Chapter 7
Learning Discriminative Models in General

supplementary slides to
Machine Learning Fundamentals
©**Hui Jiang 2020**
published by Cambridge University Press

August 2020

Outline

Revisit Soft SVMs

- review soft SVM formulation:

SVM3

$$\min_{\mathbf{w},b,\xi_i} \quad \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

subject to

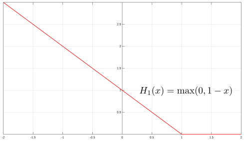
$$y_i(\mathbf{w}^\intercal\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i \in \{1,2,\cdots,N\}$$

$$\xi_i \geq 0 \qquad \forall i \in \{1,2,\cdots,N\}$$

$$\begin{cases} \xi_i \geq 1 - y_i(\mathbf{w}^\intercal\mathbf{x}_i + b) \\ \xi_i \geq 0 \end{cases}$$

$$\Longrightarrow \; \xi_i \geq \max\Big(0, 1 - y_i(\mathbf{w}^\intercal\mathbf{x}_i + b)\Big)$$

introduce the hinge function $H_1(\cdot)$:



$$\xi_i \geq H_1\Big(y_i(\mathbf{w}^\intercal\mathbf{x}_i + b)\Big)$$

- reformulate the objective function:

$$\xi_i^* = H_1\Big(y_i(\mathbf{w}^\intercal\mathbf{x}_i + b)\Big) \quad \forall i \in \{1,2,\cdots,N\}$$

Learning Discriminative Models

- soft SVMs can be reformulated as:

$$\min_{\mathbf{w},b} \left[\underbrace{\sum_{i=1}^{N} H_1\Big(y_i(\mathbf{w}^\intercal \mathbf{x}_i + b)\Big)}_{\text{empirical loss}} + \underbrace{\lambda \cdot \|\mathbf{w}\|^2}_{\text{regularization term}} \right]$$

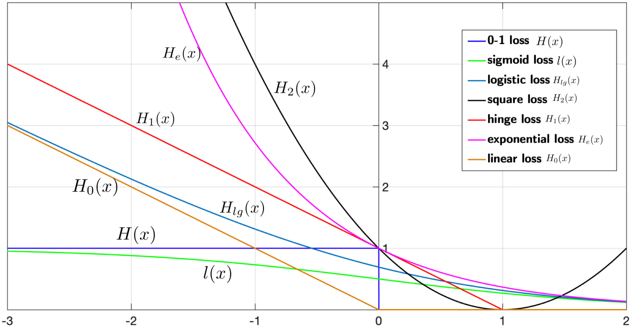
  - the empirical loss is summed over all training samples when evaluated using the **hinge loss** function
  - the **regularization term** is the L_2 norm of model parameters
- a general way to learn discriminative models is to minimize:

$$\text{empirical loss} + \text{regularization term}$$

Loss Functions in Machine Learning (1)

ML method	loss function
-	0-1 loss: $H(x) = \begin{cases} 1 & x \le 0 \\ 0 & x > 0 \end{cases}$
Perceptron	rectified linear loss: $H_0(x) = \max(0, -x)$
MCE	sigmoid loss: $l(x) = \frac{1}{1+e^x}$
Logistic Regression	logistic loss: $H_{\mathrm{lg}}(x) = \ln(1 + e^{-x})$
Linear Regression	square loss: $H_2(x) = (1-x)^2$
Soft SVM	hinge loss: $H_1(x) = \max(0, 1-x)$
Boosting	exponential loss: $H_e(x) = e^{-x}$

Loss Functions in Machine Learning (2)



1. monotone non-increasing
   - ex. *square loss*
2. convex
   - ex. *0-1 loss*
   - ex. *sigmoid loss*
3. sharply increasing as $x \to -\infty$
   - e.g. *square loss*
   - e.g. *exponential loss*

Regularization in Machine Learning

- soft SVMs can be formulated as the constrained optimization

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{N} H_1\Big(y_i(\mathbf{w}^\intercal \mathbf{x}_i + b)\Big)$$

subject to

$$\|\mathbf{w}\|^2 \leq 1$$

- regularization $\Longrightarrow$ constraining model space in learning
- L_p norm regularization $(\forall p \geq 0)$

$$\|\mathbf{w}\|_p \leq 1$$

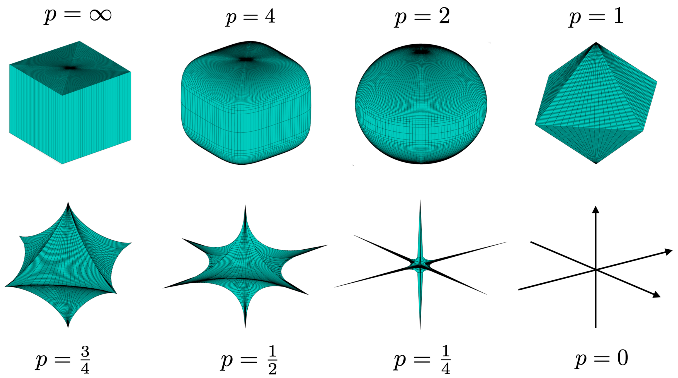
where

$$\|\mathbf{w}\|_p = \Big(|w_1|^p + |w_2|^p + \cdots + |w_n|^p\Big)^{\frac{1}{p}}$$

L_p norm (1)

- L_2 norm:
$$\|\mathbf{w}\|_2 = \sqrt{|w_1|^2 + \cdots + |w_n|^2}$$
- L_1 norm:
$$\|\mathbf{w}\|_1 = |w_1| + \cdots + |w_n|$$
- L_0 norm:
$$\|\mathbf{w}\|_0 = |w_1|^0 + \cdots + |w_n|^0$$
  - $\|\mathbf{w}\|_0$ ($\in \mathbb{Z}$) equals the number of non-zero elements in $\mathbf{w}$
- L_∞ norm:
$$\|\mathbf{w}\|_\infty = \max\Big(|w_1|, \cdots, |w_n|\Big)$$
  - $\|\mathbf{w}\|_\infty$ equals to the largest magnitude of all elements in $\mathbf{w}$

L_p norm (2)



- $p \downarrow \Longrightarrow$ stronger regularization
- $p \geq 1 \iff$ convex set
- $p = 1$ is important

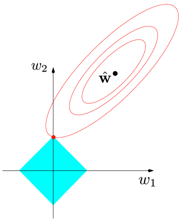
L_1 Norm Regularization Promotes Sparsity

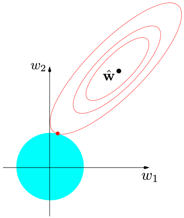
- L_1 norm leads to sparse solutions
- the gradient of L_1 norm is constant until the parameter becomes zero

$$\frac{\partial \|\mathbf{w}\|_1}{\partial w_i} = \mathsf{sgn}(w_i) = \begin{cases} 1 & w_i > 0 \\ 0 & w_i = 0 \\ -1 & w_i < 0 \end{cases}$$

- the gradient of L_2 norm shrinks as the parameter becomes small

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial w_i} = 2w_i$$





Ridge Regression

- ridge regression = linear regression + L_2 norm regularization
- given a training set as $\mathcal{D} = \big\{(\mathbf{x}_i, y_i) \mid i = 1, 2, \cdots, N\big\}$

$$\mathbf{w}^*_{\text{ridge}} = \arg\min_{\mathbf{w}} \left[\sum_{i=1}^{N} \big(\mathbf{w}^\intercal \mathbf{x}_i - y_i\big)^2 + \lambda \cdot \|\mathbf{w}\|_2^2 \right]$$

- the closed form solution:

$$\mathbf{w}^*_{\text{ridge}} = \Big(\mathbf{X}^\intercal \mathbf{X} + \lambda \cdot \mathbf{I}\Big)^{-1} \mathbf{X}^\intercal \mathbf{y}$$

- may use gradient descent to avoid the matrix inversion

LASSO

- LASSO = linear regression + L_1 norm regularization
- given a training set as $\mathcal{D} = \big\{(\mathbf{x}_i, y_i) \mid i = 1, 2, \cdots, N\big\}$

$$\mathbf{w}^*_{\text{lasso}} = \arg\min_{\mathbf{w}} \underbrace{\left[\frac{1}{2} \sum_{i=1}^{N} (\mathbf{w}^\intercal \mathbf{x}_i - y_i)^2 + \lambda \cdot \|\mathbf{w}\|_1 \right]}_{Q_{\text{lasso}}(\mathbf{w})}$$

- no closed-form solution exists, need to use gradient descent:

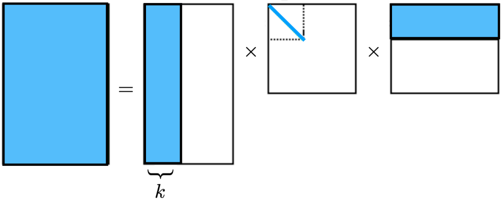
$$\frac{\partial Q_{\text{lasso}}(\mathbf{w})}{\partial \mathbf{w}} = \Big(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^\intercal \Big) \mathbf{w} - \sum_{i=1}^{N} y_i \mathbf{x}_i + \lambda \cdot \mathrm{sgn}(\mathbf{w})$$

- LASSO imposes stronger L_1 regularization and gives sparse solutions, suitable for feature selection and interpretation

Matrix Factorization (MF): SVD

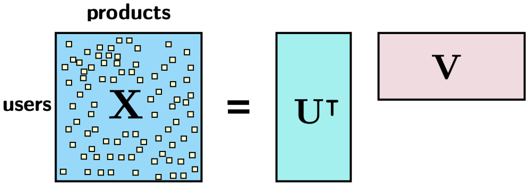
- use singular value decomposition (SVD) to factorize a matrix
- truncate to approximate: $\Big[\mathbf{X}\Big]_{n\times m} \approx \Big[\mathbf{U}\Big]_{n\times k}\Big[\mathbf{V}\Big]_{k\times m}$
- matrix factorization leads to many real-world applications

$$\Big[\mathbf{X}\Big]_{n\times m} = \Big[\mathbf{U}\Big]_{n\times m}\Big[\mathbf{\Sigma}\Big]_{m\times m}\Big[\mathbf{V}\Big]_{m\times m}$$



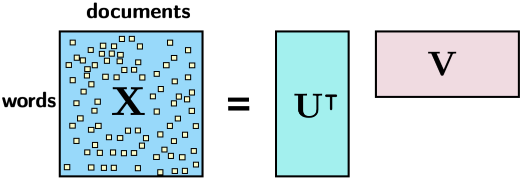
$$\Big[\mathbf{X}\Big]_{n\times m} \approx \Big[\mathbf{U}\Big]_{n\times k}\Big[\mathbf{\Sigma}\Big]_{k\times k}\Big[\mathbf{V}\Big]_{k\times m}$$

MF Application (1): Collaborative Filtering



- collaborative filtering is the key technique for recommendation
- rely on factorizing a sparse matrix into two small dense matrices
  - construct the so-call user-product matrix
  - yielding product vectors and user vectors
  - measure similarity between different products (or users)

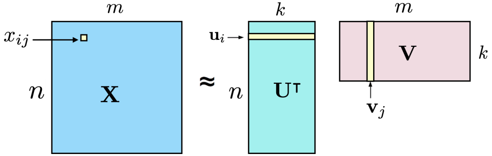
MF Application (2): Latent Semantic Analysis



- latent semantic analysis (LSA) is an important technique in natural language processing
- rely on factorizing a sparse matrix into two small dense matrices
  - construct the so-call document-word matrix
  - yielding word vectors and document vectors
  - measure similarity between different words (or documents)

Matrix Factorization as Machine Learning

- SVD is not efficient for large sparse matrices, not suitable for partially observed matrices
- cast matrix factorization as a machine learning problem



- learn $\mathbf{U}$ and $\mathbf{V}$ to minimize the reconstruction error for all observed elements in Ω and some regularization terms:

$$Q(\mathbf{U},\mathbf{V}) = \sum_{(i,j)\in\Omega} \left(x_{ij} - \mathbf{u}_i^\intercal \mathbf{v}_j\right)^2 + \lambda_1 \sum_{i=1}^{n} \|\mathbf{u}_i\|_2^2 + \lambda_2 \sum_{j=1}^{m} \|\mathbf{v}_j\|_2^2$$

Alternating Algorithm for Matrix Factorization

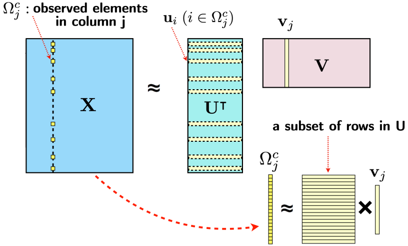
bilinear models suggest an alternating algorithm

1. keep $\mathbf{U}$ constant, estimate all $\mathbf{v}_j$ in $\mathbf{V}$

$$\mathbf{v}_j = \Big(\sum_{i\in\Omega_j^c} \mathbf{u}_i\mathbf{u}_i^\intercal + \lambda_2\,\mathbf{I}\Big)^{-1}\Big(\sum_{i\in\Omega_j^c} x_{ij}\mathbf{u}_i\Big)$$

2. keep $\mathbf{V}$ constant, estimate all $\mathbf{u}_j$ in $\mathbf{U}$

$$\mathbf{u}_i = \Big(\sum_{j\in\Omega_i^r} \mathbf{v}_j\mathbf{v}_j^\intercal + \lambda_1\,\mathbf{I}\Big)^{-1}\Big(\sum_{j\in\Omega_i^r} x_{ij}\mathbf{v}_i\Big)$$



$$\arg\min_{\mathbf{v}_j} \sum_{i\in\Omega_j^c} \big(x_{ij} - \mathbf{u}_i^\intercal\mathbf{v}_j\big)^2 + \lambda_2\cdot\|\mathbf{v}_j\|_2^2$$

Alternating Algorithm for Matrix Factorization

set $t = 0$
randomly initialize $\mathbf{v}_j^{(0)}$ $(j = 1, 2, \cdots, m)$
while not converged **do**
  for $i = 1, \cdots, n$ **do**

$$\mathbf{u}_i^{(t+1)} = \Big(\sum_{j \in \Omega_i^r} \mathbf{v}_j^{(t)} (\mathbf{v}_j^{(t)})^\intercal + \lambda_1 \, \mathbf{I} \Big)^{-1} \Big(\sum_{j \in \Omega_i^r} x_{ij} \mathbf{v}_i^{(t)} \Big)$$

  end for
  for $j = 1, \cdots, m$ **do**

$$\mathbf{v}_j^{(t+1)} = \Big(\sum_{i \in \Omega_j^c} \mathbf{u}_i^{(t+1)} (\mathbf{u}_i^{(t+1)})^\intercal + \lambda_2 \, \mathbf{I} \Big)^{-1} \Big(\sum_{i \in \Omega_j^c} x_{ij} \mathbf{u}_j^{(t+1)} \Big)$$

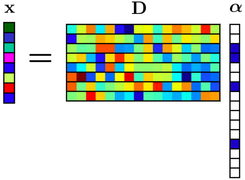
  end for
  $t = t + 1$
end while

- all updates in each step are parallelizable
- it may be faster to use SGD for very sparse matrices

Dictionary Learning (I)

- **sparse coding assumption**: each real sample is constructed from a large dictionary based on a sparse code

$$\mathbf{x} = \begin{bmatrix} | & & | \\ \mathbf{d}_1 & \cdots & \mathbf{d}_n \\ | & & | \end{bmatrix}_{d\times n} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{D}\,\boldsymbol{\alpha}$$



- **dictionary learning**: jointly learn the dictionary $\mathbf{D}$ and sparse codes from $\{\boldsymbol{\alpha}_i\}$ from some training samples $\{\mathbf{x}_i\}$

$$\mathbf{X} = \begin{bmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_N \\ | & & | \end{bmatrix}_{d\times N} \qquad \mathbf{A} = \begin{bmatrix} | & & | \\ \boldsymbol{\alpha}_1 & \cdots & \boldsymbol{\alpha}_N \\ | & & | \end{bmatrix}_{n\times N}$$

Dictionary Learning (II)

- dictionary learning is cast as a machine learning problem:
  - minimize the reconstruction error
  - add L_1 regularization to impose sparsity of all codes
  - add L_2 regularization on dictionary to avoid overfitting

$$\arg\min_{\mathbf{D},\mathbf{A}} \quad \underbrace{\frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{x}_i - \mathbf{D}\,\boldsymbol{\alpha}_i\right\|_2^2 + \lambda_1 \sum_{i=1}^{N}\left\|\boldsymbol{\alpha}_i\right\|_1 + \frac{\lambda_2}{2}\sum_{j=1}^{n}\left\|\mathbf{d}_j\right\|_2^2}_{Q(\mathbf{D},\mathbf{A})}$$

- compute all gradients:

$$\frac{\partial Q(\mathbf{D},\mathbf{A})}{\partial \mathbf{A}} = \mathbf{D}^\intercal\mathbf{D}\mathbf{A} - \mathbf{D}^\intercal\mathbf{X} + \lambda_1 \cdot \mathsf{sgn}\big(\mathbf{A}\big)$$

$$\frac{\partial Q(\mathbf{D},\mathbf{A})}{\partial \mathbf{D}} = \mathbf{D}\mathbf{A}\mathbf{A}^\intercal - \mathbf{X}\mathbf{A}^\intercal + \lambda_2 \cdot \mathbf{D}$$

Dictionary Learning (III)

Gradient Descent for Dictionary Learning

set $t = 0$ and η_0
randomly initialize $\mathbf{D}^{(0)}$ and $\mathbf{A}^{(0)}$
while not converged **do**

update $\mathbf{A}$:

$$\mathbf{A}^{(t+1)} = \mathbf{A}^{(t)} - \eta_t \Big(\big(\mathbf{D}^{(t)}\big)^\intercal \mathbf{D}^{(t)} \mathbf{A}^{(t)} - \big(\mathbf{D}^{(t)}\big)^\intercal \mathbf{X} + \lambda_1 \cdot \mathrm{sgn}\big(\mathbf{A}^{(t)}\big) \Big)$$

update $\mathbf{D}$:

$$\mathbf{D}^{(t+1)} = \mathbf{D}^{(t)} - \eta_t \Big(\mathbf{D}^{(t)} \mathbf{A}^{(t+1)} \big(\mathbf{A}^{(t+1)}\big)^\intercal - \mathbf{X} \big(\mathbf{A}^{(t+1)}\big)^\intercal + \lambda_2 \cdot \mathbf{D}^{(t)} \Big)$$

adjust $\eta_t \rightarrow \eta_{t+1}$
$t = t + 1$

end while

Sparse Coding

- sparse coding: given a dictionary $\mathbf{D}$, find the sparse code $\boldsymbol{\alpha}$ for a new observation $\mathbf{x}$
- ideal but infeasible:

$$\arg\min_{\boldsymbol{\alpha}} \quad \|\boldsymbol{\alpha}\|_0 \qquad \text{s. t.} \qquad \mathbf{D}\,\boldsymbol{\alpha} = \mathbf{x}$$

- a practical solution:
  - replace intractable L_0 norm with L_1
  - relax to an imperfect reconstruction

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \; \underbrace{\frac{1}{2}\left\|\mathbf{x} - \mathbf{D}\,\boldsymbol{\alpha}\right\|_2^2 + \lambda_1 \cdot \left\|\boldsymbol{\alpha}\right\|_1}_{Q'(\boldsymbol{\alpha})}$$

- use gradient descent:

$$\frac{\partial Q'(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \mathbf{D}^\intercal \mathbf{D}\,\boldsymbol{\alpha} - \mathbf{D}^\intercal \mathbf{x} + \lambda_1 \cdot \mathsf{sgn}(\boldsymbol{\alpha})$$