CS229 Section: Midterm Review

Nandita Bhaskhar

Content from past CS229 teams and ML Cheatsheets from Shervine & Afshine Amidi

October 22, 2021
Outline

1 Supervised Learning
2 Optimization
3 Linear Regression
4 Logistic Regression
5 Exponential Family
6 GLMs
7 Generative Algorithms
8 SVMs
9 NNs
Supervised Learning: Recap

- **Given**: a set of data points (or attributes) \( \{x^{(1)}, x^{(2)}, ..., x^{(m)}\} \) and their associated labels \( \{y^{(1)}, y^{(2)}, ..., y^{(m)}\} \)
- **Dimensions**: \( x \) usually \( d \)-dimensional \( \in \mathbb{R}^d \), \( y \) typically scalar
- **Goal**: build a model that predicts \( y \) from \( x \) for unseen \( x \)
Supervised Learning: Recap

Types of predictions

- y is continuous, real-valued: Regression
  - Example: Linear regression
- y is discrete classes: Classification
  - Example: Logistic regression, SVM, Naive Bayes
Supervised Learning: Recap

Types of models

- **Discriminative**
  - Directly estimate $p(y|x)$ by learning decision boundary
  - Example: Logistic regression, SVM

- **Generative**
  - Estimate $p(x|y)$ and infer $p(y|x)$ from it
  - Can generate new samples
  - Example: GDA, Naive Bayes
Notations and Concepts

- **Hypothesis**: Denoted by $h_\theta$. Given an input $x^{(i)}$, predicted output is $h_\theta(x^{(i)})$
- **Loss Function**: Function $L(z, y) : \mathbb{R} \times \mathbb{Y} \mapsto \mathbb{R}$ computes how different the predicted value $z$ and the ground truth label are

<table>
<thead>
<tr>
<th>Least squared error</th>
<th>Logistic loss</th>
<th>Hinge loss</th>
<th>Cross-entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2}(y - z)^2$</td>
<td>$\log(1 + \exp(-yz))$</td>
<td>$\max(0, 1 - yz)$</td>
<td>$-\left[ y \log(z) + (1 - y) \log(1 - z) \right]$</td>
</tr>
</tbody>
</table>

- **Linear regression**
- **Logistic regression**
- **SVM**
- **Neural Network**
Notations and Concepts

- **Cost function**: Function $J$ taking model parameters $\theta$ as input and giving a score to reflect how badly the model performs. Sum of loss over all predictions

  $$J(\theta) = \sum_{i=1}^{m} L(h_{\theta}(x^{(i)}), y^{(i)})$$

- **Likelihood**: Maximizing likelihood $L(\theta)$ corresponds to finding the "best" distribution of data given a set of parameters. We usually find the log likelihood $\ell(\theta) = \log L(\theta)$ and maximize it.

  $$\theta^* = \arg\max_{\theta} \ell(\theta)$$
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Optimization: Gradient Descent

- To find the optimal $\theta$ that minimizes the cost function $J(\theta)$, we can use gradient descent with a learning rate $\alpha \in \mathbb{R}$

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_{\theta} J(\theta^{(t)})$$

Stochastic Gradient Descent

- In Stochastic gradient descent (SGD), we update the parameter based on each training example, whereas in batch gradient descent we update based on a batch of training examples.
Optimization: Newton’s method

- Numerical method to estimate $\theta$ such that $J'(\theta)$ is 0
- We update $\theta$ as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{J'(\theta^{(t)})}{J''(\theta^{(t)})}$$

- For the multi-dimensional case:

$$\theta^{(t+1)} = \theta^{(t)} - \left[ \nabla^2_{\theta} J(\theta^{(t)}) \right]^{-1} \nabla_{\theta} J(\theta^{(t)})$$
Recap: Gradients and Hessians

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

$$
\nabla_x f = \begin{bmatrix}
\frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_d}
\end{bmatrix}^T \in \mathbb{R}^d
$$

$$
\nabla^2_x f = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_d^2}
\end{bmatrix} \in \mathbb{R}^{d \times d}
$$
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Linear Regression

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

\[
\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^n \left( h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}
\]

**Stochastic Gradient Descent (SGD)**
Pick one data point \( x^{(i)} \) and then update:

\[
\theta^{(t+1)} = \theta^{(t)} - \alpha \left( h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}
\]

Supervised Learning
Optimization
Linear Regression
Logistic Regression
Exponential Family
GLMs
Generative Algorithms
SVMs
NNs

CS229 Midterm Review Fall 2021
Nandita Bhaskhar 13 / 40
Solving Least Squares: 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):
  \[
  X^T (X\theta^* - y) = 0
  \]
- Closed form solution:
  \[
  \theta^* = \left(X^T X\right)^{-1} X^T y
  \]

Connection to Newton’s Method

\[
\theta^* = \left[\nabla^2_\theta J\right]^{-1} \nabla_\theta J, \quad \text{when the gradient is evaluated at } \theta = 0
\]

Newton’s method is exact with only one step iteration if we started from \( \theta^{(0)} = 0 \).
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Logistic Regression

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

- Assumed model:

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

- Log-likelihood function:

$$\ell(\theta) = \sum_{i=1}^{n} \log p(y^{(i)} \mid x^{(i)}; \theta)$$

$$= \sum_{i=1}^{n} \left[ y^{(i)} \log g_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - g_\theta(x^{(i)})) \right]$$

- Find parameters through maximizing log-likelihood, $\arg\max_\theta \ell(\theta)$ (in Pset1).
Sigmoid and Softmax

- **Sigmoid**: The sigmoid function (also known as logistic function) is given by:

\[
g(z) = \frac{1}{1 + e^{-z}}
\]

- **Softmax regression**: Also called as multi-class logistic regression, it generalizes logistic regression to multi-class cases.

\[
p(y = k|x; \theta) = \frac{\exp \theta_k^T x}{\sum_j \exp \theta_j^T x}
\]
Exponential Family

Definition
Probability distribution with **natural or canonical parameter** $\eta$, **sufficient statistic** $T(y)$ and a **log-partition** function $a(\eta)$ whose density (or mass function) can be written as

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

- Oftentimes, $T(y) = y$
- In many cases, $\exp (-a(\eta))$ can be considered as a normalization term that makes the probabilities sum to one
Common Exponential Distributions

Bernoulli distribution:

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

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

More examples:
Categorical distribution, Poisson distribution, Multivariate normal distribution, etc
## Common Exponential Distributions

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$\eta$</th>
<th>$T(y)$</th>
<th>$a(\eta)$</th>
<th>$b(y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bernoulli</td>
<td>$\log \left( \frac{\phi}{1-\phi} \right)$</td>
<td>$y$</td>
<td>$\log(1 + \exp(\eta))$</td>
<td>$1$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\mu$</td>
<td>$y$</td>
<td>$\frac{\eta^2}{2}$</td>
<td>$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)$</td>
</tr>
<tr>
<td>Poisson</td>
<td>$\log(\lambda)$</td>
<td>$y$</td>
<td>$e^\eta$</td>
<td>$\frac{1}{y!}$</td>
</tr>
<tr>
<td>Geometric</td>
<td>$\log(1 - \phi)$</td>
<td>$y$</td>
<td>$\log \left( \frac{e^\eta}{1-e^\eta} \right)$</td>
<td>$1$</td>
</tr>
</tbody>
</table>
Properties

- \( \mathbb{E} [ T (Y) ; \eta ] = \nabla_\eta a (\eta) \)

- \( \text{Var} ( T (Y) ; \eta ) = \nabla^2_\eta a (\eta) \)

**Non-exponential Family Distribution**

Uniform distribution over interval \([a, b]\):

\[
p(y; a, b) = \frac{1}{b - a} \cdot 1\{a \leq y \leq b\}
\]

Reason: \(b(y)\) cannot depend on parameter \(\eta\).
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Generalized Linear Model (GLM)

Generalized Linear Models (GLM) aim at predicting a random variable \( y \) as a function of \( x \) and rely on the following components:

**Assumed model:**

\[
p(y | x; \theta) \sim \text{ExponentialFamily}(\eta)
\]

- \( \eta = \theta^T x \)
- Prediction: \( h(x) = \mathbb{E}[T(Y); \eta] = \nabla_\eta a(\eta) \).
- Fitting through maximum likelihood:

\[
\max_{\theta} \ell(\theta) = \max_{\theta} \sum_{i=1}^{n} p(y^{(i)} | x^{(i)}; \eta)
\]
Generalized Linear Model (GLM)

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
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Gaussian Discriminant Analysis (GDA)

Generative Algorithm for Classification

- Learn $p(x \mid y)$ and $p(y)$
- Classify through Bayes rule: $\arg\max_y p(y \mid x) = \arg\max_y p(x \mid y) p(y)$

GDA Formulation

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

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

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

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

\[
p(x_j \mid y) = \frac{\sum_{i=1}^{n} 1\{x^{(i)}_j = x_j, y^{(i)} = y\}}{\sum_{i=1}^{n} 1\{y^{(i)} = y\}}, \quad p(y) = \frac{\sum_{i=1}^{n} 1\{y^{(i)} = y\}}{n}
\]

Laplace Smoothing
Assume \( x_j \) takes value in \( \{1, 2, \ldots, k\} \), the corresponding modified estimator is

\[
p(x_j \mid y) = \frac{1 + \sum_{i=1}^{n} 1\{x^{(i)}_j = x_j, y^{(i)} = y\}}{k + \sum_{i=1}^{n} 1\{y^{(i)} = y\}}
\]
Outline

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Kernel

- Core idea: reparametrize parameter $\theta$ as a linear combination of featurized vectors
- 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
Kernel

- Given a feature mapping \( \phi \), we define the kernel \( K \) as follows:

\[
K(x, z) = \phi(x)^T \phi(z)
\]

- "Kernel trick" to compute the cost function using the kernel because we actually don’t need to know the explicit mapping \( \phi \), which is often very complicated.

- Instead, only the values \( K(x, z) \) are needed.

- Suppose \( K(x^{(i)}, x^{(j)}) = K_{ij} \)

- If \( K = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \) then is \( K \) a valid kernel function?

- If \( K = \begin{bmatrix} 3 & 5 \\ 5 & 3 \end{bmatrix} \) then is \( K \) a valid kernel function?
Kernel

Theorem

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

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

is positive semi-definite.

Examples

- Polynomial kernels: $K(x, z) = (x^T z + c)^d$, $\forall \; c \geq 0$ and $d \in \mathbb{N}$
- Gaussian kernels: $K(x, z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$, $\forall \; \sigma^2 > 0$
Support Vector Machine (SVM)

**Goal**: find the line that maximizes the minimum distance to the line

The optimal margin classifier $h$ with $(y \in \{-1, 1\})$ is such that:

$$h(x) = \text{sign}(w^T x - b)$$

$$\min \{w, b\} \quad \frac{1}{2} \|w\|_2^2$$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad \forall \ i \in \{1, \ldots, n\}$

**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 $\ell_2$ regularization:

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

1. Supervised Learning
2. Optimization
3. Linear Regression
4. Logistic Regression
5. Exponential Family
6. GLMs
7. Generative Algorithms
8. SVMs
9. NNs
Neural Networks

By noting $i$ the $i^{th}$ layer of the network and $j$ the $j^{th}$ hidden unit of the layer, we have:

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

where we note $w$, $b$, $z$ the weight, bias and output respectively.
Neural Networks

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

$$a[1] = f \left( W[1]x + b[1] \right)$$


$$\ldots$$

$$a[r-1] = f \left( W[r-1]a[r-2] + b[r-1] \right)$$

$$h_\theta(x) = a[r] = W[r]a[r-1] + b[r]$$
# Activation Functions

<table>
<thead>
<tr>
<th>Sigmoid</th>
<th>Tanh</th>
<th>ReLU</th>
<th>Leaky ReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g(z) = \frac{1}{1 + e^{-z}}$</td>
<td>$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$</td>
<td>$g(z) = \max(0, z)$</td>
<td>$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$</td>
</tr>
</tbody>
</table>

![Activation Function Graphs](attachment:image)
Updating Weights

- Step 1: Take a batch of training data
- Step 2: Perform forward propagation to obtain the corresponding loss
- Step 3: Backpropagate the loss to get the gradients
- Step 4: Use the gradients to update the weights of the network
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 $r - 2, \ldots, 1$. 
Tips

- Practice, practice, practice
- For proofs, give reasoning and show how you go from one step to the next
- Prepare a cheat sheet – easy to run out of time in open book exams
- Pay attention to notation and indices. "Silly mistakes" can completely change the meaning of your reasoning
- Think in vector terms!

All the best :)

CS229 Midterm Review Fall 2021

Nandita Bhaskhar