Lecture 3: Shrinkage

Isabelle Guyon
guyoni@inf.ethz.ch

References

Structural risk minimization for character recognition
Isabelle Guyon et al.

Kernel Ridge Regression
Isabelle Guyon

Ockham’s Razor

- Principle proposed by William of Ockham in the fourteenth century: “Pluralitas non est ponenda sine necesseitate”.
- Of two theories providing similarly good predictions, prefer the simplest one.
- Shave off unnecessary parameters of your models.

The Power of Amnesia

- The human brain is made out of billions of cells or Neurons, which are highly interconnected by synapses.
- Exposure to enriched environments with extra sensory and social stimulation enhances the connectivity of the synapses, but children and adolescents can lose them up to 20 million per day.
### Artificial Neurons

\[ f(x) = w \cdot x + b \]

### Hebb’s Rule

\[ w_j \leftarrow w_j + y_i x_{ij} \]

### Weight Decay

\[
\begin{align*}
    w_j &\leftarrow w_j + y_i x_{ij} & \text{Hebb’s rule} \\
    w_j &\leftarrow (1-\lambda) w_j + y_i x_{ij} & \text{Weight decay}
\end{align*}
\]

\( \lambda \in [0, 1] \), decay parameter

### Sigma-Pi Unit
One Dimensional Example

\[ f(x) = w \cdot \xi \]

Polynomial function

Conventions

- \( X = \{x_{ij}\} \) training data matrix, \( i=1:m, j=1:n \)
- \( x_i = \{x_j\} \) matrix line, training pattern \( i \)
- \( x \) test pattern, dim \( n \)
- \( y_i \) target value of pattern \( i \)
- \( y \) target value of test pattern
- \( w \) weight vector, dim \( n \)
- \( \alpha \) weight vector, dim \( m \)
### Matrix Notations

\[ w_j = \sum_i y_i x_{ij} \]
\[ w = y^T X \]
\[ w^T = X^T y \]

\[ f(x) = \sum_j w_j x_j \]
\[ f(x) = x w^T = w x^T \]

### Linear Regression

- What we want:
  \[ \sum_j w_j x_{ij} = y_i \text{ for all examples } i=1 \ldots m \] (\( b=w_0 \))
  or for classification, \( y_i=\pm 1, \text{ sign} (\sum_j w_j x_{ij}) = y_i \)

- Solve:
  \[ X w^T = y \]

### Regression: \( m>n \)

- Solve:
  \[ X w^T = y \]

- Normal equations
  \[ X^T X w^T = X^T y \]

- Solution:
  \[ w^T = (X^T X)^{-1} X^T y \]

### Pseudo-Inverse

- Solution:
  \[ w^T = (X^T X)^{-1} X^T y \]

- Predictor:
  \[ f(x) = x w^T = x X^T y \]

- Residual:
  \[ y - y = y - X w^T = (I - XX^T) y \]
### Least-Squares

The pseudo-inverse solution is optimal in the least-square sense:

\[
\min_w \| y - Xw^T \|^2 = \| (I-XX^T)y \|^2
\]

### Gradient Descent

- **Square loss:**
  \[
  L_i = (x_iw^T - y_i)^2
  \]
- **Sum of squares:**
  \[
  R = \sum (x_iw^T - y_i)^2 = \| Xw^T - y \|^2 = wX^TXw^T - 2X^Ty + y^Ty
  \]
- **Gradient:**
  \[
  \nabla_w R = 2 (X^TXw^T - X^Ty)
  \]

### Normal Equations

- At the optimum:
  \[
  \nabla_w R = 0
  \]
  \[
  2 (X^TXw^T - X^Ty) = 0
  \]
- **Normal equations (again):**
  \[
  X^TXw^T = X^Ty
  \]
  Solve by inverting \( X^TX \), if regular.
- What if \( X^TX \) is singular?

### Regularization

- **Normal equations:**
  \[
  X^TXw^T = X^Ty
  \]
- Case \( m<n \) (interpolation), rank(\( X \)) \( \leq m<n \), matrix \( X^TX \) singular.
- Replace \( X^TX \) by \( (X^TX + \lambda I)^{-1}X^Ty \), \( \lambda > 0 \)
- **Solution:**
  \[
  w^T = (X^TX + \lambda I)^{-1}X^Ty
  \]
  Regularized inverse
**Why it works**

- **Diagonalization:**
  \[ X'X = U D U^T \]
  
  - U orthogonal matrix of eigenvectors (\(UU^T=I\))
  - D diagonal matrix of eigenvalues
  - Singularity: some eigenvalues are zero.
- **Regularization:**
  \[ X'X + \lambda I = U (D + \lambda I) U^T \]
  \(\lambda > 0\)
  no more zero eigenvalue.

**Penalized Risk**

- **Sum of squares:**
  \[ R = \Sigma \langle x_i, w^T - y \rangle^2 = ||Xw^T - y||^2 \]
- **Add “regularizer”:**
  \[ R = ||Xw^T - y||^2 + \lambda ||w||^2 \]
- **Gradient:**
  \[ \nabla_{w}R = 2 ((X'X + \lambda I)w^T - X'y) - 2\lambda w \]

**Mechanical Interpretation**

- **Quadratic form:**
  \[ R = ||Xw^T - y||^2 + \lambda ||w||^2 \]
- **One dimension:**
  \[ R = p (w-w_0)^2 + \lambda w^2 \]
- **Two dimensions:**

  - Problem: Construct features that are linear combinations of the original features, such that the reconstructed patterns are as close as possible to the original in the least square sense.
  - \(f'_k = X u_k\) linear combinations of columns of X
  - \(x''_i = x'_i  U' = \Sigma x' u_k\) reconstructed pattern
  - \(x''_i \neq x'_i U' \neq \Sigma X' \neq X' \neq \Sigma U' \neq X''_i\)
### PCA Solution

- $X' = X U$
- $X'' = X' U^T$
- $\min_U ||X - X U U^T||^2$
- Can be brought back to solving and eigenvalue problem: $X^T X = U D U^T$ i.e. $X^T X = D$
- Compare:
  - Regularization $X^T X + \lambda I = U (D + \lambda I) U^T$
  - PCA: Remove the dimensions with smallest eigenvalues.

### Kernel “Trick” ($m<n$)

- Solve: $X w^T = y$
- Assume: $w = \sum_i \alpha_i x_i = \alpha^T X$
- Solve instead: $X^T X \alpha = y$
- Solution: $\alpha = (X^T X)^{-1} y$

### Kernel Ridge Regression

- $\Xi = \Phi(X)$
- Solve: $\Xi w^T = y$
- Assume: $w = \sum_{i} \alpha_i \xi_i = \alpha^T \Xi$
- Solve instead: $\Xi^T \alpha = y$
- Solution: $\alpha = K^{-1} y$
- Regularization: replace $K$ by $K + \lambda I$

### Regularization and PI

- Case $m>n$ and $\text{rank}(X^T X) = n$
  - $X^* = (X^T X)^{-1} X^T$
- Case $m<n$ and $\text{rank}(X^T X) = m$
  - $X^* = X^T (X^T X)^{-1}$
- Either case:
  - $X^* = \lim_{\lambda \to 0} (X^T X + \lambda I)^{-1} X^T$
  - $= \lim_{\lambda \to 0} X^T (X^T X + \lambda)^{-1}$
**Weight Decay for MLP**

Replace: \( w_j \leftarrow w_j + \text{back}_\text{prop}(j) \)

by:

\[
\begin{align*}
    w_j & \leftarrow (1-\lambda) w_j + \text{back}_\text{prop}(j) \\
\end{align*}
\]

---

**Priors and Bayesian Learning**

- Double random process:
  - Draw a target function \( f \) in a family of functions \( \{f\} \)
  - Draw the data pairs \( (x_i, y_i = f(x_i) + \text{noise}) \)
- The distribution of \( f \) is called the “prior” \( P(f) \).
- Our revised opinion about \( f \) once we see the data is the “posterior” \( P(f|D) \).
- Bayesian “learning”:
  \[
  P(y|x,D) \propto \int P(y|x,D,f) \, dP(f|D) 
  \]
- MAP:
  \[
  f = \underset{f}{\text{argmax}} \ P(f|D) \\
  = \underset{f}{\text{argmax}} \ P(D|f) \, P(f) 
  \]

---

**MAP = RRM**

- Maximum A Posteriori (MAP):
  \[
  f = \underset{f}{\text{argmax}} \ P(D|f) \, P(f) \\
  = \underset{f}{\text{argmin}} \ -\log P(D|f) \quad -\log P(f) 
  \]
  = Empirical risk \( R[f] \) = Regularizer \( \Omega[f] \)

- Regularized Risk Minimization (RRM):
  \[
  f = \underset{f}{\text{argmin}} \ R[f] + \Omega[f] 
  \]

---

**Example: Gaussian Prior**

- Linear model:
  \[
  f(x) = x^TW^T 
  \]
- Square loss \( \Leftrightarrow \) Gaussian noise:
  \[
  P(D|f) = \exp\left(-\frac{||X^T w - y||^2}{\sigma^2}\right) 
  \]
  \[
  R[f] = -\log P(D|f) \sim ||X^T w - y||^2 
  \]
- Weight decay \( \Leftrightarrow \) Gaussian prior:
  \[
  P(f) = \exp\left(-\lambda||w||^2\right) 
  \]
  \[
  \Omega[f] = -\log P(f) = \lambda||w||^2 
  \]
Structural Risk Minimization

- Nested subsets of models, increasing complexity/capacity:
  \[ S_1 \subset S_2 \subset \ldots \subset S_N \]
- Example, rank with \( \|w\|^2 \)
  \[ S_k = \{ w \mid \|w\|^2 < A_k \}, A_1 < A_2 < \ldots < A_k \]
- Minimization under constraint:
  \[ \min R_{emp}[f] \quad \text{s.t.} \quad \|w\|^2 < A_k \]
- Lagrangian:
  \[ R_{reg}[f] = R_{emp}[f] + \lambda \|w\|^2 \]

Conclusion

- Weight decay is a means of avoiding "overfitting" that is justifiable from many perspectives:
  - Ockham's razor
  - Synaptic decay
  - Regularization
  - Gaussian prior on the weights
  - Structural risk minimization
- It works for linear models, kernel methods, and neural networks.
- It can be combined with various loss functions.

Practical Work

Homework 3:

1) Same data and software as homework 2.
2) Create a heatmap of the 100 top ranking features you selected.
3) Make a scatter plot of the 3 top ranking features you selected.
4) Email the result zip file with the figures to guyoni@inf.ethz.ch with subject "Homework3" no later than:
   Tuesday November 15th.
Risk Minimization

- **Learning problem**: find the best function \( f(x; \alpha) \) minimizing the risk functional
  \[
  R[f] = \int \mathcal{L}(f(x; \alpha), y) \, dP(x, y)
  \]
- **Examples are given**:
  \((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\)

Loss Functions

- **Logistic loss**: \( \log(1 + e^{-z}) \)
- **SVC loss**: \( \max(0, 1 - z) \)
- **Perceptron loss**: \( \max(0, -z) \)
- **Kernel “Trick”**
  - \( f(x) = \sum_{i} \alpha_i k(x_i, x) \)
  - \( k(x_i, x) = \Phi(x_i) \cdot \Phi(x) \)
  - **Dual forms**
    - \( f(x) = w \cdot \Phi(x) \)
    - \( w = \sum_{i} \alpha_i \Phi(x_i) \)
  - **Examples**:
    - \( k(s, t) = \exp(-||s-t||^2/\sigma^2) \) Gaussian kernel
    - \( k(s, t) = 1/||s-t|| \) Potential function
    - \( k(s, t) = (s \cdot t)^q \) Polynomal kernel

What is a Kernel?

A kernel is a dot product in some feature space: \( k(s, t) = \Phi(s) \cdot \Phi(t) \)

- **Examples**:
  - \( k(s, t) = \exp(-||s-t||^2/\sigma^2) \) Gaussian kernel
  - \( k(s, t) = 1/||s-t|| \) Potential function
  - \( k(s, t) = (s \cdot t)^q \) Polynomal kernel

\[
\begin{align*}
  k(s, t) &= \begin{bmatrix} s_1^2 & s_2^2 & \ldots & s_n^2 \\
                        s_1 t_1 & s_2 t_2 & \ldots & s_n t_n \\
                        \vdots       & \vdots       & \ddots & \vdots
                        s_1 t_n & s_2 t_n & \ldots & t_n^2
\end{bmatrix} \\
  \Phi(s) &= \begin{bmatrix} s_1 \\
                        s_2 \\
                        \vdots \\
                        s_n
\end{bmatrix} \\
  \Phi(t) &= \begin{bmatrix} t_1 \\
                        t_2 \\
                        \vdots \\
                        t_n
\end{bmatrix}
\end{align*}
\]
Simple Kernel Methods

\[ f(x) = w \cdot \Phi(x) \]
\[ w = \sum \alpha_i \Phi(x_i) \]

Perceptron algorithm
\[ w \leftarrow w + y_i \Phi(x_i) \quad \text{if } y_i f(x_i) < 0 \]
(Rosenblatt 1958)

Minover (optimum margin)
\[ w \leftarrow w + y_i \Phi(x_i) \quad \text{for min } y_i f(x_i) \]
(Krauth-Mézard 1987)

LMS regression
\[ w \leftarrow w + \eta (y_i - f(x_i)) \Phi(x_i) \]

Potential Function algorithm
\[ \alpha_i \leftarrow \alpha_i + y_i \quad \text{if } y_i f(x_i) > 0 \]
(Aizerman et al 1964)

Dual minover
\[ \alpha_i \leftarrow \alpha_i + y_i \quad \text{for min } y_i f(x_i) \]

Dual LMS
\[ \alpha_i \leftarrow \alpha_i + \eta (y_i - f(x_i)) \]

Exercise: Gradient Descent

- Linear discriminant \( f(x) = \sum w_j x_j \)
- Functional margin \( z = y f(x), y = \pm 1 \)
- Compute \( \frac{\partial z}{\partial w_j} \)
- Derive the learning rules \( \Delta w_j = -\eta \frac{\partial L}{\partial w_j} \)
  corresponding to the following loss functions:
  - square loss \( (1 - z)^2 \)
  - SVC loss \( \max(0, 1-z) \)
  - Adaboost loss \( e^{-z} \)
  - Perceptron loss \( \max(0, -z) \)
  - logistic loss \( \log(1 + e^z) \)

Exercise: Dual Algorithms

- From the derive the \( \Delta w_j \) derive the \( \Delta w \)
- \( w = \sum \alpha_i x_i \)
- From the \( \Delta w \), derive the \( \Delta \alpha \) of the dual algorithms.

Exercise: Linear Algebra

- Prove that if \( X \) is of rank \( r \), \( X^T X \) and \( XX^T \) have the same rank.
- Show that \( X^T X \) and \( XX^T \) have only positive eigenvalues.