Introduction to Machine Learning

6. Kernels Methods

Alex Smola
Carnegie Mellon University

http://alex.smola.org/teaching/cmu2013-10-701
10-701
Regression

"Under hypnosis you revealed that in your last eight lives you were ... er ... a cat."
Regression Estimation

- Find function $f$ minimizing regression error
  \[ R[f] := \mathbb{E}_{x,y \sim p(x,y)} [l(y, f(x))] \]

- Compute empirical average
  \[ R_{\text{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i)) \]

  Overfitting as we minimize empirical error

- Add regularization for capacity control
  \[ R_{\text{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i)) + \lambda \Omega[f] \]
Squared loss

\[ l(y, f(x)) = \frac{1}{2} (y - f(x))^2 \]
$l(y, f(x)) = |y - f(x)|$
$\varepsilon$-insensitive Loss

$$l(y, f(x)) = \max(0, |y - f(x)| - \varepsilon)$$
Penalized least mean squares

• Optimization problem

\[
\min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle x_i, w \rangle)^2 + \frac{\lambda}{2} \|w\|^2
\]

• Solution

\[
\partial_w [\ldots] = \frac{1}{m} \sum_{i=1}^{m} \left[ x_i x_i^\top w - x_i y_i \right] + \lambda w
\]

\[
= \left[ \frac{1}{m} X X^\top + \lambda 1 \right] w - \frac{1}{m} X y = 0
\]

hence \( w = \left[ X X^\top + \lambda m 1 \right]^{-1} X y \)

Conjugate Gradient
Sherman Morrison Woodbury

Outer product matrix in X
Penalized least mean squares ... now with kernels

- **Optimization problem**

\[
\text{minimize} \quad \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \| w \|^2
\]

- **Representer Theorem** (Kimeldorf & Wahba, 1971)

\[
\| w \|^2 = \| w \|^2 + \| w_\perp \|^2
\]

**empirical risk dependent**
Penalized least mean squares
... now with kernels

- Optimization problem

\[
\min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2
\]

- Representer Theorem (Kimeldorf & Wahba, 1971)
  - Optimal solution is in span of data
  \[ w = \sum_i \alpha_i \phi(x_i) \]
  - Proof - risk term only depends on data via \( \phi(x_i) \)
  - Regularization ensures that orthogonal part is 0

- Optimization problem in terms of \( w \)

\[
\min_\alpha \frac{1}{2m} \sum_{i=1}^{m} \left( y_i - \sum_j K_{ij} \alpha_j \right)^2 + \frac{\lambda}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij}
\]

solve for \( \alpha = (K + m\lambda 1)^{-1} y \) as linear system
SVM Regression (\(\varepsilon\)-insensitive loss)

In SV regression, a tube with radius \(\varepsilon\) is fitted to the data. The trade-off between model complexity and points lying outside of the tube is determined by minimizing (1.48).

Note that the term \(\|w\|^2\) is the same as in pattern recognition (cf. (1.41)); for further details, cf. Chapter 9.

We can transform this into a constrained optimization problem by introducing slack variables, akin to the soft margin case. In the present case, we need two types of slack variable for the two cases \(f(x_i) - y_i > \varepsilon\) and \(y_i - f(x_i) > \varepsilon\), respectively. We denote them by \(\xi\) and \(\xi^*\), respectively, and collectively refer to the masses \(\xi(\ast)\).

The optimization problem consists in finding

\[
\min_{w \in H, \xi(\ast) \in \mathbb{R}^m, b \in \mathbb{R}} \left( \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \right) = (1.48)
\]

subject to

\[
f(x_i) - y_i \leq \varepsilon + \xi_i
\]

\[
y_i - f(x_i) \leq \varepsilon + \xi_i^*
\]

\[
\xi_i, \xi_i^* \geq 0
\]

for all \(i = 1, \ldots, m\).

Note that according to (1.49) and (1.50), any error smaller than \(\varepsilon\) does not require an nonzero \(\xi_i\) or \(\xi_i^*\) and hence does not enter the objective function (1.48).

Generalization to kernel-based regression estimation is carried out in an analogous manner to the case of pattern recognition. Introducing Lagrange multipliers, one arrives at the following optimization problem: for \(C > 0\) choose an a priori,

\[
\max_{\alpha, \alpha^* \in \mathbb{R}^m} W(\alpha, \alpha^*) = -\varepsilon \sum_{i=1}^{m} \left( \alpha_i^* + \alpha_i \right) + \sum_{i=1}^{m} \left( \alpha_i^* - \alpha_i \right) y_i - \frac{1}{2} m \sum_{i,j=1}^{m} \left( \alpha_i^* - \alpha_i \right) \left( \alpha_j^* - \alpha_j \right) k(x_i, x_j), (1.52)
\]

subject to

\[
0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, m ,
\]

and

\[
\sum_{i=1}^{m} \left( \alpha_i - \alpha_i^* \right) = 0
\]

(1.53)

\(\text{don’t care about deviations within the tube}\)
SVM Regression
(\(\varepsilon\)-insensitive loss)

• Optimization Problem (as constrained QP)

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} [\xi_i + \xi_i^*] \\
\text{subject to} & \quad \langle w, x_i \rangle + b \leq y_i + \varepsilon + \xi_i \quad \text{and} \quad \xi_i \geq 0 \\
& \quad \langle w, x_i \rangle + b \geq y_i - \varepsilon - \xi_i^* \quad \text{and} \quad \xi_i^* \geq 0
\end{align*}
\]

• Lagrange Function

\[
L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} [\xi_i + \xi_i^*] - \sum_{i=1}^{m} [\eta_i \xi_i + \eta_i^* \xi_i^*] + \\
\sum_{i=1}^{m} \alpha_i [\langle w, x_i \rangle + b - y_i - \varepsilon - \xi_i] + \sum_{i=1}^{m} \alpha_i^* [y_i - \varepsilon - \xi_i^* - \langle w, x_i \rangle - b]
\]
SVM Regression (ε-insensitive loss)

• First order conditions

\[ \partial_w L = 0 = w + \sum_i [\alpha_i - \alpha_i^*] x_i \]

\[ \partial_b L = 0 = \sum_i [\alpha_i - \alpha_i^*] \]

\[ \partial_{\xi_i} L = 0 = C - \eta_i - \alpha_i \]

\[ \partial_{\xi_i^*} L = 0 = C - \eta_i^* - \alpha_i^* \]

• Dual problem

\[
\text{minimize} \quad \frac{1}{2}(\alpha - \alpha^*)^T K(\alpha - \alpha^*) + \epsilon 1^T (\alpha + \alpha^*) + y^T (\alpha - \alpha^*) \\
\text{subject to} \quad 1^T (\alpha - \alpha^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]
\]
Properties

• Ignores ‘typical’ instances with small error
• Only upper or lower bound active at any time
• QP in 2n variables as cheap as SVM problem
• Robustness with respect to outliers
  • l1 loss yields same problem without epsilon
  • Huber’s robust loss yields similar problem but with added quadratic penalty on coefficients
Regression example

Figure 9.3
From top to bottom: approximation of the function sinc $x$ with precisions $\varepsilon = 0.1$, $0.2$, and $0.5$. The solid top and dashed bottom lines indicate the size of the $\varepsilon$-tube, here drawn around the target function sinc $x$. The dotted line between them is the regression function.
Regression example

Figure 9.3 From top to bottom: approximation of the function sinc $\times$ with precisions $\varepsilon = 0.1$, $0.2$, and $0.5$. The solid top and dashed bottom lines indicate the size of the $\varepsilon$-tube, here drawn around the target function sinc $\times$. The dotted line between them is the regression function.
Regression example

From top to bottom: approximation of the function sinc $x$ with precisions $\varepsilon = 0.1$, $0.2$, and $0.5$. The solid top and dashed bottom lines indicate the size of the $\varepsilon$-tube, here drawn around the target function sinc $x$. The dotted line between them is the regression function.
9.3 $\nu$-SV Regression

Figure 9.4 Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation of sinc $x$ (dotted line) with $\epsilon = 0.1$, 0.2, and 0.5. Note the decrease in the number of SVs.

The parameter $\epsilon$ of the $\epsilon$-insensitive loss is useful if the desired accuracy of the approximation can be specified beforehand. In some cases, however, we just want the estimate to be as accurate as possible, without having to commit ourselves to as specific level of accuracy a priori. We describe an modification of the $\epsilon$-SVR algorithm, called $\nu$-SVR, which automatically computes $\epsilon$.

To estimate functions (9.2) from empirical data (9.3) we proceed as follows. At each point $x_i$, we allow error $\epsilon$. Everything above $\epsilon$ is captured in slack variables $\xi^*(i)$, which are penalized in the objective function via a regularization constant $C$, chosen a priori. The size of $\epsilon$ is traded off against model complexity and slack variables via a constant $\nu \geq 0$:

$$\min_{w \in H}, \xi^*(i) \in \mathbb{R}^m, \epsilon, b \in \mathbb{R}$$

subject to

$$\langle w, x_i \rangle + b - y_i \leq \epsilon + \xi_i$$ (9.32)

$$y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi^*_i$$ (9.33)

$$\xi^*_i \geq 0, \epsilon \geq 0.$$ (9.34)
Huber’s robust loss

\[ l(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| < 1 \\ |y - f(x)| - \frac{1}{2} & \text{otherwise} \end{cases} \]
Novelty Detection
Basic Idea

Data
Observations \((x_i)\) generated from some \(P(x)\), e.g.,
- network usage patterns
- handwritten digits
- alarm sensors
- factory status

Task
Find unusual events, clean database, distinguish typical examples.
Applications

Network Intrusion Detection
Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else unusual on the network.

Jet Engine Failure Detection
You can’t destroy jet engines just to see how they fail.

Database Cleaning
We want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album.

Fraud Detection
Credit Cards, Telephone Bills, Medical Records

Self calibrating alarm devices
Car alarms (adjusts itself to where the car is parked), home alarm (furniture, temperature, windows, etc.)
**Novelty Detection via Density Estimation**

**Key Idea**
- Novel data is one that we don’t see frequently.
- It must lie in low density regions.

**Step 1: Estimate density**
- Observations $x_1, \ldots, x_m$
- Density estimate via Parzen windows

**Step 2: Thresholding the density**
- Sort data according to density and use it for rejection
- Practical implementation: compute

$$p(x_i) = \frac{1}{m} \sum_{j} k(x_i, x_j) \text{ for all } i$$

and sort according to magnitude.
- Pick smallest $p(x_i)$ as novel points.
Order Statistics of Densities

![Graph of unnormalized density vs. rank](image-url)

- X-axis: Rank
- Y-axis: Unnormalized Density
- The graph shows a smooth increase in density as the rank increases.
Typical Data

3 4 8 6 1 1 3 6
0 0 4 7 1 4 4 2
6 0 4 3 3 7 4 1
3 5 0 0 2 1 0 0
1 7 9 2 0 6 0 0
A better way

Problems

- We do not care about estimating the density properly in regions of high density (waste of capacity).
- We only care about the relative density for thresholding purposes.
- We want to eliminate a certain fraction of observations and tune our estimator specifically for this fraction.

Solution

- Areas of low density can be approximated as the level set of an auxiliary function. No need to estimate \( p(x) \) directly — use proxy of \( p(x) \).
- Specifically: find \( f(x) \) such that \( x \) is novel if \( f(x) \leq c \) where \( c \) is some constant, i.e. \( f(x) \) describes the amount of novelty.
Problems with density estimation

- Exponential Family for density estimation
  \[ p(x|\theta) = \exp \left( \langle \phi(x), \theta \rangle - g(\theta) \right) \]

- MAP estimation
  \[
  \minimize_{\theta} \sum_i g(\theta) - \langle \phi(x_i), \theta \rangle + \frac{1}{2\sigma^2} \|\theta\|^2
  \]

Advantages
- Convex optimization problem
- Concentration of measure

Problems
- Normalization \( g(\theta) \) may be painful to compute
- For density estimation we need no normalized \( p(x|\theta) \)
- No need to perform particularly well in high density regions
Thresholding
Optimization Problem

MAP
\[
\sum_{i=1}^{m} - \log p(x_i | \theta) + \frac{1}{2\sigma^2} \| \theta \|^2
\]

Novelty
\[
\sum_{i=1}^{m} \max \left( - \log \frac{p(x_i | \theta)}{\exp(\rho - g(\theta))}, 0 \right) + \frac{1}{2} \| \theta \|^2
\]
\[
\sum_{i=1}^{m} \max(\rho - \langle \phi(x_i), \theta \rangle, 0) + \frac{1}{2} \| \theta \|^2
\]

Advantages
- No normalization \( g(\theta) \) needed
- No need to perform particularly well in high density regions (estimator focuses on low-density regions)
- Quadratic program
Idea Find hyperplane, given by \( f(x) = \langle w, x \rangle + b = 0 \) that has maximum distance from origin yet is still closer to the origin than the observations.

Hard Margin

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 \\
\text{subject to} & \quad \langle w, x_i \rangle \geq 1
\end{align*}
\]

Soft Margin

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{subject to} & \quad \langle w, x_i \rangle \geq 1 - \xi_i \\
& \quad \xi_i \geq 0
\end{align*}
\]
Optimization Problem

Primal Problem

minimize \[ \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} \xi_i \]

subject to \( \langle w, x_i \rangle - 1 + \xi_i \geq 0 \) and \( \xi_i \geq 0 \)

Lagrange Function \( L \)

- Subtract constraints, multiplied by Lagrange multipliers (\( \alpha_i \) and \( \eta_i \)), from Primal Objective Function.
- Lagrange function \( L \) has **saddlepoint** at optimum.

\[
L = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left( \langle w, x_i \rangle - 1 + \xi_i \right) - \sum_{i=1}^{m} \eta_i \xi_i
\]

subject to \( \alpha_i, \eta_i \geq 0 \).
Dual Problem

Optimality Conditions

\[ \partial_w L = w - \sum_{i=1}^{m} \alpha_i x_i = 0 \implies w = \sum_{i=1}^{m} \alpha_i x_i \]

\[ \partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \implies \alpha_i \in [0, C] \]

Now substitute the optimality conditions back into \( L \).

Dual Problem

minimize \( \frac{1}{2} \sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i \)

subject to \( \alpha_i \in [0, C] \)

All this is only possible due to the convexity of the primal problem.
Minimum enclosing ball

- Observations on surface of ball
- Find minimum enclosing ball
- Equivalent to single class SVM

Figure 8.3

For RBF kernels, which depend only on $x - x'$, $k(x, x)$ is constant, and the mapped data points thus lie on a hypersphere in feature space. In this case, finding the smallest sphere enclosing the data is equivalent to maximising the margin of separation from the origin (cf. Figure 8.2).

8.4 Optimization

The previous section formulated quadratic programs (QPs) for computing regions that capture a certain fraction of the data. These constrained optimization problems can be solved via an off-the-shelf QP package (cf. Chapter 6). In the present section, however, we describe an algorithm which takes advantage of the precise form of the QPs [459], which is an adaptation of the SMO (Sequential Minimal Optimization) algorithm [392]. Although most of the material on implementations is in Chapter 10, we will spend a few minutes to describe the single class algorithm here. Further information on SMO in general can be found in Section 10.5; additional information on single-class SVM implementations, and specifically on variants which work in an online setting, can be found in Section 10.6.3.

The SMO algorithm has been reported to work well in C-SV classification, to which the structure of the present optimization problem, which uses $\nu$ instead of $C$, is quite similar. The dual problem has only one equality constraint (8.15), just as the dual of C-SV classification (7.37).
Adaptive thresholds

Problem

- Depending on $C$, the number of novel points will vary.
- We would like to **specify the fraction** $\nu$ beforehand.

Solution

Use hyperplane separating data from the origin

$$H := \{x|\langle w, x \rangle = \rho\}$$

where the threshold $\rho$ is **adaptive**.

Intuition

- Let the hyperplane shift by shifting $\rho$
- Adjust it such that the 'right' number of observations is considered novel.
- Do this automatically
Optimization Problem

Primal Problem

\[
\text{minimize } \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu \rho \\
\text{where } \langle w, x_i \rangle - \rho + \xi_i \geq 0 \\
\xi_i \geq 0
\]

Dual Problem

\[
\text{minimize } \frac{1}{2} \sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle \\
\text{where } \alpha_i \in [0, 1] \text{ and } \sum_{i=1}^{m} \alpha_i = \nu m.
\]
The $\nu$-property theorem

• Optimization problem

$$\min_w \frac{1}{2} \|w\|^2 + \sum_{i=1}^{m} \xi_i - m \nu \rho$$

subject to $\langle w, x_i \rangle \geq \rho - \xi_i$ and $\xi_i \geq 0$

• Solution satisfies
  • At most a fraction of $\nu$ points are novel
  • At most a fraction of $(1-\nu)$ points aren’t novel
  • Fraction of points on boundary vanishes for large $m$ (for non-pathological kernels)
Proof

- Move boundary at optimality
  - For smaller threshold \( m^- \) points on wrong side of margin contribute \( \delta(m^- - \nu m) \leq 0 \)
  - For larger threshold \( m^+ \) points not on ‘good’ side of margin yield \( \delta(m^+ - \nu m) \geq 0 \)
- Combining inequalities
  \[ \frac{m^-}{m} \leq \nu \leq \frac{m^+}{m} \]
- Margin set of measure 0
Proposition 8.3 (ν-Property)
Assume the solution of (8.6), (8.7) satisfies \( \rho \neq 0 \).

The following statements hold:

(i) \( \nu \) is an upper bound on the fraction of outliers.

(ii) \( \nu \) is a lower bound on the fraction of SVs.

(iii) Suppose the data (8.32) were generated independently from a distribution \( P(x) \) which does not contain discrete components. Suppose, moreover, that the kernel is analytic and non-constant. With probability 1, asymptotically, \( \nu \) equals both the fraction of SVs and the fraction of outliers.

The proof can be found in [459]. The result also applies to the soft margin ball algorithm of [524], provided that it is stated in the \( \nu \)-parameterization given in (8.17).

Figure 8.5 displays a 2-D toy example, illustrating how the choice of \( \nu \) and the kernel width influence the solution.

<table>
<thead>
<tr>
<th>( \nu ), width ( c )</th>
<th>0.5, 0.5</th>
<th>0.5, 0.5</th>
<th>0.1, 0.5</th>
<th>0.5, 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>frac. SVs/OLs</td>
<td>0.54, 0.43</td>
<td>0.59, 0.47</td>
<td>0.24, 0.03</td>
<td>0.65, 0.38</td>
</tr>
<tr>
<td>margin ( \rho/|w| )</td>
<td>0.84</td>
<td>0.70</td>
<td>0.62</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Toy example

threshold and smoothness requirements
Better estimates since we only optimize in low density regions.

Specifically tuned for small number of outliers.

Only estimates of a level-set.

For $\nu = 1$ we get the Parzen-windows estimator back.
Classification with the \( \nu \)-trick

Figure 7.9 Toy problem (task: separate circles from disks) solved using \( \nu \)-SV classification, with parameter values ranging from \( \nu = 0.1 \) (top left) to \( \nu = 0.8 \) (bottom right). The larger we make \( \nu \), the more points are allowed into the margin (depicted by dotted lines). Results are shown for a Gaussian kernel, \( k(x, x') = \exp(-\|x - x'\|^2) \).

Table 7.1 Fractions of errors and SVs, along with the margins of class separation, for the toy example in Figure 7.9.

<table>
<thead>
<tr>
<th>( \nu )</th>
<th>Fraction of errors</th>
<th>Fraction of SVs</th>
<th>Margin ( \rho / |w| )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.00</td>
<td>0.29</td>
<td>0.005</td>
</tr>
<tr>
<td>0.2</td>
<td>0.07</td>
<td>0.36</td>
<td>0.018</td>
</tr>
<tr>
<td>0.3</td>
<td>0.25</td>
<td>0.43</td>
<td>0.115</td>
</tr>
<tr>
<td>0.4</td>
<td>0.32</td>
<td>0.46</td>
<td>0.156</td>
</tr>
<tr>
<td>0.5</td>
<td>0.39</td>
<td>0.57</td>
<td>0.364</td>
</tr>
<tr>
<td>0.6</td>
<td>0.50</td>
<td>0.68</td>
<td>0.419</td>
</tr>
<tr>
<td>0.7</td>
<td>0.61</td>
<td>0.79</td>
<td>0.461</td>
</tr>
<tr>
<td>0.8</td>
<td>0.71</td>
<td>0.86</td>
<td>0.546</td>
</tr>
</tbody>
</table>

Note that \( \nu \) upper bounds the fraction of errors and lower bounds the fraction of SVs, and that increasing \( \nu \), i.e., allowing more errors, increases the margin.

Slightly more complicated. We consider the Lagrangian

\[
L(w, \xi, b, \rho, \alpha, \beta, \delta) = \frac{1}{2} \|w\|^2 - \nu \rho + \frac{1}{m} \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \left( \alpha_i (y_i (\langle x_i, w \rangle + b) - \rho + \xi_i) \right) + \beta_i \xi_i - \delta \rho,
\]

(7.44)

using multipliers \( \alpha_i, \beta_i, \delta \geq 0 \). This function has to be minimized with respect to the primal variables \( w, \xi, b, \rho \), and maximize with respect to the dual variables \( \alpha, \beta, \delta \). To eliminate the former, we compute the corresponding partial derivatives and set them to 0, obtaining the following conditions:

\[
w = \sum_{i=1}^{m} \alpha_i y_i x_i,
\]

(7.45)
Convex Optimization
Selecting Variables
Constrained Quadratic Program

- Optimization Problem
  \[ \min_{\alpha} \frac{1}{2} \alpha^\top Q \alpha + l^\top \alpha \text{ subject to } C \alpha + b \leq 0 \]

- Support Vector classification
- Support Vector regression
- Novelty detection

- Solving it
  - Off the shelf solvers for small problems
  - Solve sequence of subproblems
  - Optimization in primal space (the \( w \) space)
Subproblems

• Original optimization problem

\[ \begin{align*}
\text{minimize} \quad & \frac{1}{2} \alpha^\top Q\alpha + l^\top \alpha \\
\text{subject to} \quad & C\alpha + b \leq 0
\end{align*} \]

• Key Idea - solve subproblems one at a time and decompose into active and fixed set \( \alpha = (\alpha_a, \alpha_f) \)

\[ \begin{align*}
\text{minimize} \quad & \frac{1}{2} \alpha_a^\top Q_{aa}\alpha_a + \left[ l_a + Q_{af}\alpha_f \right]^\top \alpha_a \\
\text{subject to} \quad & C_a\alpha_a + [b + C_f\alpha_f] \leq 0
\end{align*} \]

• Subproblem is again a convex problem

• Updating subproblems is cheap
Picking observations

\[ w = \sum_i y_i \alpha_i x_i \]

\[ \alpha_i [y_i [\langle w, x_i \rangle + b] + \xi_i - 1] = 0 \]

\[ \eta_i \xi_i = 0 \]

- Most violated margin condition
- Points on the boundary
- Points with nonzero Lagrange multiplier that are correct

\[ \alpha_i = 0 \implies y_i [\langle w, x_i \rangle + b] \geq 1 \]

\[ 0 < \alpha_i < C \implies y_i [\langle w, x_i \rangle + b] = 1 \]

\[ \alpha_i = C \implies y_i [\langle w, x_i \rangle + b] \leq 1 \]
Selecting variables

- Incrementally increase (chunking)
- Select promising subset of actives (SVMLight)
- Select pairs of variables (SMO)
## Being smart about hardware

### Data flow from disk to CPU

![Diagram showing data flow from disk to CPU](image)

- **Data**: Disk
- **Cached Data (Working Set)**: RAM
- **Parameter**

### IO speeds

<table>
<thead>
<tr>
<th>System</th>
<th>Capacity</th>
<th>Bandwidth</th>
<th>IOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
<td>3TB</td>
<td>150MB/s</td>
<td>$10^2$</td>
</tr>
<tr>
<td>SSD</td>
<td>256GB</td>
<td>500MB/s</td>
<td>$5 \cdot 10^4$</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB</td>
<td>30GB/s</td>
<td>$10^8$</td>
</tr>
<tr>
<td>Cache</td>
<td>16MB</td>
<td>100GB/s</td>
<td>$10^9$</td>
</tr>
</tbody>
</table>
Being smart about hardware

- **Data flow from disk to CPU**

![Data flow diagram](Image)

- **IO speeds**

<table>
<thead>
<tr>
<th>System</th>
<th>Capacity</th>
<th>Bandwidth</th>
<th>IOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
<td>3TB</td>
<td>150MB/s</td>
<td>$10^2$</td>
</tr>
<tr>
<td>SSD</td>
<td>256GB</td>
<td>500MB/s</td>
<td>$5 \cdot 10^4$</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB</td>
<td>30GB/s</td>
<td>$10^8$</td>
</tr>
<tr>
<td>Cache</td>
<td>16MB</td>
<td>100GB/s</td>
<td>$10^9$</td>
</tr>
</tbody>
</table>

**Carnegie Mellon University**
Runtime Example
(Matsushima, Vishwanathan, Smola, 2012)
Gradient Descent

- Assume we can optimize in feature space directly
- Minimize regularized risk
- Compute gradient and update
- This fails in narrow canyons
- Wasteful if we have lots of similar data

Gradient Descent

\[ R[w] = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, w) + \frac{\lambda}{2} \| w \|^2 \]

- Compute gradient \( g = \partial_w R[w] \)
  and update \( w \leftarrow w - \gamma g \)

Schölkopf and Smola: Learning with Kernels — Confidential draft, please do not circulate — 2012/01/14 15:35

6.2.3 Convergence Properties of Gradient Descent

Let us analyze the convergence properties of Algorithm 6.3 in more detail. To keep matters simple, we assume that \( f \) is a quadratic function, i.e.

\[ f(x) = \frac{1}{2} (x - x^*)^\top K (x - x^*) + c_0 \]
Stochastic gradient descent

- Empirical risk as expectation
  \[
  \frac{1}{m} \sum_{i=1}^{m} l (y_i - \langle \phi(x_i), w \rangle) = \mathbb{E}_{i \sim \{1, \ldots, m\}} [l (y_i - \langle \phi(x_i), w \rangle)]
  \]

- Stochastic gradient descent (pick random x, y)
  \[
  w_{t+1} \leftarrow w_t - \eta_t \partial_w (y_t, \langle \phi(x_t), w_t \rangle)
  \]

- Often we require that parameters are restricted to some convex set X, hence we project on it
  \[
  w_{t+1} \leftarrow \pi_x [w_t - \eta_t \partial_w (y_t, \langle \phi(x_t), w_t \rangle)]
  \]
  here \( \pi_X(w) = \arg\min_{x \in X} ||x - w|| \)
Some applications

- Classification
- Soft margin loss \( l(x, y, w) = \max(0, 1 - y \langle w, \phi(x) \rangle) \)
- Logistic loss \( l(x, y, w) = \log (1 + \exp (-y \langle w, \phi(x) \rangle)) \)
- Regression
- Quadratic loss \( l(x, y, w) = (y - \langle w, \phi(x) \rangle)^2 \)
- \( l_1 \) loss \( l(x, y, w) = \|y - \langle w, \phi(x) \rangle\| \)
- Huber’s loss \( l(x, y, w) = \begin{cases} \frac{1}{2\sigma^2} (y - \langle w, \phi(x) \rangle)^2 & \text{if } |y - \langle w, \phi(x) \rangle| \leq \sigma \\ \frac{1}{\sigma} |y - \langle w, \phi(x) \rangle| - \frac{1}{2} & \text{if } |y - \langle w, \phi(x) \rangle| > \sigma \end{cases} \)
- Novelty detection \( l(x, w) = \max(0, 1 - \langle w, \phi(x) \rangle) \)

... and many more
Convergence in Expectation

\[ \mathbb{E}_{\bar{\theta}} [l(\bar{\theta})] - l^* \leq \frac{R^2 + L^2 \sum_{t=0}^{T-1} \eta_t^2}{2 \sum_{t=0}^{T-1} \eta_t} \]

where

\[ l(\theta) = \mathbb{E}_{(x,y)} [l(y, \langle \phi(x), \theta \rangle)] \quad \text{and} \quad l^* = \inf_{\theta \in \mathcal{X}} l(\theta) \quad \text{and} \quad \bar{\theta} = \frac{\sum_{t=0}^{T-1} \theta_t \eta_t}{\sum_{t=0}^{T-1} \eta_t} \]

• Proof

Show that parameters converge to minimum

\[ \theta^* \in \arg\min_{\theta \in \mathcal{X}} l(\theta) \quad \text{and} \quad r_t := ||\theta^* - \theta_t|| \]

from Nesterov and Vial
Proof

\[ r_{t+1}^2 = \| \pi X [\theta_t - \eta_t g_t] - \theta^* \|^2 \]
\[ \leq \| \theta_t - \eta_t g_t - \theta^* \|^2 \]
\[ = r_t^2 + \eta_t^2 \| g_t \|^2 - 2\eta_t \langle \theta_t - \theta^*, g_t \rangle \]

hence \( \mathbb{E} \left[ r_{t+1}^2 - r_t^2 \right] \leq \eta_t^2 L^2 + 2\eta_t \left[ l^* - \mathbb{E}[l(\theta_t)] \right] \]
\[ \leq \eta_t^2 L^2 + 2\eta_t \left[ l^* - \mathbb{E}[l(\bar{\theta})] \right] \] by convexity

- Summing over inequality for \( t \) proves claim
- This yields randomized algorithm for minimizing objective functions (try log times and pick the best / or average median trick)
Rates

• **Guarantee**

\[ E_{\tilde{\theta}} [l(\tilde{\theta})] - l^* \leq \frac{R^2 + L^2 \sum_{t=0}^{T-1} \eta_t^2}{2 \sum_{t=0}^{T-1} \eta_t} \]

• If we know R, L, T pick constant learning rate

\[ \eta = \frac{R}{L\sqrt{T}} \text{ and hence } E_{\tilde{\theta}}[l(\tilde{\theta})] - l^* \leq \frac{R[1 + 1/T]L}{2\sqrt{T}} < \frac{LR}{\sqrt{T}} \]

• If we don’t know T pick \( \eta_t = O(t^{-\frac{1}{2}}) \)

This costs us an additional log term

\[ E_{\tilde{\theta}}[l(\tilde{\theta})] - l^* = O \left( \frac{\log T}{\sqrt{T}} \right) \]
Strong Convexity

\[ l_i(\theta') \geq l_i(\theta) + \langle \partial_\theta l_i(\theta), \theta' - \theta \rangle + \frac{1}{2} \lambda \|\theta - \theta'\|^2 \]

- Use this to bound the expected deviation

\[
\begin{align*}
    r_{t+1}^2 &\leq r_t^2 + \eta_t^2 \|g_t\|^2 - 2\eta_t \langle \theta_t - \theta^*, g_t \rangle \\
    &\leq r_t^2 + \eta_t^2 L^2 - 2\eta_t [l_t(\theta_t) - l_t(\theta^*)] - 2\lambda \eta_t r_k^2 \\
    \text{hence } E[r_{t+1}^2] &\leq (1 - \lambda h_t) E[r_t^2] - 2\eta_t \left[ E[l(\theta_t)] - l^* \right]
\end{align*}
\]

- Exponentially decaying averaging

\[
\bar{\theta} = \frac{1 - \sigma}{1 - \sigma T} \sum_{t=0}^{T-1} \sigma^{T-1-t} \theta_t
\]

and plugging this into the discrepancy yields

\[
l(\bar{\theta}) - l^* \leq \frac{2L^2}{\lambda T} \log \left[ 1 + \frac{\lambda R T^{1/2}}{2L} \right] \quad \text{for } \eta = \frac{2}{\lambda T} \log \left[ 1 + \frac{\lambda R T^{1/2}}{2L} \right]
\]
More variants

• Adversarial guarantees

\[ \theta_{t+1} \leftarrow \pi_x [\theta_t - \eta_t \partial \theta (y_t, \langle \phi(x_t), \theta_t \rangle)] \]

has low regret (average instantaneous cost) for arbitrary orders (useful for game theory)

• Ratliff, Bagnell, Zinkevich

\( O(t^{-\frac{1}{2}}) \) learning rate

• Shalev-Shwartz, Srebro, Singer (Pegasos)

\( O(t^{-1}) \) learning rate (but need constants)

• Bartlett, Rakhlin, Hazan

(add strong convexity penalty)
Regularization
Problems with Kernels

Myth
Support Vectors work because they map data into a high-dimensional feature space.

And your statistician (Bellmann) told you . . .
The higher the dimensionality, the more data you need

Example: Density Estimation
Assuming data in $[0, 1]^m$, 1000 observations in $[0, 1]$ give you on average 100 instances per bin (using binsize $0.1^m$) but only $\frac{1}{100}$ instances in $[0, 1]^5$.

Worrying Fact
Some kernels map into an infinite-dimensional space, e.g., $k(x, x') = \exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right)$

Encouraging Fact
SVMs work well in practice . . .
The Truth is in the Margins

Maybe the maximum margin requirement is what saves us when finding a classifier, i.e., we minimize \( \|w\|^2 \).

Risk Functional

Rewrite the optimization problems in a unified form

\[
R_{\text{reg}}[f] = \sum_{i=1}^{m} c(x_i, y_i, f(x_i)) + \Omega[f]
\]

\( c(x, y, f(x)) \) is a loss function and \( \Omega[f] \) is a regularizer.

- \( \Omega[f] = \frac{\lambda}{2} \|w\|^2 \) for linear functions.
- For classification \( c(x, y, f(x)) = \max(0, 1 - yf(x)) \).
- For regression \( c(x, y, f(x)) = \max(0, |y - f(x)| - \epsilon) \).
Typical SVM loss

Soft Margin Loss

$\varepsilon$-insensitive Loss
Soft Margin Loss

Original Optimization Problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{subject to} & \quad y_i f(x_i) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } 1 \leq i \leq m
\end{align*}
\]

Regularization Functional

\[
\begin{align*}
\text{minimize} & \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{m} \max(0, 1 - y_i f(x_i))
\end{align*}
\]

- For fixed \( f \), clearly \( \xi_i \geq \max(0, 1 - y_i f(x_i)) \).
- For \( \xi > \max(0, 1 - y_i f(x_i)) \) we can decrease it such that the bound is matched and improve the objective function.
- Both methods are equivalent.
What we really wanted . . .
Find some $f(x)$ such that the expected loss $E[c(x, y, f(x))]$ is small.

What we ended up doing . . .
Find some $f(x)$ such that the empirical average of the expected loss $E_{\text{emp}}[c(x, y, f(x))]$ is small.

$$E_{\text{emp}}[c(x, y, f(x))] = \frac{1}{m} \sum_{i=1}^{m} c(x_i, y_i, f(x_i))$$

However, just minimizing the empirical average does not guarantee anything for the expected loss (overfitting).

Safeguard against overfitting
We need to constrain the class of functions $f \in \mathcal{F}$ somehow. Adding $\Omega[f]$ as a penalty does exactly that.
Some regularization ideas

**Small Derivatives**
We want to have a function $f$ which is smooth on the entire domain. In this case we could use

$$
\Omega[f] = \int_X \|\partial_x f(x)\|^2 \, dx = \langle \partial_x f, \partial_x f \rangle.
$$

**Small Function Values**
If we have no further knowledge about the domain $X$, minimizing $\|f\|^2$ might be sensible, i.e.,

$$
\Omega[f] = \|f\|^2 = \langle f, f \rangle.
$$

**Splines**
Here we want to find $f$ such that both $\|f\|^2$ and $\|\partial_x^2 f\|^2$ are small. Hence we can minimize

$$
\Omega[f] = \|f\|^2 + \|\partial_x^2 f\|^2 = \langle (f, \partial_x^2 f), (f, \partial_x^2 f) \rangle.$$

Regularization Operators
We map \( f \) into some \( Pf \), which is small for desirable \( f \) and large otherwise, and minimize

\[
\Omega[f] = \|Pf\|^2 = \langle Pf, Pf \rangle.
\]

For all previous examples we can find such a \( P \).

Function Expansion for Regularization Operator
Using a linear function expansion of \( f \) in terms of some \( f_i \), that is for \( f(x) = \sum_i \alpha_i f_i(x) \) we can compute

\[
\Omega[f] = \left\langle P \sum_i \alpha_i f_i(x), P \sum_j \alpha_j f_i(x) \right\rangle = \sum_{i,j} \alpha_i \alpha_j \langle Pf_i, Pf_j \rangle.
\]
Regularization and Kernels

Regularization for $\Omega[f] = \frac{1}{2}||w||^2$

$$w = \sum_i \alpha_i \Phi(x_i) \implies ||w||^2 = \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

This looks very similar to $\langle Pf_i, Pf_j \rangle$.

**Key Idea**

So if we could find a $P$ and $k$ such that

$$k(x, x') = \langle Pk(x, \cdot), Pk(x', \cdot) \rangle$$

we could show that using a kernel means that we are minimizing the empirical risk plus a regularization term.

**Solution: Greens Functions**

A sufficient condition is that $k$ is the Greens Function of $P^*P$, that is $\langle P^*Pk(x, \cdot), f(\cdot) \rangle = f(x)$.

One can show that this is necessary and sufficient.
Kernels from Regularization Operators:
Given an operator $P^*P$, we can find $k$ by solving the self consistency equation

$$\langle Pk(x, \cdot), Pk(x', \cdot) \rangle = k^\top(x, \cdot)(P^*P)k(x', \cdot) = k(x, x')$$

and take $f$ to be the span of all $k(x, \cdot)$. So we can find $k$ for a given measure of smoothness.

Regularization Operators from Kernels:
Given a kernel $k$, we can find some $P^*P$ for which the self consistency equation is satisfied. So we can find a measure of smoothness for a given $k$. 
Effective Function Class

Keeping $\Omega[f]$ small means that $f(x)$ cannot take on arbitrary function values. Hence we study the function class

$$\mathcal{F}_C = \left\{ f \left| \frac{1}{2} \langle Pf, Pf \rangle \leq C \right. \right\}$$

Example

For $f = \sum_i \alpha_i k(x_i, x)$ this implies $\frac{1}{2} \alpha^\top K \alpha \leq C$.

Kernel Matrix

$$K = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix}$$

Coefficients

Function Values
Goal
Find measure of smoothness that depends on the frequency properties of $f$ and not on the position of $f$.

A Hint: Rewriting $\| f \|^2 + \| \partial_x f \|^2$

Notation: $\tilde{f}(\omega)$ is the Fourier transform of $f$.

\[
\| f \|^2 + \| \partial_x f \|^2 = \int |f(x)|^2 + |\partial_x f(x)|^2 \, dx
\]
\[
= \int |\tilde{f}(\omega)|^2 + \omega^2 |\tilde{f}(\omega)|^2 \, d\omega
\]
\[
= \int \frac{|\tilde{f}(\omega)|^2}{p(\omega)} \, d\omega \quad \text{where} \quad p(\omega) = \frac{1}{1 + \omega^2}.
\]

Idea
Generalize to arbitrary $p(\omega)$, i.e. $\Omega[f] := \frac{1}{2} \int \frac{|\tilde{f}(\omega)|^2}{p(\omega)} \, d\omega$
Theorem

For regularization functionals \( \Omega[f] := \frac{1}{2} \int \frac{|\hat{f}(\omega)|^2}{p(\omega)} d\omega \) the self-consistency condition

\[
\langle Pk(x, \cdot), Pk(x', \cdot) \rangle = k^\top(x, \cdot)(P^* P)k(x', \cdot) = k(x, x')
\]

is satisfied if \( k \) has \( p(\omega) \) as its Fourier transform, i.e.,

\[
k(x, x') = \int \exp(-i\langle \omega, (x - x') \rangle)p(\omega)d\omega
\]

Consequences

- small \( p(\omega) \) correspond to high penalty (regularization).
- \( \Omega[f] \) is translation invariant, that is \( \Omega[f(\cdot)] = \Omega[f(\cdot - x)] \).
Examples

Laplacian Kernel

\[ k(x, x') = \exp(-\|x - x'\|) \]
\[ p(\omega) \propto (1 + \|\omega\|^2)^{-1} \]

Gaussian Kernel

\[ k(x, x') = e^{-\frac{1}{2} \sigma^{-2} \|x - x'\|^2} \]
\[ p(\omega) \propto e^{-\frac{1}{2} \sigma^2 \|\omega\|^2} \]

Fourier transform of \( k \) shows regularization properties. The more rapidly \( p(\omega) \) decays, the more high frequencies are filtered out.
Fourier transform is sufficient to check whether $k(x, x')$ satisfies Mercer’s condition: only check if $\tilde{k}(\omega) \geq 0$.

Example: $k(x, x') = \text{sinc}(x - x')$.

$\tilde{k}(\omega) = \chi_{[-\pi, \pi]}(\omega)$, hence $k$ is a proper kernel.

Width of kernel often more important than type of kernel (short range decay properties matter).

Convenient way of incorporating prior knowledge, e.g.: for speech data we could use the autocorrelation function.

Sum of derivatives becomes polynomial in Fourier space.
Polynomial Kernels

Functional Form

\[ k(x, x') = \kappa(\langle x, x' \rangle) \]

Series Expansion

Polynomial kernels admit an expansion in terms of Legendre polynomials \((L_n^N : \text{order } n \text{ in } \mathbb{R}^N)\).

\[ k(x, x') = \sum_{n=0}^{\infty} b_n L_n(\langle x, x' \rangle) \]

Consequence:

\(L_n\) (and their rotations) form an orthonormal basis on the unit sphere, \(P^*P\) is rotation invariant, and \(P^*P\) is diagonal with respect to \(L_n\). In other words

\[ (P^*P)L_n(\langle x, \cdot \rangle) = b_n^{-1}L_n(\langle x, \cdot \rangle) \]
Decay properties of $b_n$ determine smoothness of functions specified by $k(\langle x, x' \rangle)$.

For $N \to \infty$ all terms of $L^N_n$ but $x^n$ vanish, hence a Taylor series $k(x, x') = \sum_i a_i \langle x, x' \rangle^i$ gives a good guess.

**Inhomogeneous Polynomial**

\[
k(x, x') = (\langle x, x' \rangle + 1)^p
\]

\[
a_n = \binom{p}{n} \text{ if } n \leq p
\]

**Vovk's Real Polynomial**

\[
k(x, x') = \frac{1 - \langle x, x' \rangle^p}{1 - (\langle x, x' \rangle)}
\]

\[
a_n = 1 \text{ if } n < p
\]
Mini Summary

Regularized Risk Functional
- From Optimization Problems to Loss Functions
- Regularization
- Safeguard against Overfitting

Regularization and Kernels
- Examples of Regularizers
- Regularization Operators
- Greens Functions and Self Consistency Condition

Fourier Regularization
- Translation Invariant Regularizers
- Regularization in Fourier Space
- Kernel is inverse Fourier Transformation of Weight

Polynomial Kernels and Series Expansions
Text Analysis
(string kernels)
String Kernel (pre)History
The Kernel Perspective

• Design a kernel implementing good features

\[ k(x, x') = \langle \phi(x), \phi(x') \rangle \text{ and } f(x) = \langle \phi(x), w \rangle = \sum_i \alpha_i k(x_i, x) \]

• Many variants
  • Bag of words (AT&T labs 1995, e.g. Vapnik)
  • Matching substrings (Haussler, Watkins 1998)
  • Spectrum kernel (Leslie, Eskin, Noble, 2000)
  • Suffix tree (Vishwanathan, Smola, 2003)
  • Suffix array (Teo, Vishwanathan, 2006)
  • Rational kernels (Mohri, Cortes, Haffner, 2004 ...)

Carnegie Mellon University
Bag of words

- At least since 1995 known in AT&T labs

\[ k(x, x') = \sum_w n_w(x)n_w(x') \quad \text{and} \quad f(x) = \sum_w \omega_w n_w(x') \]

(to be or not to be) \( \rightarrow \) (be:2, or:1, not:1, to:2)

- Joachims 1998: Use sparse vectors
- Haffner 2001: Inverted index for faster training
- Lots of work on feature weighting (TF/IDF)
- Variants of it deployed in many spam filters
Substring (mis)matching

- Watkins 1998+99 (dynamic alignment, etc)
- Haussler 1999 (convolution kernels)

\[ k(x, x') = \sum_{w \in x} \sum_{w' \in x'} k(w, w') \]

- In general \( O(x \times x') \) runtime
  (e.g. Cristianini, Shawe-Taylor, Lodhi, 2001)
- Dynamic programming solution for pair-HMM
Spectrum Kernel

- Leslie, Eskin, Noble & coworkers, 2002
- Key idea is to focus on features directly
  - Linear time operation to get features
  - Limited amount of mismatch (exponential in number of missed chars)
  - Explicit feature construction (good & fast for DNA sequences)
Suffix Tree Kernel

• Vishwanathan & Smola, 2003 \( (O(x + x') \text{ time}) \)
• Mismatch-free kernel + arbitrary weights

\[ k(x, x') = \sum_w \omega_w n_w(x)n_w(x') \]

• Linear time construction (Ukkonen, 1995)
• Find matches for second string in linear time (Chang & Lawler, 1994)
• Precompute weights on path
Are we done?

- Large vocabulary size
- Need to build dictionary
- Approximate matches are still a problem
- Suffix tree/array is storage inefficient (40-60x)
- Realtime computation
- Memory constraints (keep in RAM)
- Difficult to implement
Multitask Learning
From: bat@kilian@gmail.com
Subject: hey what's up check this meds place out
Date: April 6, 2008 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat@kilian@gmail.com

Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

Get Woodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required. Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
http://jenkins.jsr.c3.blogspot.com

The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

To read the rest of this story, go here:
http://toledo.scout.com/2/769390.html
Multitask Learning
Multitask Learning
Collaborative Classification

- **Primal representation**
  \[ f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle \]

- **Kernel representation**
  \[ k((x, u), (x', u')) = k(x, x')[1 + \delta_{u, u'}] \]

  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem** - dimensionality is \(10^{13}\). That is 40TB of space
Collaborative Classification

- **Primal representation**
  \[ f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle \]

- **Kernel representation**
  \[ k((x, u), (x', u')) = k(x, x')[1 + \delta_{u, u'}] \]
  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem** - dimensionality is \(10^{13}\). That is 40TB of space
**Collaborative Classification**

- **Primal representation**
  \[ f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle \]

- **Kernel representation**
  \[ k((x, u), (x', u')) = k(x, x')[1 + \delta_{u,u'}] \]

  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well...

- **Problem** - dimensionality is \(10^{13}\). That is 40TB of space
Hashing
Hash Kernels

*in the old days
Hey,
please mention subtly during your talk that people should use Yahoo search more often.
Thanks,

*in the old days
Hey,
please mention subtly during your talk that people should use Yahoo* search more often. Thanks,

*in the old days
instance:

Hey,
please mention subtly during your talk that people should use Yahoo search more often. Thanks,

task/user (=barney):

Similar to count hash (Charikar, Chen, Farrach-Colton, 2003)
Advantages of hashing
Advantages of hashing

- No dictionary!
- Content drift is no problem
- All memory used for classification
- Finite memory guarantee (via online learning)
Advantages of hashing

• No dictionary!
• Content drift is no problem
• All memory used for classification
• Finite memory guarantee (via online learning)
• No Memory needed for projection. (vs LSH)
Advantages of hashing

- No dictionary!
- Content drift is no problem
- All memory used for classification
- Finite memory guarantee (via online learning)
- No Memory needed for projection. (vs LSH)
- Implicit mapping into high dimensional space!
Advantages of hashing

• No dictionary!
• Content drift is no problem
• All memory used for classification
• Finite memory guarantee (via online learning)
• No Memory needed for projection. (vs LSH)
• Implicit mapping into high dimensional space!
• It is sparsity preserving! (vs LSH)
Approximate Orthogonality

We can do multi-task learning!
Guarantees

• For a random hash function the inner product vanishes with high probability via
  \[ \Pr\{|\langle w_v, h_u(x) \rangle| > \epsilon\} \leq 2e^{-C\epsilon^2 m} \]

• We can use this for multitask learning

  \textbf{Direct sum in Hilbert Space} \quad \textbf{Sum in Hash Space}

• The hashed inner product is unbiased
  Proof: take expectation over random signs

• The variance is \(O(1/n)\)
  Proof: brute force expansion

• Restricted isometry property (Kumar, Sarlos, Dasgupta 2010)
Spam classification results

N=20M, U=400K
Lazy users ...

Labeled emails per user

number of users

number of labels
Results by user group
Results by user group

labeled emails:
- [0]
- [1]
- [2,3]
- [4,7]
- [8,15]
- [16,31]
- [32,64]
- [64,∞]
- baseline

spam miss-rate (relative to baseline)

b bits in hash-table
Results by user group

![Graph showing the spam miss-rate (relative to baseline) vs. b bits in hash-table for different user groups labeled emails: [0], [1], [2,3], [4,7], [8,15], [16,31], [32,64], [64,∞], and baseline. The y-axis represents the spam miss-rate and the x-axis represents the b bits in hash-table.](graph.png)
Estimation details

• Works best with stochastic gradient descent (or any other primal space method)
• Never instantiate hash map explicitly

\[ f(x) = \langle w, \phi(x) \rangle = \sum_s w[h(s)]n_s(x) \]

• Random memory access pattern (latency)
• Multiclass classification - joint hash
Approximate Matches

- **General idea**
  \[ k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w') \text{ for } |w - w'| \leq \delta \]

- **Simplification**
  - Weigh by mismatch amount \(|w-w'|\)
  - **Map into fragments:** dog -> (*og, d*g, do*)
  - Hash fragments and weigh them based on mismatch amount
  - **Exponential in amount of mismatch**
    But not in alphabet size
Memory access patterns

- Cache size is a few MBs
  Very fast random memory access
- RAM (DDR3 or better) is GBs
  - Fast sequential memory access (burst read)
  - CPU caches memory read from RAM
  - Random memory access is very slow
  - CPU caches memory read from RAM

vector
hashed sequence
Speeding up access

- Key idea - bound the range of $h(i,j)$
  for $j=1$ to $n$ access $h(i,j)$
  $h(i, j) = h(i) + j$

- Linear offset
  bad collisions in $i$
  $h(i, j) = h(i) + h'(j)$

- Sum of hash functions
  bad collisions in $j$
  $h(i, j) = h(i) + \text{OGR}(j)$

- Optimal Golomb Ruler (Langford)
  NP hard in general

- Feistel Network / Cryptography (new)
  $h(i, j) = h(i) + \text{crypt}(j|i)$