STAT 535

11/3/22

Lecture 11

Training predictors

Poll

all material < today (up to l10) ← Q2 - Thu 11/10

HW4 posted

L IV - 1 posted

Lecture IV: Training predictors, Part I

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2022

Analytic optimization

Optimization generalities

- Optimization glossary
- Descent methods zoo

Descent methods

- The steepest descent method
- Line minimization algorithms
- The Newton-Raphson method
- Examples: Logistic regression and Backpropagation

Reading HTF Ch.: Lasso 3.1,2,4, Logistic regression 4.4, Neural networks 11, Murphy Ch.: Ridge regression (including numerics) 7.5, Descent methods 8.3.2,3,5, Neural networks 16.5.1–4, Autoencoders 28, Bach Ch.: 5.

For more mathematical background, look e.g. at "Numerical recipes" chapter 10 or, for really advanced treatment Nocedal and Wrigth (Ch 3, 6).

The typical training problem

↳ find $f \in \mathcal{F}$ that "fits data" "predicts well"

- L = loss fction

Given $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$ data set (and implicitly a prediction task)

- $\mathcal{F}$ class of predictors (e.g Linear $\{f(x) = \beta^T x\}$, quadratic $f(x) = x^T A x + \beta^T x + \gamma$, neural net)

and $J(f; \mathcal{D})$ **objective function** ← optimization vocabulary

find $f = \underset{\mathcal{F}}{\operatorname{argmin}}\, J(f; \mathcal{D})$

Remarks

- Typically objective functions J

$$1)\quad J(f; \mathcal{D}) = \hat{L}(f) \quad \text{empirical loss} = \text{best } f \text{ for } \mathcal{D} \tag{1}$$

increase bias, reduce var

OR

$$2)\quad J_\lambda(f; \mathcal{D}) = \hat{L}(f) + \lambda R(f) \quad \text{regularized objective} \tag{2}$$

$R(f)$ is called regularization (functional) and $\lambda \geq 0$ is the **regularization constant (or parameter)**
↳ Bias - Var tradeoff
λ is fixed in the minimization of J. Setting λ can be thought as a form of model selection; as we shall see later λ plays the role of a smoothing parameter, or hyperparameter

- This lecture: algorithms for minimizing J's.

"Model class" = $\mathcal{F}, \lambda$ ← chosen at model selection time

Exceptions : no J

- k- nearest neighbors
- Kernel prediction

CART : max "purity" → greedy algorithm

Training vs Model Selection

given $\mathcal{D}$

model class 1 $\quad \mathcal{F}_1$ (linear) $\rightarrow f^{(1)} \in \mathcal{F}_1$

— " — 2 $\quad \mathcal{F}_2$ (CART) $\rightarrow f^{(2)} \in \mathcal{F}_2$

3 $\quad \mathcal{F}_3$ kernel h Gaussian $\rightarrow f^{(3)} \in \mathcal{F}_3$

$\mathcal{F}_4 \quad h' \dashrightarrow f^{(4)} \in \mathcal{F}_4$

5 $\quad \mathcal{F}$ (linear, R, λ) $\rightarrow f^{(5)}_{(\lambda)} \in \mathcal{F}$

$\mathcal{F}$ (— " —, R, λ') $\rightarrow f^{(6)}_{(\lambda')} \in \mathcal{F}$

. . . . .

e.g. Lasso with different λ

OR Ridge R, Lasso, OLS

$\Downarrow$

same $\mathcal{F}$

different $\hat{f}$

Model Selection

choose 1 predictor

Training

Linear Least Squares regression

- Problem $\mathcal{F} = \{f(x) = \beta^T x\}$, $L_{LS}(y, f(x)) = (y - f(x))^2$, $y \in \mathbb{R}$, $J = \hat{L}$
- Solution
  - Finding $f \in \mathcal{F}$ is equivalent to finding $\beta \in \mathbb{R}^d$ (or $\mathbb{R}^{d+1}$)

$\mathcal{D} = \{(x^i, y^i)\}_{i=1:n}$

Linear Least Squares regression

- Problem $\mathcal{F} = \{f(x) = \beta^T x\}$, $L_{LS}(y, f(x)) = (y - f(x))^2$, $y \in \mathbb{R}$, $J = \hat{L}$
- Solution
  - Finding $f \in \mathcal{F}$ is equivalent to finding $\beta \in \mathbb{R}^d$ (or $\mathbb{R}^{d+1}$)
  - define **data matrix** or (transpose) **design matrix**

(design)

$$\mathrm{X} = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \dots \\ (x^i)^T \\ \dots \\ (x^n)^T \end{bmatrix} \in \mathbb{R}^{n \times d} \quad \text{and} \quad Y = \begin{bmatrix} y^1 \\ y^2 \\ \dots \\ y^n \end{bmatrix}, \quad E = \begin{bmatrix} \varepsilon^1 \\ \varepsilon^2 \\ \dots \\ \varepsilon^n \end{bmatrix} \in \mathbb{R}^d$$

  - Then we can write

$$Y = X\beta + E$$

OLS (Ordinary Linear Regression)

  - The solution $\hat{\beta}$ is chosen to minimize the sum of the squared errors

$$J(\beta) = \hat{L}_{LS} = \sum_{i=1}^{n} (y^i - \beta^T x_i)^2 = ||E||^2 \tag{3}$$

which gives Exercise Derive it

$$\boxed{\hat{\beta} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T y} \tag{4}$$

- closed form
- exact minimization

(attains global min)

Linear Least Squares regression

- Problem $\mathcal{F} = \{f(x) = \beta^T x\}$, $L_{LS}(y, f(x)) = (y - f(x))^2$, $y \in \mathbb{R}$, $J = \hat{L}$
- Solution
  - Finding $f \in \mathcal{F}$ is equivalent to finding $\beta \in \mathbb{R}^d$ (or $\mathbb{R}^{d+1}$)
  - define **data matrix** or (transpose) **design matrix**

$$X = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \cdots \\ (x^i)^T \\ \cdots \\ (x^n)^T \end{bmatrix} \in \mathbb{R}^{N \times n} \quad \text{and} \quad Y = \begin{bmatrix} y^1 \\ y^2 \\ \cdots \\ y^n \end{bmatrix}, \quad E = \begin{bmatrix} \varepsilon^1 \\ \varepsilon^2 \\ \cdots \\ \varepsilon^n \end{bmatrix} \in \mathbb{R}^d$$

  - Then we can write

$$Y = X\beta + E$$

$\max_\beta$ Likelihood(β) $\Longleftrightarrow$ $\min_\beta - \ln$ (Likelihood(β))

  - The solution $\hat{\beta}$ is chosen to minimize the sum of the squared errors

$$J(\beta) = \hat{L}_{LS} = \sum_{i=1}^{n} (y^i - \beta^T x_i)^2 = ||E||^2 \tag{3}$$

which gives Exercise Derive it

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{4}$$

- Underlying statistical model $y = \beta^T x + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$ (and $(x^{1:n}, y^{1:n})$ sampled i.i.d., of course)
  - Define the negative log-likelihood $L_{logl} = -\ln P(y|x, \beta)$ → L_{logl} $\Longleftrightarrow$ MaxLikelihood
  - Then, $(y - \beta^T x)^2 = \sum_{i=1}^n (\varepsilon^i)^2 = -2\sigma^2 \ln P(y^i | x^i, \beta) = -2\sigma^2 \hat{L}_{logl}$
  - Hence, $\hat{\beta}$ from (4) is the Maximum Likelihood (ML) estimator of the parameter β and the minimizer of $\hat{L}_{logl}$
- This is an example where the minimizer of $J(f)$ has an analytical formula. A few more examples of this kind follow.

Regularized Linear Regression

$\beta \in \mathbb{R}^d$ $\quad \|\beta\| \equiv \|\beta\|_2 = \sqrt{\sum_{j=1}^{d} \beta_j^2}$

$\|\beta\|_1 = \sum_{j=1}^{d} |\beta_j|$

- L2 regularized (linear) LS regression (**Ridge regression**)

closed form
global optimum

$$J(f;\mathcal{D}) \ = \overbrace{\sum_{i=1}^{n}(y^i - \beta^T x_i)^2}^{LS} + \lambda\overbrace{||\beta||^2}^{R_2(\beta)} \tag{5}$$

Solution $\hat{\beta} = (X^T X + \lambda I_n)^{-1} X^T Y$ Exercise Derive it.

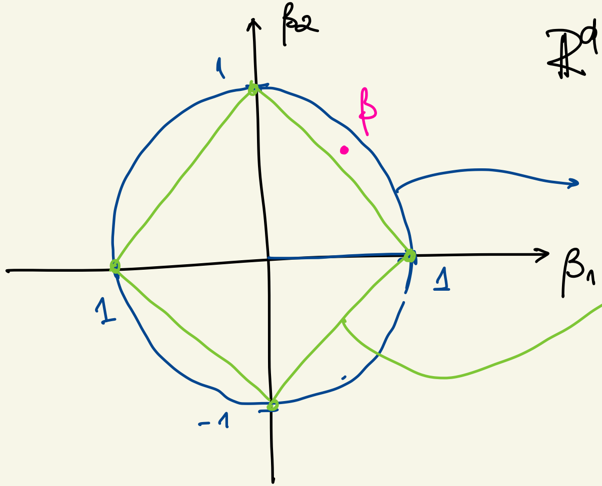
Ex: $\|\hat{\beta}_{OLS}\|_2 > \|\hat{\beta}_{Ridge}\|_2$

- L1 regularized (linear) regression (**LASSO**)

$$J(f;\mathcal{D}) \ = \underbrace{\sum_{i=1}^{n}(y^i - \beta^T x_i)^2}_{LS} + \lambda\overbrace{||\beta||_1}^{R_1(\beta)} \tag{6}$$

By contrast with the previos problem(s), this one does not have an analytic solution.

- solved numerically (= iterative algorithm)
- global opt.



$\mathbb{R}^d$

$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$

R_1 vs R_2

$\|\beta\|_2 = 1 \iff R_2(\beta) = 1$

$\|\beta\|_1 = 1$

$\|$

$|\beta_1| + |\beta_2|$

$\|\beta\|_2 < 1$

$\|\beta\|_1 > 1 > \|\beta\|_2$

$X^T y = b$

$A = X^T X$

$\beta_{OLS} = A^{-1} b$

$\beta_R = (A + \lambda I)^{-1} b$

$\|\beta\|_2^2 = \beta^T \beta$

Generative models for classification

- If each class conditional distribution $P_{X|Y=y}$ can be estimated by an analytic formula, then the generative classifier can be estimated analytically
- Examples LDA, QDA

Example (Naive Bayes for text classification)

The **bag of words** model of text. Let $D = \{\text{words}\}$ be a dictonary. For simplicity, we shall assume $D = \{$ and, average, batting, score, variance$\}$. We are given a **corpus** $\mathcal{D}$ containing N documents of two classes $\{$ sports$= -1$, statistics$= 1\}$. For each document in $\mathcal{D}$, we form a vector $x \in \{0,1\}^{|D|}$ by setting $x_w = 1$ if the document contains word w and 0 otherwise. For example, the document "Reddick's batting average is 0.50" has $x = [0\,1\,1\,0\,xs0]$. The x vectors are the data to which we fit a Naive Bayes model. Assume our toy corpus is now the table on the left, and the estimated class conditional distributions are on the right.

x					y
0	1	1	0	0	-1
1	1	0	1	0	-1
1	0	1	1	0	-1
1	0	1	0	0	-1
1	0	1	0	0	-1
1	1	0	0	1	1
0	0	0	1	0	1
1	1	0	0	1	1
1	1	0	0	0	1
1	0	0	0	1	1

$P_{X_j=1\|Y=-1}$					y
.8	.4	.8	.4	0	-1
.8	.6	0	.2	.6	1

$P_Y(1) = 0.5 = p$

Example (Naive Bayes for text (continued))

Before we go further, we will "adjust" the 0 and 1 estimated probabilities for "variance" and "batting", setting them to $1/M$, respectively $1-1/M$ for some "large" $M=10$ Exercise Do you think this problem occurs often in real applications? Find a statistical framework to justify the adjustment. The probability of a new document, e.g $x=[10010]$ in class 1, respectively -1 is

$$
\begin{aligned}
P(x|y=-1) &= 0.8\times(1-0.4)\times(1-0.8)\times 0.4\times 0.9 && (7)\\
&= 0.8^{x_1}(1-0.8)^{1-x_1}\times 0.4^{x_2}(1-0.4)^{1-x_2}\ldots && (8)\\
P(x|y=1) &= 0.8\times(1-0.6)\times(1-0.1)\times 0.2\times(1-0.6) && (9)
\end{aligned}
$$

The NB classifier we obtain is

$$
\begin{aligned}
f(x) \;=\; & \underbrace{\ln\frac{p}{1-p}}_{1} + x_1\underbrace{\ln\frac{P_{and|1}}{P_{and|-1}}}_{1} + x_2\underbrace{\ln\frac{P_{average|1}}{P_{average|-1}}}_{2/3} + x_3\underbrace{\ln\frac{P_{batting|1}}{P_{batting|-1}}}_{+\infty} + x_4\underbrace{\ln\frac{P_{score|1}}{P_{score|-1}}}_{2.} + x_5\underbrace{\ln\frac{P_{variance|1}}{P_{variance|-1}}}_{0}\\
& +(1-x_1)\ln\frac{1-P_{and|1}}{1-P_{and|-1}} + (1-x_2)\ln\frac{1-P_{average|1}}{1-P_{average|-1}} + (1-x_3)\ln\frac{1-P_{batting|1}}{1-P_{batting|-1}} + (1-x_4)\ln
\end{aligned}
$$

which evaluates to $f([1\,0\,0\,1\,0]) = \ln 1 + \ln\frac{3}{2} + \ln\frac{2}{9} + \ln 2 + \ln\frac{9}{4} = 0.405 > 0$ Notes As you saw above, we are not required to include all possible words in the dictionary Exercise Find some reasons why. A common preprocessing step stemming which aims to map words in the same word family to a single w; e.g "batting" $\rightarrow$ "bat". Sometimes stop words like "the", "and" are removed.

Most predictors can't be estimated analytically

- Unfortunately, minimizing J analytically is possible only in a handful of examples.
- In all other cases, we find $f = \operatorname*{argmin}_{\mathcal{F}} J$ by numerical/iterative methods also called search (or training/learning, of course). For example
  - CART algorithm Exercise What J is the CART algorithm minimizing?
  - Perceptron algorithm (will be revisited this lecture)
- Therefore now we study generic algorithms for finding minima of functions of n variables. This is called **(numerical) optimization**.

Optimization glossary

Change notation

Pb: $\min_{x \in \mathcal{X}} f(x)$

parameters $\exists$

Change in notation!

Here, f is a function to be minimized, and x the variable in the domain of the function. In a learning task, f will be replaced by an objective J like $\hat{L}_{logl}$ and x by the parameters of the predictor, e.g. $w, \beta, \theta, \ldots$.

The methods in this lecture belong to the class of **unconstrained optimization** methods.
Problem Find $\min_x f(x)$ for $x \in \mathbb{R}^d$ or $x \in D$ the **domain** of f. We assume that f is a twice differentiable function with continuous second derivatives.
Notation The **gradient** of f is the column vector

$$\nabla_x f(x) \; = \; \left[\frac{\partial f}{\partial x_i}(x) \right]_{i=1}^{n} \equiv \frac{\partial f}{\partial x} \sim \mathbb{R}^d \tag{11}$$

and the **Hessian** of f is the square symmetric matrix of second partial derivatives of f

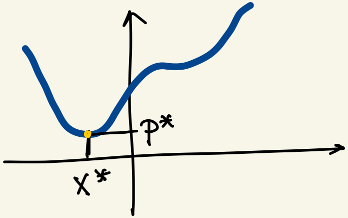
$$\nabla^2 f(x) \; = \; \left[\frac{\partial^2 f}{\partial x_i x_j}(x) \right]_{i,j=1}^{n} \in \mathbb{R}^{d \times d} \tag{12}$$

Assumed: symmetric

$$p^* = \min_{x \in \mathcal{X}} f(x)$$ min value

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

$$f(x^*) = p^*$$



Optimization basics

$A \in \mathbb{R}^{d\times d}$

- $A \succeq 0$ positive definite $\iff x^T A x \geq 0$ for all $x \in \mathbb{R}^d$

  $\iff \lambda(A) \geq 0$ for all e-values

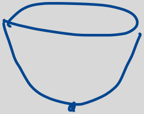
- $A \succ 0$ strictly p.d.

  $\iff x^T A x > 0$ for $x \neq 0$

  $\iff \lambda(A) > 0$ for all e-values

Ex: $\nabla^2 f(x^*) \succ 0 \iff$

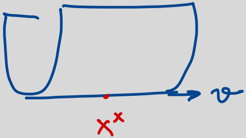
$-(x^*) = 0$

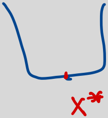


x^* isolated local opt & strict

Optimization basics

- $\nabla f(x^*) = 0$, $\nabla^2 f(x^*) \succeq 0 \Rightarrow \exists\ \lambda = 0$ e-value, v = e-vector
  $\not\succ 0$





$|x^3|$

Local and global minima

- A **local minimum** for f is point x^* for which

$$f(x^*) \leq f(x) \quad \text{whenever } ||x - x^*|| < \epsilon$$

- A **global minimum** for f is point x^* for which

$$f(x^*) \leq f(x) \quad \text{for all } x \text{ in the domain of } f$$

  We say x^* is a **strict local/global minimum** when the above inequalities are strict for $x \neq x^*$.
- A minimum is **isolated** if it is the only local minimum in an ϵ-ball around itself.
- A **stationary point** for f is a point x^* for which $\nabla f(x^*) = 0$.

Theorem

If f has continuous second derivative everywhere in $\overset{\mathcal{X}}{D}$, and $x^* \in \overset{\mathcal{X}}{D}$ is a point for which $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \geq 0$ ($\nabla^2 f(x^*) > 0$) then x^* is a (nonsingular) local minimum for f.

In what follows, we will deal only with non-singular local minima. A non-singular local minimum is strict and isolated.

Descent methods

Many unconstrained optimization methods for finding a local minimum are of the form:

$$x^{k+1} = x^k + \eta^k d^k \tag{13}$$

where $d^k \in \mathbb{R}^d$ represents an **(unnormalized) direction** and $\eta^k > 0$ is a scalar called the **step size**.

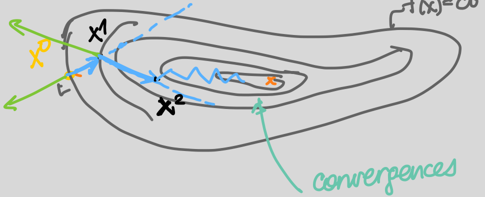
Direction choice

- gradient based $d^k = -D^k \nabla f(x^k)$ with $D^k \in \mathbb{R}^{n\times n}$
  - steepest descent $D^k = I$
  - stochastic gradient (more about it later)
  - Newton-Raphson $D^k = \nabla^2 f(x^k)^{-1}$
  - conjugate gradient – implicity multistep rescaling of the axes “equivalent” to $D^k = \nabla^2 f(x^k)^{-1}$
  - quasi-Newton – implicit multistep approximation of $D^k = \nabla^2 f(x^k)^{-1}$
- non-gradient based
  - coordinate descent d^k = one of the basis vectors in $\mathbb{R}^d$

Step size choice

- line minimization $\eta^k = \min_\eta f(x^k + \eta d^k)$
- Armijo rule (also called Backtracking) = search but not minimization
- constant step size $\eta^k = s$
- diminishing step size $\eta^k \to 0$; $\sum_k \eta^k = \infty$

Steepest descent



Algorithm STEEPEST-DESCENT

Input x^0 initial point f

For $k = 0, 1, \ldots$ ← iterate

1. calculate $d^k = \nabla f(x^k)$ ← steepest ascent
2. find η^k by line minimization
3. $x^{k+1} \to x^k - \eta^k d^k$ ← steepest descent

until stopping condition satisfied   step size (NOT too large!)

Output x^{k+1}

↳ $\approx x^*$ local opt

η^k · constant $\eta^k = \eta$ →

\ variable ← by Line Minimization

- golden ratio
- Armijo Rule