Chapter 7
Learning Discriminative Models in General

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

August 2020
Outline

1. A General Framework to Learn Discriminative Models
2. Ridge Regression and LASSO
3. Matrix Factorization
4. Dictionary Learning
Revisit Soft SVMs

- review soft SVM formulation:

\[
\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\
\text{subject to} \\
y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i \in \{1, 2, \cdots, N\} \\
\xi_i \geq 0 \quad \forall i \in \{1, 2, \cdots, N\}
\]

- reformulate the objective function:

\[
\xi_i^* = H_1(y_i(w^T x_i + b)) \quad \forall i \in \{1, 2, \cdots, N\}
\]

\[
\xi_i \geq H_1(y_i(w^T x_i + b))
\]
soft SVMs can be reformulated as:

$$\min_{w,b} \left[ \sum_{i=1}^{N} H_1 \left( y_i \left( w^T x_i + b \right) \right) + \lambda \cdot \|w\|^2 \right]$$

- 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:

**empirical loss** + **regularization term**
# Loss Functions in Machine Learning (1)

<table>
<thead>
<tr>
<th>ML method</th>
<th>loss function</th>
</tr>
</thead>
</table>
| -               | 0-1 loss: \( H(x) = \begin{cases} 
                        1 & x \leq 0 \\
                        0 & x > 0 
                      \end{cases} \) |
| Perceptron      | rectified linear loss: \( H_0(x) = \max(0, -x) \) |
| MCE             | sigmoid loss: \( l(x) = \frac{1}{1 + e^x} \)       |
| Logistic Regression | logistic loss: \( H_{lg}(x) = \ln(1 + e^{-x}) \) |
| 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

\[
\min_{w,b} \sum_{i=1}^{N} H_1\left(y_i\left(w^\top x_i + b\right)\right)
\]

subject to

\[\|w\|^2 \leq 1\]

- regularization \(\Rightarrow\) constraining model space in learning

- \(L_p\) norm regularization \((\forall p \geq 0)\)

\[\|w\|_p \leq 1\]

where

\[\|w\|_p = \left(|w_1|^p + |w_2|^p + \cdots + |w_n|^p\right)^{\frac{1}{p}}\]
\( L_p \) 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 \( \mathbf{w} \)

- **\( L_\infty \) norm:**
  \[
  \| \mathbf{w} \|_\infty = \max \left( |w_1|, \cdots, |w_n| \right)
  \]
  \( \| \mathbf{w} \|_\infty \) equals to the largest magnitude of all elements in \( \mathbf{w} \)
$L_p$ norm (2)

- $p = \infty$
- $p = 4$
- $p = 2$
- $p = 1$
- $p = \frac{3}{4}$
- $p = \frac{1}{2}$
- $p = \frac{1}{4}$
- $p = 0$

- $p \downarrow \implies$ 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

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

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

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

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

$$w_{\text{ridge}}^* = \arg \min_w \left[ \sum_{i=1}^{N} (w^\top x_i - y_i)^2 + \lambda \cdot \|w\|_2^2 \right]$$

- the closed form solution:

$$w_{\text{ridge}}^* = \left( X^\top X + \lambda \cdot I \right)^{-1} X^\top y$$

- may use gradient descent to avoid the matrix inversion

supplementary slides to *Machine Learning Fundamentals* \(\copyright\) Hui Jiang 2020 published by Cambridge University Press
LASSO

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

$$w_{\text{lasso}}^* = \arg \min_w \left[ \frac{1}{2} \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \cdot \|w\|_1 \right]$$

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

$$\frac{\partial Q_{\text{lasso}}(w)}{\partial w} = \left( \sum_{i=1}^{N} x_i x_i^T \right) w - \sum_{i=1}^{N} y_i x_i + \lambda \cdot \text{sgn}(w)$$

- 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: \[
X_{n \times m} \approx U_{n \times k} \Sigma_{k \times k} V_{k \times m}
\]
- matrix factorization leads to many real-world applications
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).
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

\[
Q(U, V) = \sum_{(i,j) \in \Omega} (x_{ij} - u_i^T v_j)^2 + \lambda_1 \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda_2 \sum_{j=1}^{m} \|v_j\|_2^2
\]
bilinear models suggest an alternating algorithm

1. keep $U$ constant, estimate all $v_j$ in $V$

$$v_j = \left( \sum_{i \in \Omega_j^c} u_i u_i^T + \lambda_2 I \right)^{-1} \left( \sum_{i \in \Omega_j^c} x_{ij} u_i \right)$$

2. keep $V$ constant, estimate all $u_j$ in $U$

$$u_i = \left( \sum_{j \in \Omega_i^c} v_j v_j^T + \lambda_1 I \right)^{-1} \left( \sum_{j \in \Omega_i^c} x_{ij} v_j \right)$$

$$\arg\min_{v_j} \sum_{i \in \Omega_j^c} (x_{ij} - u_i^T v_j)^2 + \lambda_2 \cdot \|v_j\|_2^2$$
Alternating Algorithm for Matrix Factorization

set $t = 0$
randomly initialize $v_j^{(0)} \ (j = 1, 2, \cdots, m)

while not converged do
  for $i = 1, \cdots, n$ do
    $u_i^{(t+1)} = \left( \sum_{j \in \Omega_i} v_j^{(t)} (v_j^{(t)})^\top + \lambda_1 I \right)^{-1} \left( \sum_{j \in \Omega_i} x_{ij} v_j^{(t)} \right)$
  end for
  for $j = 1, \cdots, m$ do
    $v_j^{(t+1)} = \left( \sum_{i \in \Omega_j} u_i^{(t+1)} (u_i^{(t+1)})^\top + \lambda_2 I \right)^{-1} \left( \sum_{i \in \Omega_j} x_{ij} u_i^{(t+1)} \right)$
  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

\[
\mathbf{x} = \begin{bmatrix}
\mathbf{d}_1 & \cdots & \mathbf{d}_n
\end{bmatrix} \begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_n
\end{bmatrix} = \mathbf{D} \alpha
\]

- **dictionary learning**: jointly learn the dictionary \( \mathbf{D} \) and sparse codes from \( \{\alpha_i\} \) from some training samples \( \{\mathbf{x}_i\} \)

\[
\mathbf{X} = \begin{bmatrix}
\mathbf{x}_1 & \cdots & \mathbf{x}_N
\end{bmatrix} \quad \mathbf{A} = \begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_N
\end{bmatrix}
\]
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

\[
\arg\min_{D,A} \frac{1}{2} \sum_{i=1}^{N} \| x_i - D \alpha_i \|_2^2 + \lambda_1 \sum_{i=1}^{N} \| \alpha_i \|_1 + \frac{\lambda_2}{2} \sum_{j=1}^{n} \| d_j \|_2^2
\]

- compute all gradients:

\[
\frac{\partial Q(D,A)}{\partial A} = D^TDA - D^TX + \lambda_1 \cdot \text{sgn}(A)
\]

\[
\frac{\partial Q(D,A)}{\partial D} = DAA^T - XA^T + \lambda_2 \cdot D
\]
Dictionary Learning (III)

Gradient Descent for Dictionary Learning

set $t = 0$ and $\eta_0$
randomly initialize $D^{(0)}$ and $A^{(0)}$

while not converged do

update $A$:

$$A^{(t+1)} = A^{(t)} - \eta_t \left( (D^{(t)})^T D^{(t)} A^{(t)} - (D^{(t)})^T X + \lambda_1 \cdot \text{sgn}(A^{(t)}) \right)$$

update $D$:

$$D^{(t+1)} = D^{(t)} - \eta_t \left( D^{(t)} A^{(t+1)} (A^{(t+1)})^T - X (A^{(t+1)})^T + \lambda_2 \cdot D^{(t)} \right)$$

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

end while
Sparse Coding

- **Sparse coding**: given a dictionary $D$, find the sparse code $\alpha$ for a new observation $x$
- Ideal but infeasible:
  \[
  \text{arg min}_{\alpha} \| \alpha \|_0 \quad \text{s. t.} \quad D \alpha = x
  \]
  
  a practical solution:
  - replace intractable $L_0$ norm with $L_1$
  - relax to an imperfect reconstruction
  \[
  \alpha^* = \text{arg min}_{\alpha} \left\{ \frac{1}{2} \| x - D \alpha \|_2^2 + \lambda_1 \cdot \| \alpha \|_1 \right\}_{Q'(\alpha)}
  \]

- Use gradient descent:
  \[
  \frac{\partial Q'(\alpha)}{\partial \alpha} = D^T D \alpha - D^T x + \lambda_1 \cdot \text{sgn}(\alpha)
  \]