optimization problem:

  $\min_w f(w)$
  s.t. $g_i(w) \leqslant 0, i = 1, \ldots, k$
  $h_i(w) = 0, i = 1, \ldots, l$

- KKT conditions:

$$\frac{\partial}{\partial w_i} L(w^*, \alpha^*, \beta^*) = 0$$
$$\frac{\partial}{\partial \beta_i} L(w^*, \alpha^*, \beta^*) = 0$$
$$\alpha_i^* g_i(w^*) = 0$$
$$g_i(w^*) \leqslant 0$$
$$\alpha^* \geqslant 0$$

- If there exist w^*, α^*, and β^* that satisfy KKT conditions, we have a solution to the optimization problem and $d^* : \max_{\alpha \geqslant 0, \beta} \min_w = p^* : \min_w \max_{alpha \geqslant 0, \beta}$. (d for dual)

- Suppose f and g_i's are convex, and h_i's are affine. Further, there exists w so that $g_i(w) \leqslant 0$ for all i. Then it is shown that there must exist solution for the optimal problem. These condition satisfies KKT conditions.

  Question. how do we check the above fact about KKT condition, and further, why do we have that $\alpha_i > 0$ only for the training examples that have function margin exactly equal to one? (page 10)

- $L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$

  Now dual problem is reduced to

$$\max_\alpha W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} > .$$

  s.t.

$$\alpha_i \geqslant 0, i = 1, \ldots, m$$
$$\sum_{i=1}^m \alpha_i y^{(i)} = 0$$

- We can express w and thus $w^T x + b$ only in terms of inner product between x and the points in the training set.

  Inner product will be useful when we do kernel trick.

  (p13, ?) why does (11), the expression for b, hold true? α's will all be zero except for the support vectors $\rightarrow$ why?

Definition 6. *Given a feature mapping ϕ, we define the corresponding Kernel to be*

$$K(x, z) = \phi(x)^T \phi(z)$$

.

- We have an efficient algorithm to calculate $K(x, z)$ even when computing ϕ is expensive.

  Definition 7. $K(x, z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$ *is called the Gaussian kernel.*

  This comes from the intuition that the original definition measures the similarity between x and z in a way.

- This is indeed a special case when kernel corresponds to an infinite dimensional feature mapping. Mercer theorem explains when we have a valid Kernel in general(p18, p15-17 for examples).

  'l_1 regularization for SVM makes the algorithm less sensitive to outliers. This works for non-linearly separable datasets.

- (p20) α_i and r_i both correspond to α before and this is why both are $\geqslant 0$.

- what is changed?

  1. $0 \leqslant \alpha_i \rightarrow 0 \leqslant \alpha_i \leqslant C$
  2. calculation for b^* has to be modified.

- So how do we solve dual problem? **SMO (sequential minimal optimization) algorithm** Since the constraint demands that $\sum_{i=1}^m \alpha_i y^{(i)} = 0$, we are fixing $m-2$ *alpha's* (versus **coordinate ascent algorithm** where we fix $m-1$ α's) and express α_1 in terms of α_2. Then we now are reduced to optimizing quadratic function of α_2's.

  Two things more to consider.

  1. How to determine α_i and α_j?
  2. How do we update b?

Learning Theory

- bias-variance trade-off. Essentially, bias (of $\hat{\beta}^{-j}$) increases when we under-fit (model too small), and variance (of μ_x and of $\hat{\beta}_k$ for $k \neq j$ when in reality $\beta_j \neq 0$) increases when we over-fit (model too large): lec(7a, check). The definition can be different for non-linear case.

  Lemma 1. ***(Chernoff bound)*** *As m gets large, probability of ϕ being far from $\phi = (1/m)\sum_{i=1}^m Z_i$ is small.*

Definition 8. ***Training error*** *(empirical risk or empirical error) is defined as*

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}$$

where S is a training set of size m.

generalization error *is defined as* $\epsilon(h) = P_{(x,y)\sim D}(h(x) \neq y)$

We assume that we drew training set from the same distribution D over and over. This is referred to as PAC(probably approximately correct) assumption.

Definition 9. *Picking* $\hat{\theta} = \arg\min_{\theta} \hat{\epsilon}(h_{\theta})$ *is called* ***empirical risk minimization (ERM)****.*

This is the most basic learning algorithm. (so, instead of the sum of number of non-equal cases, we could have done otherwise)

- We have $\hat{} = \arg\min_{h\in H} \hat{\epsilon}(h)$ where $H = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geqslant 0\}, \theta \in \mathbb{R}^{n+1}\}$, a set of linear classifiers over the domain of the inputs.