Lecture 4:
Regularization, Sparsity & Lasso
Recap: Complete Pipeline

- **Training Data**: \( S = \{(x_i, y_i)\}_{i=1}^{N} \)
- **Model Class(es)**: \( f(x | w, b) = w^T x - b \)
- **Loss Function**: \( L(a, b) = (a - b)^2 \)

\[
\arg\min_{w,b} \sum_{i=1}^{N} L(y_i, f(x_i | w, b)) \quad \text{SGD!}
\]

**Cross Validation & Model Selection**

**Profit!**
Different Model Classes?

• Option 1: SVMs vs ANNs vs LR vs LS
• Option 2: Regularization

Cross Validation & Model Selection

\[
\text{argmin}_{w,b} \sum_{i=1}^{N} L(y_i, f(x_i | w, b)) \quad \text{SGD!}
\]
Notation Part 1

• **L0 Norm** (not actually a norm)
  – # of non-zero entries
  \[ \| w \|_0 = \sum_d 1_{[w_d \neq 0]} \]

• **L1 Norm**
  – Sum of absolute values
  \[ |w| = \| w \|_1 = \sum_d |w_d| \]

• **L2 Norm & Squared L2 Norm**
  – Sum of squares
  – Sqrt(sum of squares)
  \[ \| w \| = \sqrt{\sum_d w_d^2} \equiv \sqrt{w^T w} \]
  \[ \| w \|^2 = \sum_d w_d^2 \equiv w^T w \]

• **L-infinity Norm**
  – Max absolute value
  \[ \| w \|_\infty = \lim_{p \to \infty} \sqrt[p]{\sum_d |w_d|^p} = \max_d |w_d| \]
Notation Part 2

• Minimizing Squared Loss
  – Regression
  – Least-Squares

  – (Unless Otherwise Stated)
    • E.g., Logistic Regression = Log Loss

\[
\text{argmin}_w \sum_i (y_i - w^T x_i + b)^2
\]
Ridge Regression

\[
\text{argmin}_{w,b} \lambda w^T w + \sum_{i} (y_i - w^T x_i + b)^2
\]

- aka L2-Regularized Regression
- Trades off model complexity vs training loss
- Each choice of \( \lambda \) a “model class”
  - Will discuss the further later
\[
\text{argmin}_{w,b} \lambda w^T w + \sum_i (y_i - w^T x_i + b)^2
\]

<table>
<thead>
<tr>
<th>Person</th>
<th>Age&gt;10</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Carol</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Erin</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Frank</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Harold</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Irene</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>John</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Kelly</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Larry</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Updated Pipeline

\[ S = \left\{ (x_i, y_i) \right\}_{i=1}^{N} \]

Training Data

\[ f(x \mid w, b) = w^T x - b \]

Model Class

\[ L(a, b) = (a - b)^2 \]

Loss Function

\[ \arg\min_{w,b} \lambda w^T w + \sum_{i=1}^{N} L(y_i, f(x_i \mid w, b)) \]

Choosing \( \lambda \)!

Cross Validation & Model Selection

Profit!
<table>
<thead>
<tr>
<th>Person</th>
<th>Age&gt;10</th>
<th>Male</th>
<th>Height &gt; 55”</th>
<th>Model Score w/ Increasing Lambda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.91 0.89 0.83 0.75 0.67</td>
</tr>
<tr>
<td>Bob</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.42 0.45 0.50 0.58 0.67</td>
</tr>
<tr>
<td>Carol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.17 0.26 0.42 0.50 0.67</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.16 1.06 0.91 0.83 0.67</td>
</tr>
<tr>
<td>Erin</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.91 0.89 0.83 0.79 0.67</td>
</tr>
<tr>
<td>Frank</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.42 0.45 0.50 0.54 0.67</td>
</tr>
<tr>
<td>Gena</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.17 0.27 0.42 0.50 0.67</td>
</tr>
<tr>
<td>Harold</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.16 1.06 0.91 0.83 0.67</td>
</tr>
<tr>
<td>Irene</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.91 0.89 0.83 0.79 0.67</td>
</tr>
<tr>
<td>John</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.42 0.45 0.50 0.54 0.67</td>
</tr>
<tr>
<td>Kelly</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.91 0.89 0.83 0.79 0.67</td>
</tr>
<tr>
<td>Larry</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.16 1.06 0.91 0.83 0.67</td>
</tr>
</tbody>
</table>
Choice of Lambda Depends on Training Size

25 Training Points

50 Training Points

75 Training Points

100 Training Points

25 dimensional space
Randomly generated linear response function + noise
Recap: Ridge Regularization

• Ridge Regression:
  – L2 Regularized Least-Squares
    \[
    \arg\min_{w,b} \lambda w^T w + \sum_i \left( y_i - w^T x_i + b \right)^2
    \]

• Large $\lambda \Rightarrow$ more stable predictions
  – Less likely to overfit to training data
  – Too large $\lambda \Rightarrow$ underfit

• Works with other loss
  – Hinge Loss, Log Loss, etc.
Aside: Stochastic Gradient Descent

\[
\text{argmin}_{w,b} \lambda \|w\|^2 + \sum_{i=1}^{N} L(y_i, f(x_i \mid w, b))
\]

\[
\tilde{L}(w, b) = \sum_{i=1}^{N} \left[ \frac{1}{N} \lambda \|w\|^2 + L_i(w, b) \right]
\]

\[
\frac{1}{N} \tilde{L}(w, b) = \mathbb{E}_i \left[ \tilde{L}_i(w, b) \right]
\]

Do SGD on this
Model Class Interpretation

\[ \arg\min_{w,b} \lambda w^T w + \sum_{i=1}^{N} L(y_i, f(x_i \mid w, b)) \]

• This is not a model class!
  – At least not what we’ve discussed...

• An optimization procedure
  – Is there a connection?
Norm Constrained Model Class

\[ f(x \mid w, b) = w^T x - b \quad \text{s.t.} \quad w^T w \leq c \equiv \|w\|^2 \leq c \]

Seems to correspond to lambda...

\[ \arg\min_{w, b} \lambda w^T w + \sum_{i=1}^{N} L(y_i, f(x_i \mid w, b)) \]
Lagrange Multipliers

\[ \arg\min_w L(y, w) \equiv (y - w^T x)^2 \]

- Optimality Condition:
  - Gradients aligned!
  - Constraint Boundary
  - Loss

\[ \exists \lambda \geq 0 : \left( \partial_w L(y, w) = -\lambda \partial_w w^T w \right) \land \left( w^T w \leq c \right) \]

Omitting b & 1 training data for simplicity

http://en.wikipedia.org/wiki/Lagrange_multiplier
**Norm Constrained Model Class Training:**

\[
\arg\min_w L(y, w) \equiv \left(y - w^T x\right)^2 \quad \text{s.t. } w^T w \leq c
\]

**Observation about Optimality:**

\[
\exists \lambda \geq 0 : \left(\partial_w L(y, w) = -\lambda \partial_w w^T w\right) \land \left(w^T w \leq c\right)
\]

**Lagrangian:**

\[
\arg\min_{w, \lambda} \Lambda(w, \lambda) = \left(y - w^T x\right)^2 + \lambda \left(w^T w - c\right)
\]

**Claim:** Solving Lagrangian Solves Norm-Constrained Training Problem

**Optimality Implication of Lagrangian:**

\[
\partial_w \Lambda(w, \lambda) = -2x \left(y - w^T x\right)^T + 2\lambda w \equiv 0
\]

\[
\Rightarrow 2x (y - w^T x)^T = -2\lambda w
\]

http://en.wikipedia.org/wiki/Lagrange_multiplier
Norm Constrained Model Class Training:
\[ \arg\min_w L(y, w) \equiv (y - w^T x)^2 \quad \text{s.t.} \quad w^T w \leq c \]

Observation about Optimality:
\[ \exists \lambda \geq 0 : \left( \partial_w L(y, w) = -\lambda \partial_w w^T w \right) \land \left( w^T w \leq c \right) \]

Lagrangian:
\[ \arg\min_{w, \lambda} \Lambda(w, \lambda) = (y - w^T x)^2 + \lambda \left( w^T w - c \right) \]

Claim: Solving Lagrangian Solves Norm-Constrained Training Problem

Optimality Implication of Lagrangian:
\[ \partial_\lambda \Lambda(w, \lambda) = \begin{cases} 
0 & \text{if } w^T w < c \\
 w^T w - c & \text{if } w^T w \geq c 
\end{cases} \equiv 0 \quad \Rightarrow \quad w^T w \leq c \]

http://en.wikipedia.org/wiki/Lagrange_multiplier
Norm Constrained Model Class Training:
\[
\arg\min_w L(y, w) \equiv \left( y - w^T x \right)^2 \quad \text{s.t. } w^T w \leq c
\]

L2 Regularized Training:
\[
\arg\min_w \lambda w^T w + \left( y - w^T x \right)^2
\]

Lagrangian:
\[
\arg\min_{w, \lambda} \Lambda(w, \lambda) = \left( y - w^T x \right)^2 + \lambda \left( w^T w - c \right)
\]

- Lagrangian = Norm Constrained Training:
  \[
  \exists \lambda \geq 0 : \left( \partial_w L(y, w) = -\lambda \partial_w w^T w \right) \land \left( w^T w \leq c \right)
  \]

- Lagrangian = L2 Regularized Training:
  - Hold \( \lambda \) fixed
  - Equivalent to solving Norm Constrained!
  - For some \( c \)

http://en.wikipedia.org/wiki/Lagrange_multiplier
Recap #2: Ridge Regularization

• Ridge Regression:
  – L2 Regularized Least-Squares = Norm Constrained Model

\[ \arg\min_{w,b} \lambda w^T w + L(w) \equiv \arg\min_{w,b} L(w) \text{ s.t. } w^T w \leq c \]

• Large \( \lambda \) \( \Rightarrow \) more stable predictions
  – Less likely to overfit to training data
  – Too large \( \lambda \) \( \Rightarrow \) underfit

• Works with other loss
  – Hinge Loss, Log Loss, etc.
Hallucinating Data Points

\[
\arg\min_w \lambda w^T w + \sum_{i=1}^{N} \left( y_i - w^T x_i \right)^2
\]

\[
\partial_w = 2\lambda w - 2\sum_{i=1}^{N} x \left( y_i - w^T x_i \right)^T
\]

• Instead hallucinate D data points?

\[
\arg\min_w \sum_{d=1}^{D} \left( 0 - w^T \sqrt{\lambda} e_d \right)^2 + \sum_{i=1}^{N} \left( y_i - w^T x_i \right)^2
\]

\[
\partial_w = 2\sum_{d=1}^{D} \sqrt{\lambda} e_d \left( w^T \sqrt{\lambda} e_d \right)^T - 2\sum_{i=1}^{N} x \left( y_i - w^T x_i \right)^T
\]

\[
= 2\sum_{d=1}^{D} \lambda e_d^T w = 2\sum_{d=1}^{D} \lambda w_d = 2\lambda w
\]

Identical to Regularization!  

Unit vector along d-th Dimension  

\[
e_d = \begin{bmatrix}
0 \\
\vdots \\
0 \\
1 \\
\vdots \\
0
\end{bmatrix}
\]

\[
\{ (\sqrt{\lambda} e_d, 0) \}_{d=1}^{D}
\]  

Omitting b & for simplicity
Extension: Multi-task Learning

- 2 prediction tasks:
  - Spam filter for Alice
  - Spam filter for Bob

- Limited training data for both...
  - ... but Alice is similar to Bob
Extension: Multi-task Learning

- Two Training Sets
  - N relatively small

- **Option 1: Train Separately**
  \[
  S^{(1)} = \left\{ (x_i^{(1)}, y_i^{(1)}) \right\}_{i=1}^{N}
  \]
  \[
  S^{(2)} = \left\{ (x_i^{(2)}, y_i^{(2)}) \right\}_{i=1}^{N}
  \]

\[
\begin{align*}
\arg\min_w & \lambda w^T w + \sum_{i=1}^{N} \left( y_i^{(1)} - w^T x_i^{(1)} \right)^2 \\
\arg\min_v & \lambda v^T v + \sum_{i=1}^{N} \left( y_i^{(2)} - v^T x_i^{(2)} \right)^2
\end{align*}
\]

Both models have high error.

Omitting b &
for simplicity
Extension: Multi-task Learning

• Two Training Sets
  – N relatively small

• Option 2: Train Jointly

\[
S^{(1)} = \left\{ (x_i^{(1)}, y_i^{(1)}) \right\}_{i=1}^{N}
\]

\[
S^{(2)} = \left\{ (x_i^{(2)}, y_i^{(2)}) \right\}_{i=1}^{N}
\]

\[
\arg \min_{w,v} \lambda w^T w + \sum_{i=1}^{N} \left( y_i^{(1)} - w^T x_i^{(1)} \right)^2
\]

\[
+ \lambda v^T v + \sum_{i=1}^{N} \left( y_i^{(2)} - v^T x_i^{(2)} \right)^2
\]

Doesn’t accomplish anything!
(w & v don’t depend on each other)

Omitting b & for simplicity
Multi-task Regularization

\[ \arg\min_{w,v} \lambda w^T w + \lambda v^T v + \gamma (w - v)^T (w - v) + \sum_{i=1}^{N} \left( y^{(1)}_i - w^T x^{(1)}_i \right)^2 + \sum_{i=1}^{N} \left( y^{(2)}_i - v^T x^{(2)}_i \right)^2 \]

- Prefer \( w \) & \( v \) to be “close”
  - Controlled by \( \gamma \)
  - Tasks similar
    - Larger \( \gamma \) helps!
  - Tasks not identical
    - \( \gamma \) not too large

![Test Loss (Task 2)](chart.png)
Lasso

L1-Regularized Least-Squares
L1 Regularized Least Squares

\[
\begin{align*}
\text{argmin } \lambda |w| + \sum_{i=1}^{N} (y_i - w^T x_i)^2 \\
\text{argmin } \lambda \|w\|^2 + \sum_{i=1}^{N} (y_i - w^T x_i)^2
\end{align*}
\]

- **L2:**
  \[
  w = \sqrt{2} \quad \text{vs} \quad w = 1
  \]
  \[
  w = 1 \quad \text{vs} \quad w = 0
  \]

- **L1:**
  \[
  w = 2 \quad \text{vs} \quad w = 1
  \]
  \[
  w = 1 \quad \text{vs} \quad w = 0
  \]

Omitting b & for simplicity
Aside: Subgradient (sub-differential)

\[ \nabla_a R(a) = \left\{ c \mid \forall a': R(a') - R(a) \geq c(a' - a) \right\} \]

- Differentiable: \( \nabla_a R(a) = \partial_a R(a) \)

- L1:

\[ \nabla_{w_d} |w| = \begin{cases} 
-1 & \text{if } w_d < 0 \\
+1 & \text{if } w_d > 0 \\
[-1, +1] & \text{if } w_d = 0 
\end{cases} \]

Continuous range for \( w=0 \)!

Omitting \( b \) & for simplicity
L1 Regularized Least Squares

$$\arg\min_w \lambda |w| + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2$$

$$\arg\min_w \lambda \|w\|^2 + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2$$

• L2:

$$\nabla_{w_d} \|w\|^2 = 2w_d$$

• L1:

$$\nabla_{w_d} |w| = \begin{cases} 
-1 & \text{if } w_d < 0 \\
+1 & \text{if } w_d > 0 \\
[-1, +1] & \text{if } w_d = 0
\end{cases}$$

Omitting $b$ & for simplicity
Lagrange Multipliers

\[ \text{argmin}_{w} L(y, w) \equiv \left( y - w^T x \right)^2 \]

s.t. \[ |w| \leq c \]

\[ \nabla_{w_d} |w| \begin{cases} 
-1 & \text{if } w_d < 0 \\
+1 & \text{if } w_d > 0 \\
[-1, +1] & \text{if } w_d = 0 
\end{cases} \]

\[ \exists \lambda \geq 0 : \left( \partial_w L(y, w) \in -\lambda \nabla_{w} |w| \right) \land \left( |w| \leq c \right) \]

Omitting b & 1 training data for simplicity

http://en.wikipedia.org/wiki/Lagrange_multiplier
Sparsity

• w is sparse if mostly 0’s:
  – Small L0 Norm

\[ \|w\|_0 = \sum_d 1_{[w_d \neq 0]} \]

• Why not L0 Regularization?
  – **Not continuous!**

• L1 induces sparsity
  – And is continuous!

\[
\begin{align*}
\arg\min_w \lambda \|w\|_0 + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2 \\
\arg\min_w \lambda |w| + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2
\end{align*}
\]

Omitting b & for simplicity
Why is Sparsity Important?

• Computational / Memory Efficiency
  – Store 1M numbers in array
  – Store 2 numbers per non-zero
    • (Index, Value) pairs
    • E.g., [ (50,1), (51,1) ]
  – Dot product more efficient: $w^T x$

• Sometimes true $w$ is sparse
  – Want to recover non-zero dimensions
Lasso Guarantee

\[
\arg\min_w \lambda |w| + \sum_{i=1}^{N} \left( y_i - w^T x_i + b \right)^2
\]

• Suppose data generated as: \( y_i \sim \text{Normal}(w_*^T x_i, \sigma^2) \)

• Then if: \( \lambda > \frac{2}{\kappa} \sqrt{\frac{2\sigma^2 \log D}{N}} \)

• With high probability (increasing with N):

\[
\text{Supp}(w) \subseteq \text{Supp}(w_*)
\]

\( \forall d : |w_d| \geq \lambda c \Rightarrow \text{Supp}(w) = \text{Supp}(w_*) \)

\[
\text{Supp}(w_*) = \{ d | w_{*,d} \neq 0 \}
\]

High Precision
Parameter Recovery

Sometimes High Recall

See also: https://www.cs.utexas.edu/~pradeepr/courses/395T-LT/filez/highdimII.pdf
http://www.eecs.berkeley.edu/~wainwrig/Papers/Wai_SparseInfo09.pdf
<table>
<thead>
<tr>
<th>Person</th>
<th>Age &gt;10</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Carol</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dave</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Erin</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Frank</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Harold</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Irene</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>John</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Kelly</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Larry</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Magnitude of the two weights. (As regularization shrinks)
Aside: Optimizing Lasso

• Solving Lasso gives sparse model
  – Will stochastic gradient descent find it?

• No!
  – Hard to hit exactly 0 with gradient descent

• Solution: Iterative Soft Thresholding
  – Intuition: if gradient update passes 0, clamp at 0

Recap: Lasso vs Ridge

• Model Assumptions
  – Lasso learns sparse weight vector

• Predictive Accuracy
  – Lasso often not as accurate
  – Re-run Least Squares on dimensions selected by Lasso

• Ease of Inspection
  – Sparse w’s easier to inspect

• Ease of Optimization
  – Lasso somewhat trickier to optimize
Recap: Regularization

- **L2**
  \[ \text{argmin}_w \lambda \|w\|^2 + \sum_{i=1}^{N} (y_i - w^T x_i)^2 \]

- **L1 (Lasso)**
  \[ \text{argmin}_w \lambda |w| + \sum_{i=1}^{N} (y_i - w^T x_i)^2 \]

- **Multi-task**
  \[ \text{argmin}_{w, v} \lambda w^T w + \lambda v^T v + \gamma (w - v)^T (w - v) \]
  \[ + \sum_{i=1}^{N} (y_i^{(1)} - w^T x_i^{(1)})^2 + \sum_{i=1}^{N} (y_i^{(2)} - v^T x_i^{(2)})^2 \]

- **[Insert Yours Here!]**

Omitting b & for simplicity
Recap: Updated Pipeline

\[ S = \{(x_i, y_i)\}_{i=1}^{N} \]
Training Data

\[ f(x | w, b) = w^T x - b \]
Model Class

\[ L(a, b) = (a - b)^2 \]
Loss Function

\[ \arg\min_{w,b} \lambda w^T w + \sum_{i=1}^{N} L(y_i, f(x_i | w, b)) \]
Cross Validation & Model Selection

Choosing $\lambda$!

Profit!
Next Lectures

• Decision Trees
• Bagging
• Random Forests
• Boosting
• Ensemble Selection

• Recitation tonight: Linear Algebra