

Midterm Reviews (CS 229/ STATS 229)

Zhihan Xiong

Stanford University

15th May, 2020

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)

Optimization Methods

Gradient and Hessian (differentiable function $f : \mathbb{R}^d \mapsto \mathbb{R}$)

$$\nabla_x f = \left[\frac{\partial f}{\partial x_1} \quad \dots \quad \frac{\partial f}{\partial x_d} \right]^T \in \mathbb{R}^d \quad (\text{Gradient})$$

$$\nabla_x^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \dots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \in \mathbb{R}^{d \times d} \quad (\text{Hessian})$$

Gradient Descent and Newton's Method (objective function $J(\theta)$)

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_{\theta} J(\theta^{(t)}) \quad \swarrow \quad (\text{Gradient descent})$$

$$\theta^{(t+1)} = \theta^{(t)} - \left[\nabla_{\theta}^2 J(\theta^{(t)}) \right]^{-1} \nabla_{\theta} J(\theta^{(t)}) \quad \swarrow \quad (\text{Newton's method})$$

Least Square—Gradient Descent

- Model: $h_{\theta}(x) = \theta^T x$
- Training data: $\{(x^{(i)}, y^{(i)})\}_{i=1}^n, x^{(i)} \in \mathbb{R}^d$
- Loss: $J(\theta) = \frac{1}{2} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- Update rule:

$$\nabla_{\theta} (\theta^T x^{(i)} - y^{(i)})^2 = 2 (\theta^T x^{(i)} - y^{(i)}) x^{(i)}$$

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

Stochastic Gradient Descent (SGD)

Pick one data point $x^{(i)}$ and then update:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

Least Square—Closed Form

- Loss in matrix form: $J(\theta) = \frac{1}{2} \|X\theta - y\|_2^2$, where $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$
- Normal Equation (set gradient to 0): $\nabla_{\theta} J(\theta) = X^T(X\theta - y)$

$$X^T(X\theta^* - y) = 0$$

- Closed form solution:

$$\theta^* = (X^T X)^{-1} X^T y$$

Connection to Newton's Method

$$\theta^* = [\nabla_{\theta}^2 J]^{-1} \nabla_{\theta} J \Big|_{\theta = 0}$$

Newton's method is exact for 2nd-order objective.

Logistic Regression

A binary classification model and $y^{(i)} \in \{0, 1\}$

- Assumed model:

$$p(y | x; \theta) = \begin{cases} g_{\theta}(x) & \text{if } y = 1 \\ 1 - g_{\theta}(x) & \text{if } y = 0 \end{cases}, \quad \text{where } g_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$



- Log-likelihood function:**

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(y^{(i)} | x^{(i)}; \theta) \\ &= \sum_{i=1}^n \left[y^{(i)} \log g_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - g_{\theta}(x^{(i)})) \right] \end{aligned}$$

- Find parameters through **maximizing log-likelihood**, $\max_{\theta} \ell(\theta)$ (in Pset1).

The Exponential Family

Definition

$$\mathcal{N}(\mu, \sigma^2)$$

Probability distribution (with natural parameter η) whose density (or mass function) can be written into the following form

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

Example

Bernoulli distribution:

$$y \in \{0, 1\} \quad x > 0, \quad x = \exp(\log x)$$

$$p(y; \phi) = \phi^y (1 - \phi)^{1-y} = \exp\left(\left(\log\left(\frac{\phi}{1-\phi}\right)\right) y + \log(1 - \phi)\right)$$

$$\Rightarrow b(y) = 1, \quad T(y) = y, \quad \eta = \log\left(\frac{\phi}{1-\phi}\right), \quad a(\eta) = \log(1 + e^\eta)$$

The Exponential Family

More Examples

Categorical distribution, Poisson distribution, (Multivariate) normal distribution, etc.

Properties (In Pset1)

- $\mathbb{E}[T(Y); \eta] = \nabla_{\eta} a(\eta)$
- $\text{Var}(T(Y); \eta) = \nabla_{\eta}^2 a(\eta)$

Non-exponential Family Distribution

The Exponential Family

More Examples

Categorical distribution, Poisson distribution, (Multivariate) normal distribution, etc.

Properties (In Pset1)

- $\mathbb{E}[T(Y); \eta] = \nabla_{\eta} a(\eta)$
- $\text{Var}(T(Y); \eta) = \nabla_{\eta}^2 a(\eta)$

Non-exponential Family Distribution

Uniform distribution over interval $[a, b]$:

$$p(y; a, b) = \frac{1}{b-a} \cdot \mathbb{1}_{\{a \leq y \leq b\}} = \overset{b(y, \eta)}{\mathbb{1}_{\{a \leq y \leq b\}}} \exp\left(\log \frac{1}{b-a}\right)$$

Reason: $b(y)$ cannot depend on parameter η .

The Generalized Linear Model (GLM)

Components

- Assumed model: $p(y | x; \theta) \sim \text{ExponentialFamily}(\eta)$ with $\eta = \theta^T x$
- Predictor: $h(x) = \mathbb{E}[T(Y); \eta] = \nabla_{\eta} a(\eta)$.
- Fitting through maximum likelihood:

$$\max_{\theta} \ell(\theta) = \max_{\theta} \sum_{i=1}^n p(y^{(i)} | x^{(i)}; \eta) \quad \theta^T x^{(i)}$$

Examples

- GLM under Bernoulli distribution: Logistic regression
- GLM under Poisson distribution: Poisson regression (in Pset1)
- GLM under Normal distribution: Linear regression
- GLM under Categorical distribution: Softmax regression (multiclass logistic)

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - **Generative Algorithms**
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)

Gaussian Discriminant Analysis (GDA)

Generative Algorithm for Classification

- Learn $p(x | y)$ and $p(y)$
- Classify through Bayes rule: $\operatorname{argmax}_y p(y | x) = \operatorname{argmax}_y p(x | y) p(y)$

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

GDA Formulation

- Assume $p(x | y) \sim \mathcal{N}(\mu_y, \Sigma)$ for some $\mu_y \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$
- Estimate μ_y , Σ and $p(y)$ through maximum likelihood, which is

$$\max \sum_{i=1}^n \left[\log p(x^{(i)} | y^{(i)}) + \log p(y^{(i)}) \right]$$

$$p(y) = \frac{\sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}}}{n}, \mu_y = \frac{\sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}} x^{(i)}}{\sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}}}, \Sigma = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$$

Naive Bayes

Formulation $x \in \mathbb{R}^d$

- Assume $p(x | y) = \prod_{j=1}^d p(x_j | y)$
- Estimate $p(x_j | y)$ and $p(y)$ through maximum likelihood, which gives

$$p(x_j | y) = \frac{\sum_{i=1}^n \mathbb{1}_{\{x_j^{(i)}=x_j, y^{(i)}=y\}}}{\sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}}}, \quad p(y) = \frac{\sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}}}{n}$$

Laplace Smoothing

Assume x_j takes value in $\{1, 2, \dots, k\}$, the corresponding modified estimator is

$$p(x_j | y) = \frac{1 + \sum_{i=1}^n \mathbb{1}_{\{x_j^{(i)}=x_j, y^{(i)}=y\}}}{k + \sum_{i=1}^n \mathbb{1}_{\{y^{(i)}=y\}}}$$

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)

Kernel

Motivation

p >> d, usually

- Feature map: $\phi : \mathbb{R}^d \mapsto \mathbb{R}^p$
- Fitting linear model with gradient descent gives us $\theta = \sum_{i=1}^n \beta_i \phi(x^{(i)})$
- Predict a new example z : $h_\theta(z) = \sum_{i=1}^n \beta_i \phi(x^{(i)})^T \phi(z) = \sum_{i=1}^n \beta_i K(x^{(i)}, z)$

It brings nonlinearity without much sacrifice in efficiency as long as $K(\cdot, \cdot)$ can be computed efficiently.

Definition

$K(x, z) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is a valid kernel if there exists $\phi : \mathbb{R}^d \mapsto \mathbb{R}^p$ for some $p \geq 1$ such that $K(x, z) = \phi(x)^T \phi(z)$

Kernel (Continued)

Examples

- Polynomial kernels: $K(x, z) = (x^T z + c)^d$, $\forall c \geq 0$ and $d \in \mathbb{N}$
- Gaussian kernels: $K(x, z) = \exp\left(-\frac{\|x-z\|_2^2}{2\sigma^2}\right)$, $\forall \sigma^2 > 0 \leftarrow p = \infty$
- More in Pset2...

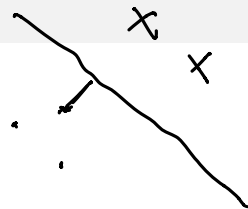
Theorem

$K(x, z)$ is a valid kernel if and only if for any set of $\{x^{(1)}, \dots, x^{(n)}\}$, its Gram matrix, defined as

$$G = \begin{bmatrix} K(x^{(1)}, x^{(1)}) & \dots & K(x^{(1)}, x^{(n)}) \\ \vdots & \ddots & \vdots \\ K(x^{(n)}, x^{(1)}) & \dots & K(x^{(n)}, x^{(n)}) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

is positive semi-definite.

SVM

at margin, $|w^T x^{(i)} + b| = 1$ 

$$\theta = (w, b)$$

Formulation ($y \in \{-1, 1\}$)

$$\begin{aligned} \min_{\{w, b\}} \quad & \frac{1}{2} \|w\|_2^2 \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad \forall i \in \{1, \dots, n\} \end{aligned} \quad (\text{Hard-SVM})$$

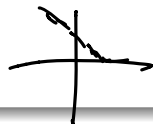
$$\begin{aligned} \min_{\{w, b, \xi\}} \quad & \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad \forall i \in \{1, \dots, n\} \\ & \xi_i \geq 0, \quad \forall i \in \{1, \dots, n\} \end{aligned} \quad (\text{Soft-SVM})$$

$\rightarrow \xi_i \geq 1 - y^{(i)}(w^T x^{(i)} + b)$

Properties

- The optimal solution has the form $w^* = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)}$ and thus can be kernelized.
- The soft-SVM can be treated as a minimization over hinge loss plus ℓ_2 regularization:

$$\ell(y) = \max\{0, 1 - x\}$$



$$\min_{\{w, b\}} \sum_{i=1}^n \max\{0, 1 - y^{(i)}(w^T x^{(i)} + b)\} + \lambda \|w\|_2^2$$

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)

Model Formulation

Multi-layer Fully-connected Neural Networks (with Activation Function f)

$$\begin{aligned}
 a^{[1]} &= f\left(W^{[1]}x + b^{[1]}\right) \\
 a^{[2]} &= f\left(W^{[2]}a^{[1]} + b^{[2]}\right) \\
 &\dots \\
 a^{[r-1]} &= f\left(W^{[r-1]}a^{[r-2]} + b^{[r-1]}\right) \\
 h_{\theta}(x) &= a^{[r]} = W^{[r]}a^{[r-1]} + b^{[r]}
 \end{aligned}$$

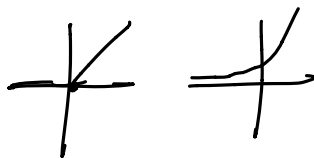
Handwritten notes:
 d, d_2
 r matrices, r vectors
 $d, d_1 + d, d_2 + \dots + d_{r-1}, d_r$
 $d_1 + d_2 + \dots + d_r$

Possible Activation Functions

• ReLU: $f(z) = \text{ReLU}(z) := \max\{z, 0\}$

• Sigmoid: $f(z) = \frac{1}{1+e^{-z}}$

• Hyperbolic Tangent: $f(z) = \tanh(z) := \frac{e^z - e^{-z}}{e^z + e^{-z}}$



Handwritten note: $f(z) = \log(1 + e^z)$

Backpropagation

$$J = \frac{1}{2} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Let J be the loss function and $z^{[k]} = W^{[k]}a^{[k-1]} + b^{[k]}$. By chain rule, we have

$$\frac{\partial J}{\partial W_{ij}^{[r]}} = \frac{\partial J}{\partial z_i^{[r]}} \frac{\partial z_i^{[r]}}{\partial W_{ij}^{[r]}} = \frac{\partial J}{\partial z_i^{[r]}} a_j^{[r-1]} \implies \frac{\partial J}{\partial W^{[r]}} = \frac{\partial J}{\partial z^{[r]}} a^{[r-1]T}, \quad \frac{\partial J}{\partial b^{[r]}} = \frac{\partial J}{\partial z^{[r]}}$$

$$\frac{\partial J}{\partial a_i^{[r-1]}} = \sum_{j=1}^{d_r} \frac{\partial J}{\partial z_j^{[r]}} \frac{\partial z_j^{[r]}}{\partial a_i^{[r-1]}} = \sum_{j=1}^{d_r} \frac{\partial J}{\partial z_j^{[r]}} W_{ji}^{[r]} \implies \frac{\partial J}{\partial a^{[r-1]}} = W^{[r]T} \frac{\partial J}{\partial z^{[r]}}$$

$$\frac{\partial J}{\partial z^{[r]}} := \delta^{[r]} \implies \frac{\partial J}{\partial z^{[r-1]}} = \left(W^{[r]T} \delta^{[r]} \right) \odot f'(z^{[r-1]}) := \delta^{[r-1]}$$

$$\implies \frac{\partial J}{\partial W^{[r-1]}} = \delta^{[r-1]} a^{[r-2]T}, \quad \frac{\partial J}{\partial b^{[r-1]}} = \delta^{[r-1]}$$

Continue for layers $r-2, \dots, 1$. $\frac{\partial J}{\partial z^{[r-2]}} = (W^{[r-1]T} \delta^{[r-1]}) \odot f'(z^{[r-2]}) := \delta^{[r-2]}$

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning**
- 4 Practice Exam Problem (If time permits)

k-means

Algorithm 1: k-means

Input: Training data $\{x^{(1)}, \dots, x^{(n)}\}$; number of clusters k

- 1 Initialize $c^{(1)}, \dots, c^{(k)} \in \mathbb{R}^d$ as clustering centers
 - 2 **while** *not converge* **do**
 - 3 Assign each $x^{(i)}$ to its ~~closest~~ ^{closest} clustering centers $c^{(j)}$
 - 4 Take the mean of each cluster as new clustering center
 - 5 **end**
-

Property

k-means tries to minimize the following loss function approximately:

$$\min_{\{c^{(1)}, \dots, c^{(k)}\}} \sum_{i=1}^n \|x^{(i)} - c^{(j(i))}\|_2^2, \quad \text{where } j(i) = \operatorname{argmin}_{j' \in \{1, \dots, k\}} \|x^{(i)} - c^{(j')}\|_2^2$$

However, it does not guarantee to find the global minimum.

Gaussian Mixture Model (GMM)

Formulation

Assume each data point $x^{(i)}$ is generated independently through the following procedure:

1. Sample $z^{(i)} \sim \text{Multinomial}(\phi)$, where $\sum_{j=1}^k \phi_j = 1$
2. Sample $x^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}})$

How to estimate parameters ϕ , $\{\mu_1, \dots, \mu_k\}$ and $\{\Sigma_1, \dots, \Sigma_k\}$ if $z^{(i)}$ cannot be observed?

Maximum Likelihood

$$\ell(\theta) = \sum_{i=1}^n \log \left(\sum_{j=1}^m \phi_j p(x^{(i)}; \mu_j, \Sigma_j) \right),$$

$$\text{where } p(x^{(i)}; \mu_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} \exp \left(-\frac{1}{2} (x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j) \right)$$

This is too complicated to optimize directly!

Expectation-Maximization (EM)

Jensen's Inequality

By Jensen's inequality, for any distribution Q_i over $\{1, \dots, k\}$, we have

$$\sum_{i=1}^n \log \left(\sum_{j=1}^m Q_i(j) \frac{p(x^{(i)}, z^{(i)} = j; \theta)}{Q_i(j)} \right) \geq \sum_{i=1}^n \sum_{j=1}^m Q_i(j) \log \frac{p(x^{(i)}, z^{(i)} = j; \theta)}{Q_i(j)} := \text{ELBO}(\theta)$$

Theorem

If we take

$Q_i(j) = p(z^{(i)} = j \mid x^{(i)}; \theta^{(t)})$ and let $\theta^{(t+1)} := \arg\max_{\theta} \text{ELBO}(\theta)$, we then have $\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)})$ (proved in lecture).

$\ell(\theta^{(t)})$ monotonic

$$\sum_{j=1}^k Q_i(j) = 1$$

Algorithm 2: EM Algorithm

Input: Training data $\{x^{(1)}, \dots, x^{(n)}\}$

- 1 Initialize $\theta^{(0)}$ by some random guess
- 2 **for** $t = 0, 1, 2, \dots$ **do**
- 3 Set $Q_i(j) = p(z^{(i)} = j \mid x^{(i)}; \theta^{(t)})$ for each i, j ; // E-step
- 4 Set $\theta^{(t+1)} = \arg\max_{\theta} \text{ELBO}(\theta)$; // M-step
- 5 **end**

EM in GMM

Posterior of $z^{(i)}$

$$p(z^{(i)} = j \mid x^{(i)}; \theta^{(t)}) = \frac{\phi_j^{(t)} p(x^{(i)}; \mu_j^{(t)}, \Sigma_j^{(t)})}{\sum_{j'=1}^k \phi_{j'}^{(t)} p(x^{(i)}; \mu_{j'}^{(t)}, \Sigma_{j'}^{(t)})}$$

GMM Update Rules

By defining $w_j^{(i)} = p(z^{(i)} = j \mid x^{(i)}; \theta^{(t)})$, we have

if $w_j^{(i)} = 1_{\{z^{(i)}=j\}}$

$$\phi_j^{(t+1)} = \frac{\sum_{i=1}^n w_j^{(i)}}{n}, \quad \mu_j^{(t+1)} = \frac{\sum_{i=1}^n w_j^{(i)} x^{(i)}}{\sum_{i=1}^n w_j^{(i)}}, \quad \forall j \in \{1, \dots, k\}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^n w_j^{(i)} (x^{(i)} - \mu_j^{(t+1)})(x^{(i)} - \mu_j^{(t+1)})^T}{\sum_{i=1}^n w_j^{(i)}}, \quad \forall j \in \{1, \dots, k\}$$

Outline

- 1 Supervised Learning
 - Discriminative Algorithms
 - Generative Algorithms
 - Kernel and SVM
- 2 Neural Networks
- 3 Unsupervised Learning
- 4 Practice Exam Problem (If time permits)