Machine Learning Lecture Notes

Haoyu Zhen

May 6, 2022

Contents

Preface

Acknowledgement

These Notes contain material developed and copyright by:

- *CS229, Machine Learning*, © 2020 Summer by Andrew Ng, Stanford. Course website:
- *Understanding Machine Learning*, © 2014 by Shai Shalev-Shwartz and Shai Ben-David.
- *CS189, Introduction to Machine Learning*, © 2021 by Marvin Zhang, UCB.
- *The Elements of Statistical Learning*, © 2017 by Trevor Hastie, Robert Tibshirani and Jerome Friedman.
- *AI-2611, Machine Learning*, © 2022 by Bingbing Ni, Shanghai Jiao Tong University.

Miscellaneous

Follow me or Contact me:

- Email: anye[underscore]zhen[at]sjtu[dot]edu[dot]cn.
- Github: anyeZHY.

1 Foundations

1.1 Model evaluation:

Hold-out, cross validation and bootstrap.
For cross validation, we often let the numbers of the folds be 10. And in bootstrap, the equation $\lim_{n\to\infty}(1-1/m)^m = 1/e$ is used to analyse the probality.

1.2 Performance

Definition 1.1 (Sensitivity and FPR). Now we consider that

	prediction+	prediction-
Actual	1	0
1	TP	FP
0	FN	TN

$$TPR = \frac{TP}{TP+FN},\ FPR = \frac{FP}{TN+FP}.$$

Remark 1.1. **ROC** space and **AUC** is also useful to select models.

Definition 1.2 (Precision and recall).

$$precision = \frac{TP}{TP+FP},\ recall = \frac{TP}{TP+FN}.$$

$$F_\beta = \frac{(1+\beta^2)\times P\times R}{\beta^2\times P+R}.$$

β depends on the preference of Precision and Recll.

1.3 Bias-Variance Decomposition

Theorem 1.1.

$$\begin{aligned} E(f;D) &= bias^2(x) + var(x) + \varepsilon^2 \\ &= (\bar{f}(x)-y)^2 + \mathbb{E}_D[f(x;D)-\bar{f}(x)] + \mathbb{E}_D[(y_D-y)^2] \end{aligned}$$

2 Regression

2.1 Linear Regression

The hypothesis class of linear regression predictors is simply the set of linear functions,

$$\mathcal{H}_{reg} = \left\{ \boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\}.$$

Intuitively,

$$\mathcal{L}_{\mathcal{S}}(h) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}) - \boldsymbol{y})^2, \forall h \in \mathcal{H}_{reg}.$$

To minimize the loss function, we need to solve $A\boldsymbol{w} = \boldsymbol{b}$ where $A \xlongequal{def} \sum \boldsymbol{x}_i \boldsymbol{x}_i^T = XX^T$ and $\boldsymbol{b} \xlongequal{def} \sum y_i \boldsymbol{x}_i = X^T \boldsymbol{y}$. If A is invertible then the solution is $w = A^{-1}\boldsymbol{b}$.

Theorem 2.1.

$$\omega = (X^T X)^{-1} X^T \boldsymbol{y}.$$

If the training instances do not span the entire space of $\mathbb{R}^d$ then A is not invertible.

Theorem 2.2. Using A's eigenvalue decomposition,we could write A as VD^+V^T where D is a diagnonal matrix and V is an orthonormal matrix. Define D^+ to be the diagonal matrix such that $D^+_{i,i} = 0$ if $D_{i,i} = 0$ otherwise $D^+_{i,i} = 1/D_{i,i}$. Then,

$$A\hat{\boldsymbol{w}} = \boldsymbol{b}$$

where $\hat{\boldsymbol{w}} = VD^+V^T\boldsymbol{b}$

Proof.

$$A\hat{\omega} = AA^+\boldsymbol{b} = VDV^TVD^+V^T\boldsymbol{b} = VDD^+V^T\boldsymbol{b} = \sum_{i:D_{i,i} \neq 0} \boldsymbol{v}_i \boldsymbol{v}_i^T \boldsymbol{b}.$$

That is, $A\hat{\omega}$ is the projection of b onto the span of those vectors v_i for which $D_{i,i} \neq 0$. Since the linear span of $x_1, \cdots, x_m$ is the same as the linear span of those v_i, and b is in the linear span of the x_i, we obtain that $A\hat{w} = b$, which concludes our argument. □

Remark 2.1. Indeed we always use the **Gradient Descent** method to optimize the loss function.

Linear regression for polynomial regression tasks $\mathcal{H}^n_{poly} = \{x \mapsto p(x)\}$ where $\psi(x) = (1, x, x^2, \cdots, x^n)$ and $p(\psi(x)) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$.

2.2 Ridge Regression

To ameliorate the effect of the invertible matrix, we could introduce the regularization.

Definition 2.1 (Ridge Regularized Loss).

$$R(w) = \lambda\|w\|^2.$$

Now the loss function reads:

$$\mathcal{L} = \mathcal{L}_{\mathcal{S}}(w) + R(w) = \frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x}) - \boldsymbol{y})^2 + \lambda\|w\|^2.$$

Hence, the solution to ridge regression becomes

$$\boldsymbol{w} = (2\lambda m I + A)^{-1}.$$

Theorem 2.3 (The stability of regularization). Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times [-1 \times 1]$, where $\mathcal{X} = \left\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| \leq 1\right\}$. Let $\mathcal{H} = \left\{\boldsymbol{w} \in \mathbb{R}^d : \|\boldsymbol{w}\| \leq B\right\}$. For any $\varepsilon \in (0, 1)$, let $m \geq 150B^2/\varepsilon^2$. Then applying the ridge regression algorithm with parameter $\lambda = \varepsilon/3B^2$ satisfies

$$\mathbb{E}_{S\sim\mathcal{D}^m}[L_D(A(S))] \leq \min_{\boldsymbol{w}\in\mathcal{H}} L(D) + \varepsilon.$$

2.3 Lasso Regression

Definition 2.2 (Lasso Regularized Loss).

$$R(w) = \lambda\|w\|_1^2.$$

Under some assumptions on the distribution and the regularization parameter λ, the LASSO will find sparse solutions

2.4 Logistic Regression

The hypothesis class is:

$$H_{sig} = \left\{x \mapsto \text{sigmoid}(\boldsymbol{w}\boldsymbol{x}) : \boldsymbol{w} \in \mathbb{R}^d\right\}$$

where $\text{sigmoid}(s) = 1/[1 + \exp(-s)]$. The loss function is

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \log\left[1 + \exp(-y_i \boldsymbol{w}\boldsymbol{x}_i)\right].$$

Remark 2.2. Optimization in logistic regression

- The advantage of the logistic loss function is that it is a convex function with respect to $\boldsymbol{w}$.
- No close form solution.
- Identical to the problem of finding a Maximum Likelihood Estimator.

3 Generalized Linear Models

3.1 The Exponential Family

Definition 3.1. We say that a class of distributions is in the exponential family if it can be written in the form

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

Here, η is called the **natural parameter** (also called the canonical parameter) of the distribution; $T(y)$ is the **sufficient statistic** (for the distributions we consider, it will often be the case that $T(y) = y$); and $a(\eta)$ is the log **partition function**. The quantity $e^{-a(\eta)}$ essentially plays the role of a normalization constant, that makes sure the distribution $p(y;\eta)$ sums/integrates over y to 1.

3.2 Constructing GLMs

1. $y \mid x;\theta \sim \text{ExponentialFamily}(\eta)$. I.e., given x and θ, the distribution of y follows some exponential family distribution, with parameter η.

2. Given x, our goal is to predict the expected value of T(y) given x. In most of our examples, we will have $T(y) = y$, so this means we would like the prediction $h(x)$ output by our learned hypothesis h to satisfy $h(x) = \mathbb{E}[y|x]$. (Note that this assumption is satisfied in the choices for $h_\theta(x)$ for both logistic regression and linear regression. For instance, in logistic regression, we had $h_\theta(x) = p(y = 1|x;\theta) = 0 \cdot p(y = 0|x;\theta) + 1 \cdot p(y = 1|x;\theta) = E[y|x;\theta]$.)

3. he natural parameter η and the inputs x are related linearly: $\eta = \theta^T x$. (Or, if η is vector-valued, then $\eta_i = \theta_i^T x$.)

Example 3.1 (Logistic Rrgression)**.** Note that: $y|x;\theta \sim \text{Bernoulli}(\phi)$. Then we have $\mathbb{E}[y|x;\theta] = \phi$. Thus

$$h_\theta(x) = \mathbb{E}[y|x;\theta] = \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}.$$

If we have a training set of n examples$\{(x^i, y^i); i = 1, \cdots, n\}$ and would like to learn the parameters θ_i of this model, we would begin by writing down the log-likelihood

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p(y^i|x^i;\theta) = \sum_{i=1}^{n} \log\left[\left(\frac{1}{1+e^{-\theta^T x}}\right)^{1\{y^i=1\}}\left(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}\right)^{1\{y^i=0\}}\right].$$

4 Kernel Method

Now we will introduce a function $\phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^p$ mapping the attributes to the features.

4.1 LMS with Features

Suppose that $\theta = \sum_{i=1}^n \beta_i x^i$. By updating rules of gradient descent,

$$\begin{aligned}\theta &:= \theta + \alpha \sum_{i=1}^n \left[y^i - \theta^T \phi(x^i)\right] \phi(x^i) \\ &= \sum_{i=1}^n \underbrace{\left\{\beta_i + \alpha\left[y^i - \theta^T \phi(x^i)\right]\right\}}_{\text{new } \beta} \phi(x^i)\end{aligned}$$

Then $i \in \{1, \cdots, n\}$:

$$\beta_i := \beta_i + \alpha \left[y^i - \sum_{j=1}^n \beta_j \phi(x^j)^T \phi(x^i)\right] = \beta_i + \alpha \left[y^i - \sum_{j=1}^n \beta_j K(\phi(x^j), \phi(x^i))\right]$$

where

$$K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle.$$

Remark 4.1. **Kernel** is a corresponding to the feature map ϕ as a function that maps $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

4.2 Properties of Kernels

Definition 4.1 (Gaussian kernel)**.**

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

The gaussian kernel is corresponding to an **infinite** dimensional feature mapping ϕ. Also, ϕ lives in Hilbert space.

Theorem 4.1. The corresponding kernel matrix $K \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite.

Theorem 4.2 (Mercer Theorem)**.** Let $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ be given. Then for K ti be a valide Mercer Kernel, it is necessary and sufficient that for any $\{x^1, \cdots, x^n\}, (n < \infty)$, the correspibonding kernel matrix is symmetic positive semidefinite. Nota Bene: the generalized form involve L^2 functions.

5 Support Vector Machines

SVMs are among the best (and many believe are indeed the best) off-the-shelf supervised learning algorithms. So, be **self-motivated** in this section.

5.1 Hard-SVM

Hard-SVM is the learning rule in which we return an ERM hyperplane that separates the training set with the largest possible margin. The Hard-SVM rule is

$$\underset{(w,b):\|w\|=1}{\arg\max} \min_{i\in[m]} \left|w^T x^i + b\right| \quad \text{s.t. } \forall i, y^i\left(w^T x^i + b\right) \geq 1.$$

Equivalently,

$$\underset{(w,b):\|w\|=1}{\arg\max} \min_{i\in[m]} y^i\left(w^T x^i + b\right) \tag{5.1}$$

Next, we give another equivalent formulation of the Hard-SVM rule as a quadratic optimization problem.[1]

Input: $(x^1,^1), \cdots, (x^m, y^m)$
Solve:

$$(w_0, b_0) = \underset{(w,b)}{\arg\min} \frac{1}{2}\|w\|^2 \quad \text{s.t. } \forall i, y^i\left(w^T x^i + b\right) \geq 1. \tag{5.2}$$

Output: $\hat{w} = w_0/\|w_0\|$, $\hat{b} = b_0/\|w_0\|$

Lemma 5.1. The output of Hard-SVM is a solution of Equation (5.1).

Proof. Let (w_1, b_1) be a solution of Equation (5.1) and $\gamma_1 = \min_{i\in[m]} y_i(w_1^T x^i + b_1)$. Then we have

$$y^i\left(\frac{w_1}{\gamma_1}^T x^i + \frac{b_1}{\gamma_1}\right) \geq 1.$$

Hence $\|w_0\| \leq \|\, w_1/\gamma_1 \,\| = 1/\gamma^*$. It follows that for all i,

$$y^i(\hat{w}^T x^i + \hat{b}) \geq \frac{1}{\|w_0\|} \geq \gamma_1.$$

Since $\|\hat{w}\| = 1$ we obtain that $(\hat{w}, \hat{b})$ is an optimal solution of Equation (5.1). □

5.1.1 The Sample Complexity of Hard-SVM*

Definition 5.1 (Separability)**.** Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$. We say that $\mathcal{D}$ is separable with a (γ, ρ)-margin if there exists (w^*, b^*) such that $\|w^*\| = 1$ and such that with probability 1 over the choice of $(x, y) \sim \mathcal{D}$ we have that $y({w^*}^T x + b^*) \geq \gamma$ and $\|x\| \leq \rho$.

[1] A quadratic optimization problem is an optimization problem in which the objective is a convex quadratic function and the constraints are linear inequalities.

Theorem 5.1. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$ that satisfies the (γ, ρ)-separability with margin assumption using a homogenous halfspace. Then, with probability of at least $1-\delta$ over the choice of a training set of size m, the 0-1 error of the output of Hard-SVM is at most

$$\sqrt{\frac{4(\rho)/\gamma^2}{m}} + \sqrt{\frac{2\log(2/\delta)}{m}}.$$

5.2 Soft-SVM and Norm Regularization

Input: $(x^1, ^1), \cdots, (x^m, y^m)$
Parameter: $\lambda > 0$
Solve:

$$\begin{aligned} &\min_{w,b,\xi} \left(\lambda \|w\|^2 + \frac{1}{m}\sum_{i=1}^m \xi_i\right) \\ &\text{s.t. } \forall i,\ y^i\left(w^T x^i + b\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \end{aligned}.$$

Output: w, b

Definition 5.2 (hinge loss).

$$l^{\text{hinge}}((w,b),(x,y)) = \max\left\{0, 1 - yw^T x + b\right\}.$$

Now we just need to optimize $\lambda\|w\|^2 + \mathcal{L}^{\text{hinge}}(w,b)$.

5.3 Duality

The Lagrangian for EQ.5.2 is:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i \left[y^{(i)}\left(w^T x^{(i)} + b\right) - 1\right].$$

By the fact that $\nabla \mathcal{L} = 0$

$$w = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)} \quad \text{and} \quad \sum_{i=1}^n \alpha_i y^{(i)} = 0.$$

Plug them back we obtain

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & \alpha_i \geq 0 \\ & \sum_{i=1}^n \alpha_i y^{(i)} = 0 \end{aligned}.$$

Note that

$$b = -\frac{\max_{y^{(i)}=-1} w^T x^{(i)} + \min_{y^{(i)}=1} w^T x^{(i)}}{2}.$$

5.4 SMO

Consider the dual form of Soft-SVM

$$\begin{aligned}\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0\end{aligned}.$$

The dual-complementrity conditions are

$$\begin{cases}\alpha_i = 0 & \implies y^{(i)}(w^T x^{(i)} + b) \geq 1 \\ \alpha_i = C & \implies y^{(i)}(w^T x^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C & \implies y^{(i)}(w^T x^{(i)} + b) = 1\end{cases}.$$

Now we introduce SMO (sequential minimal optimization). Repeat till convergence:

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

2. Reoptimize $W(\alpha)$ with respect to i and α_j, while holding all the other α_k's ($k \neq i, j$) fixed.

5.5 Implementing Soft-SVM Using SGD

Algorithm 1 SGD for Solving Soft-SVM

$\boldsymbol{\theta} = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
  $w^{(t)} = 1/\lambda t \times \boldsymbol{\theta}$
  Choose i uniformly at random for $[m]$
  if $y_i w^T x^i < 1$ **then**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^i x^i$
  end if
end for
return $\sum_{t=1}^{T} w^{(t)}/T$

6 Dimensionality Reduction

Dimensionality reduction is the process of taking data in a high dimensional space and mapping it into a new space whose dimensionality is much smaller.

6.1 Principal Component Analysis

In PCA, we have a compressing matrix $W \in \mathbb{R}^{n,d}$ and a recovering matrix $U \in \mathbb{R}^{d,n}$.For given data $x_1, x_2, \cdots, x_m$, we aim at solving the problem:

$$\underset{W,U}{\arg\min} \sum_{i=1}^{n} \|x_i - UWx_i\|^2 \tag{6.1}$$

Lemma 6.1. Let (U, W) be a solution of Equation 6.1. Then $U^T U = I$ and $W = U^T$. (The columns of U are orthonormal.)

Proof. Let $R = \left\{UWx : x \in \mathbb{R}^d\right\}$ which is an n dimensional linear subspace of $\mathbb{R}^d$. Let $\in \mathbb{R}^{n,d}$ be a matrix satisfies the range of V is R and $V^T V = I$. Then $\|x - Vy\|^2 = \|x\|^2 + \|y\|^2 - 2y^T V^T x$. Minimizing this w.r.t. y gives that $y = V^T x$. □

By the fact that

$$\left\|x - UU^T x\right\|^2 = \|x\|^2 - \operatorname{trace}(U^T x x^T U).$$

We could rewrite Equation 6.1 as follows:

$$\underset{U \in \mathbb{R}^{d,n}: U^T U = I}{\arg\max} \operatorname{trace}\left[U^T \left(\sum_{i=1}^{m} x_i x_i^T\right) U\right].$$

Theorem 6.1. Let $x_1, \cdots, x_m$ be arbitrary vectors in $\mathbb{R}^d$, let $A = \sum_{i=1}^m x_i x_i^T$, and let $u1, \cdots, u_n$ be n eigenvectors of the matrix A corresponding to the largest n eigenvalues of A. Then, the solution to the PCA optimization problem given in Equation 6.1 is to set U to be the matrix whose columns are $u_1, \cdots, u_n$ and to set $W = U^T$.

Proof. Let VDV^T be the spectral decomposition of A (suppose that $D_{1,1} \geq \cdots \geq D_{d,d}$) and let $B = V^T U$. We have

$$\operatorname{trace}\bigl(U^T A U\bigr) = \operatorname{trace}\bigl(B^T D B\bigr) = \sum_{j=1}^{d} D_{j,j} \sum_{i=1}^{n} B_{j,i}^2 \leq \max_{\boldsymbol{\beta} \in [0,1]^d : \|\boldsymbol{\beta}\| \leq n} \sum_{j=1}^{d} D_{j,j} \beta_j = \sum_{j=1}^{n} D_{j,j}.$$

Nota Bene: $B^T B = I$ which entails $\sum_{j=1}^d \sum_{i=1}^n B_{j,i}^2 = n$. □

6.2 Implementation

Algorithm 2 PCA algorithm

Input: A matrix of m examples $X \in R^{m,d}$ and number of components n.

```
if m > d then
    A = X^T X
    Let u_1, ..., u_n be the eigenvetors of A with largest eigrnvalues
else
    B = X X^T
    Let v_1, ..., v_n be the eigenvetors of B with largest eigrnvalues
    ∀i, u_i = X^T v_i / ‖X^T v_i‖
end if
return u_1, ..., u_n
```

The algorithm use a more efficient method when $d > m$. The complexity is $\mathcal{O}(m^2 d)$ under this case.