Administrative stuff

- Solutions will be posted by tomorrow
- New problem set will be available by tomorrow

- Midterm project presentations are on March 13
  - Describe what you will do
  - Why it’s important
  - What you’ve achieved so far
  - Show why you think you’re going to succeed
  - 10 minutes per team (6 slides maximum)
  - Up to 10 pages supporting documentation
5. (Generalized) Linear Models
(Generalized) Linear Models

- Kernel trick
  - Simple kernels
  - Kernel PCA
  - Mean Classifier
- Support Vectors
  - Support Vector Machine classification
  - Regression
  - Logistic regression
  - Novelty detection
- Gaussian Process Estimation
  - Regression
  - Classification
  - Heteroscedastic Regression
Kernels - a Preview
Solving XOR

- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable
Feature Space Mapping

- Naive Nonlinearization Strategy
  - Express data \( x \) in terms of features \( \phi(x) \)
  - Solve problem in feature space
  - Requires explicit feature computation
- Kernel trick
  - Write algorithm in terms of inner products
  - Replace \( \langle x, x' \rangle \) by \( k(x, x') := \langle \phi(x), \phi(x') \rangle \)
  - Works well for dimension-insensitive methods
  - Kernel matrix \( K \) is positive semidefinite
SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni
Polynomial Kernels

- Linear
  \[ k(x, x') := \langle x, x' \rangle \]

- Quadratic
  \[ k(x, x') := \left\langle \left( x_1^2, x_2^2, \sqrt{2}x_1x_2 \right), \left( x'_1^2, x'_2^2, \sqrt{2}x'_1x'_2 \right) \right\rangle = \langle x, x' \rangle^2 \]

- Homogeneous polynomial
  \[ k(x, x') := \langle x, x' \rangle^p = \sum_{|\alpha|=p} \prod_i \alpha_i! (x_ix'_i)^{\alpha_i} \text{ with } \alpha \in \mathbb{N}_0^d \]

- Inhomogeneous polynomial
  \[ k(x, x') := (\langle x, x' \rangle + c)^p = \sum_{i=0}^{p} \binom{p}{i} \langle x, x' \rangle^i \]
More Kernels

• Gaussian Kernel

\[ k(x, x') := \exp\left(-\gamma \|x - x'\|^2\right) \]

can check that this is convolution of Gaussians

• Brownian Bridge

\[ k(x, x') := \min(x, x') \text{ for } x, x' \geq 0 \]

• Set intersection

\[ k(A, B) := |A \cap B| \]

• Strings, more fancy set kernels, graphs, etc.
Support Vector Machines
Classification

minimize $\frac{1}{2} \|w\|^2$ subject to $y_i [\langle w, x_i \rangle + b] \geq 1$

$\langle w, x_1 \rangle + b = 1$
$\langle w, x_2 \rangle + b = -1$

hence $\langle w, x_1 - x_2 \rangle = 2$

hence $\left< \frac{w}{\|w\|}, x_1 - x_2 \right> = \frac{2}{\|w\|}$

**Support Vectors**

$\{x \mid \langle w, x \rangle + b = -1\}$

$\{x \mid \langle w, x \rangle + b = +1\}$

$\{x \mid \langle w, x \rangle + b = 0\}$

$y_i = -1$

$y_i = +1$

$w$

$\text{margin}$
Support Vectors

![Diagram of Support Vectors]

**dual problem**

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \alpha^T K \alpha - 1^T \alpha \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \\
& \quad \alpha_i \geq 0
\end{align*}
\]

\[K_{ij} = y_i y_j \langle x_i, x_j \rangle\]

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

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 \\
\text{subject to} & \quad y_i [\langle w, x_i \rangle + b] \geq 1
\end{align*}
\]
The Lagrangian $L$ can be written as:

$$L = \sum_{i=1}^{m} \alpha_i (y_i (\langle x_i, w \rangle + b) - 1) - \frac{C}{2} \|w\|^2.$$ 

This function is a composition of two terms: the first term, $\sum_{i=1}^{m} \alpha_i y_i (\langle x_i, w \rangle + b) - \frac{C}{2} \|w\|^2$, is a regularized objective function, and the second term, $-\sum_{i=1}^{m} \alpha_i$, is a Lagrange multiplier term.

The Karush-Kuhn-Tucker (KKT) optimality conditions are given by:

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

$y_i (\langle x_i, w \rangle + b) > 1$ implies $\alpha_i = 0$.

$\alpha_i > 0$ implies $y_i (\langle x_i, w \rangle + b) = 1$.
Properties

• Weight vector $w$ as weighted linear combination of instances
• Only points on margin matter (we can ignore the rest and get same solution)
• Only inner products matter
  • Quadratic program
  • We can replace the inner product by a kernel
• Keeps instances away from the margin

Java demo: http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml
Example
Example

Number of Support Vectors: 3  (-ve: 2, +ve: 1)  Total number of points: 15
Why large margins?

- Maximum robustness relative to uncertainty
- Symmetry breaking
- Independent of correctly classified instances
- Easy to find for easy problems

\[ \Delta x \in H \] is bounded in norm by some \( r > 0 \). Clearly, if we manage to separate the training set with a margin \( \rho > r \), we will correctly classify all test points: Since all training points have a distance of at least \( \rho \) to the hyperplane, the test patterns will still be on the correct side (Figure 7.3, cf. also [146]).

Figure 7.3
Two-dimensional toy example of a classification problem: Separate 'o' from '+' using a hyperplane. Suppose that we add bounded noise to each pattern. If the optimal margin hyperplane has margin \( \rho \), then it is bounded by \( r < \rho \), then the hyperplane will correctly separate even the noisy patterns. Conversely, if we run the perceptron algorithm (which finds some separating hyperplane, but not necessarily the optimal one) on the noisy data, then we would recover the optimal hyperplane in the limit \( r \to \rho \).

If we knew \( \rho \) beforehand, then this could actually be turned into an optimal margin classifier training algorithm, as follows. If we use a \( r \) which is slightly smaller than \( \rho \), then the patterns will be separable with a nonzero margin. In this case, the standard perceptron algorithm can be shown to converge.

1. Rosenblatt's perceptron algorithm [423] is one of the simplest conceivable iterative procedures for computing a separating hyperplane. In its simplest form, it proceeds as follows. We start with an arbitrary weight vector \( w_0 \). At step \( n \in \mathbb{N} \), we consider the training example \((x_n, y_n)\). If it is classified correctly using the current weight vector (i.e., if \( \text{sgn} \langle x_n, w_n - 1 \rangle = y_n \)), we set \( w_n : = w_n - 1 \); otherwise, we set \( w_n : = w_n - 1 + \eta y_i x_i \) (here, \( \eta > 0 \)) is a learning rate. We then loop over patterns repeatedly, until we can complete one full pass through the training set without a single error. The result in weight vector will thus classify all points correctly. Novikoff [369] proved that this procedure terminates, provided that the training set is separable with a nonzero margin.
Inseparable data

Quadratic program has no feasible solution
Adding slack variables

• Hard margin problem

\[
\minimize_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to } y_i [\langle w, x_i \rangle + b] \geq 1
\]

• With slack variables

\[
\minimize_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{subject to } y_i [\langle w, x_i \rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0
\]

problem is always feasible. Proof:

\[
w = 0 \text{ and } b = 0 \text{ and } \xi_i = 1 \quad \text{(also yields upper bound)}
\]
Support Vectors

\[ \{ \mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = -1 \} \]

\[ \{ \mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = +1 \} \]

\[ y_i = -1 \]

\[ y_i = +1 \]

\[ \mathbf{w} \]

\[ \mathbf{x}_1 \]

\[ \mathbf{x}_2 \]

dual problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \alpha^\top K \alpha - \mathbf{1}^\top \alpha \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \\
& \quad \alpha_i \in [0, C]
\end{align*}
\]

\[ K_{ij} = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \]

\[ \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \]

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \xi_i \\
\text{subject to} & \quad y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0
\end{align*}
\]
Classification with errors
Nonlinear separation

- Increasing C allows for more nonlinearities
- Decreases number of errors
- SV boundary need not be contiguous
Loss function point of view

- Constrained quadratic program

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

- Risk minimization setting

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_i \max [0, 1 - y_i [\langle w, x_i \rangle + b]]
\end{align*}
\]

Follows from finding minimal slack variable for given \((w,b)\) pair.
Soft margin as proxy for binary

- Soft margin loss: $\max(0, 1 - yf(x))$
- Binary loss: $\{yf(x) < 0\}$

Diagram:
- Convex upper bound
- Binary loss function
- Margin
More loss functions

• **Logistic** \( \log \left[ 1 + e^{-f(x)} \right] \)

• **Huberized loss**
  \[
  \begin{cases}
  0 & \text{if } f(x) > 1 \\
  \frac{1}{2} (1 - f(x))^2 & \text{if } f(x) \in [0, 1] \\
  \frac{1}{2} - f(x) & \text{if } f(x) < 0
  \end{cases}
  \]

• **Soft margin**
  \( \max(0, 1 - f(x)) \)

( asymptotically) linear

( asymptotically) 0
Risk minimization view

- Find function $f$ minimizing classification error
  \[ R[f] := \mathbb{E}_{x,y \sim p(x,y)} \{ yf(x) > 0 \} \]
- Compute empirical average
  \[ R_{\text{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} \{ y_if(x_i) > 0 \} \]
- Minimization is nonconvex
- Overfitting as we minimize empirical error
- Compute convex upper bound on the loss
- Add regularization for capacity control
  \[ R_{\text{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_if(x_i)) + \lambda \Omega[f] \]
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

\[ l(y, f(x)) = \frac{1}{2} (y - f(x))^2 \]
\( l(y, f(x)) = |y - f(x)| \)
\[ l(y, f(x)) = \max(0, |y - f(x)| - \epsilon) \]
Penalized least mean squares

• Optimization problem

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

• Solution

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

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

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

only inner product between X matters

matrix inverse use CG or SMW
SVM Regression
($\epsilon$-insensitive loss)

Figure 1.8

In SV regression, a tube with radius $\epsilon$ is fitted to the data. The trade-off between model complexity and points lying outside of the tube (positive slack variables $\xi$) 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 $f(x_i) - y_i > \epsilon$ and $y_i - f(x_i) > \epsilon$, respectively. We denote them by $\xi$ and $\xi^*$, respectively, and collectively refer to them as $\xi(\ast)$.

The optimization problem consists in finding

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

subject to

$$f(x_i) - y_i \leq \epsilon + \xi_i (1.49)$$

$$y_i - f(x_i) \leq \epsilon + \xi^*_i (1.50)$$

$$\xi_i, \xi^*_i \geq 0 (1.51)$$

for all $i = 1, \ldots, m$.

Note that according to (1.49) and (1.50), any error smaller than $\epsilon$ does not require a 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 $C > 0$ chosen a priori,

$$\max_{\alpha, \alpha^* \in \mathbb{R}^m} W(\alpha, \alpha^*) = -\epsilon \sum_{i=1}^m (\alpha^*_i + \alpha_i) + \sum_{i=1}^m (\alpha^*_i - \alpha_i) y_i - \frac{1}{2} m \sum_{i,j=1}^m (\alpha^*_i - \alpha_i)(\alpha^*_j - \alpha_j) k(x_i, x_j), (1.52)$$

subject to $0 \leq \alpha_i, \alpha^*_i \leq C, i = 1, \ldots, m$, and $m \sum_{i=1}^m (\alpha_i - \alpha^*_i) = 0 (1.53)$.

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

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

- Lagrange Function

\[
L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} [\xi_i + \xi_i^*] - \sum_{i=1}^{m} [\eta_i \xi_i + \eta_i^* \xi_i^*] + \\
\sum_{i=1}^{m} \alpha_i \left[ \langle w, x_i \rangle + b - y_i - \varepsilon - \xi_i \right] + \sum_{i=1}^{m} \alpha_i^* \left[ y_i - \varepsilon - \xi_i^* - \langle w, x_i \rangle - b \right]
\]
SVM Regression
(\(\epsilon\)-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**
  \[
  \text{minimize} \quad \frac{1}{2} (\alpha - \alpha^*)^\top K (\alpha - \alpha^*) + \epsilon 1^\top (\alpha + \alpha^*) + y^\top (\alpha - \alpha^*)
  \]
  subject to \(1^\top (\alpha - \alpha^*) = 0\) and \(\alpha_i, \alpha_i^* \in [0, C]\)
Properties

- Ignores ‘typical’ instances with small error
- Only upper or lower bound active at any time (we cannot violate both bounds simultaneously)
- Quadratic Program in 2n variables can be solved as cheaply as standard SVM problem
- Robustness with respect to outliers
  - $l_1$ 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 $\text{sinc} \, x$ 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 $\text{sinc} \, x$. The dotted line between them is the regression function.
Regression example

Figure 9.3 From top to bottom: approximation of the function \( \text{sinc} x \) 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 \( \text{sinc} x \). The dotted line between them is the regression function.
Figure 9.3

From top to bottom: approximation of the function $\text{sinc } x$ with precisions $\epsilon = 0.1, 0.2,$ and $0.5$. The solid top and dashed bottom lines indicate the size of the $\epsilon$-tube, here drawn around the target function $\text{sinc } x$. The dotted line between them is the regression function.
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 \( \varepsilon = 0.1, 0.2, 0.5 \). Note the decrease in the number of SVs.

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

\[
\min_{w \in H, \xi(\ast) \in \mathbb{R}^m, \varepsilon, b} \left( \frac{1}{2} \|w\|_2^2 + C \cdot (\nu \varepsilon + \frac{1}{m} \sum_{i=1}^{m} (\xi_i + \xi^\ast_i)) \right),
\]

subject to

\[
(\langle w, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i,
\]

\[
y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi^\ast_i,
\]

\[
\xi^\ast_i \geq 0, \quad \varepsilon \geq 0.
\]
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
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.)
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.
Typical Data
Outliers

2 4 0 5 4 2 8 5
7 9 0 5 3 8 4 8
0 4 2 9 4 7 0 7
5 2 8 8 2 2 6 3
5 8 4 0 4 8 4 3
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

#### Maximum a Posteriori

$$\min_{\theta} \sum_{i=1}^{m} g(\theta) - \langle \phi(x_i), \theta \rangle + \frac{1}{2\sigma^2} \|\theta\|^2$$

#### 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**

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

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

Lagrange Function \( L \)

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

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

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

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

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

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

- 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>

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 inside 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>

slightly more complicated. We consider the Lagrangian $L(w, \xi, b, \rho, \alpha, \beta, \delta) = \frac{1}{2} \|w\|^2 - \nu \rho + \frac{1}{m} \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} (\alpha_i (y_i (\langle x_i, w \rangle + b) - \rho + \xi_i) + \beta_i \xi_i)$.

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)
Structured Estimation (preview)
Large Margin Condition

- Binary classifier
  Correct class chosen with large margin $y f(x)$
- Multiple classes
- Score function per class $f(x, y)$
- Want that correct class has much larger score than incorrect class

$$f(x, y) - f(x, y') \geq 1 \text{ for all } y' \neq y$$

- Structured loss function (e.g. coal & diamonds)
  $$\Delta(y, y')$$
Large Margin Classifiers

- **Large Margin without rescaling** *(convex)*
  (Guestrin, Taskar, Koller)
  
  \[
  l(x, y, f) = \sup_{y' \in \mathcal{Y}} [f(x, y') - f(x, y) + \Delta(y, y')]
  \]

- **Large Margin with rescaling** *(convex)*
  (Tsochantaridis, Hofmann, Joachims, Altun)
  
  \[
  l(x, y, f) = \sup_{y' \in \mathcal{Y}} [f(x, y') - f(x, y) + 1] \Delta(y, y')
  \]

- **Both losses majorize misclassification loss**
  
  \[
  \Delta \left( y, \arg\max_{y'} f(x, y') \right)
  \]

- **Proof by plugging argmax into the definition**
Many applications

- Ranking (DCG, NDCG)
- Graph matching (linear assignment)
- ROC and $F_\beta$ scores
- Sequence annotation (named entities, activity)
- Segmentation
- Natural Language Translation
- Image annotation / scene understanding

- Caution - this loss is generally not consistent!
Extensions

• Invariances
  • Add prior knowledge (e.g. in OCR)
  • Make estimates robust against malicious abuse (e.g. spam filtering)

• Tighter upper bounds
  • Convex bound can be very loose
  • Overweights noisy data
  • Structured version of ramp loss

• Can be shown to be consistent
More Kernel Algorithms
Kernel PCA
Principal Component Analysis

- Gaussian density model

\[ p(x; \mu, \Sigma) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp \left( -\frac{1}{2} (x - \mu) \Sigma^{-1} (x - \mu) \right) \]

- Estimate variance by empirical average

\[ \hat{\Sigma} = \frac{1}{m} \sum_{i=1}^{m} x_i x_i^\top - \hat{\mu} \hat{\mu}^\top \text{ where } \hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} x_i \]

- Good approximation by low-rank model
  - Extract leading eigenvalues of covariance
  - Data might lie in a subspace
Principal Component Analysis

- Generative approximation of data
  \[ x = \sum_i \sigma_i v_i \alpha_i \text{ where } \alpha_i \sim \mathcal{N}(0, 1) \]

- Heuristic
  Good explanation of data implies that we have meaningful dimensions of the data.

- Linear feature extraction
  \[ g_i(x) = \langle v_i, x \rangle \]

- PCA is reconstruction with smallest $l_2$ error
good for exploratory data analysis
Kernel PCA

**linear PCA**

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

**kernel PCA**

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

Figure 14.1

The basic idea of Kernel PCA. In some high-dimensional feature space \( H \) (bottom right), we perform linear PCA, as with classical PCA in input space (top). Since \( H \) is nonlinearly related to input space (via \( \Phi \)), the contour lines of constant projections onto the principal eigenvector (drawn as an arrow) are nonlinear in input space. We cannot draw a pre-image of the eigenvector in input space, as it may not even exist. Crucial to Kernel PCA is the fact that there is no need to perform the map into \( H \): all necessary computations are carried out using a kernel function \( k \) in input space (here: \( \mathbb{R}^2 \)).

Figure 14.2

Feature extractor constructed using Kernel PCA (cf. (14.16)). In the first layer, the input vector is compared to the sample via a kernel function, chosen a priori (e.g. polynomial, Gaussian, or sigmoid). The outputs are then linearly combined using weights, which are found by solving an eigenvector problem. As shown in the text, the function of the network can be thought of as the projection onto an eigenvector of a covariance matrix in a high-dimensional feature space. As a function on input space, it is nonlinear.

The principal components are uncorrelated. The first \( q \) principal components have maximal mutual information (see [133, 126]) with respect to the inputs (this holds under Gaussianity assumptions in \( H \), and thus strongly depends on the particular kernel chosen and on the data).

Proof

All these statements are completely analogous to the case of standard PCA. As an example, we prove the second property, in the simplification where the data \( x_1, \ldots, x_m \) in feature space are centered. We consider an orthogonal basis transformation \( W \), and notation \( P_q \) for the projector on the first \( q \) canonical basis vectors \( \{e_1, \ldots, e_q\} \). Then the mean squared reconstruction error using \( q \)
PCA via inner products

• Eigenvector condition

\[ \Sigma v = \lambda v \]

\[ \frac{1}{m} \sum_i \bar{x}_i \bar{x}_i^\top v = \lambda v \text{ for } \bar{x}_i = x_i - \frac{1}{m} \sum_i x_i \]

hence \( v = \sum_j \alpha_j \bar{x}_j \)

using \( \bar{x}_l^\top \frac{1}{m} \sum_i \bar{x}_i \bar{x}_i^\top v = \lambda \bar{x}_l^\top v \)

yields \( \frac{1}{m} \bar{K} \bar{K} \alpha = \lambda \bar{K} \alpha \)

• Kernel PCA

\[ \frac{1}{m} \bar{K} \alpha = \lambda \alpha \text{ where } \bar{K}_{ij} = \langle \bar{x}_i, \bar{x}_j \rangle \]
14.3 Kernel PCA Experiments

In this section, we present a set of experiments in which Kernel PCA is used (in the form taking into account centering in $H$) to extract principal components. First, we take a look at a simple toy example; following this, we describe real-world experiments where we assess the utility of the extracted principal components in classification tasks.

Figure 14.3: Two-dimensional toy example, with data generated as follows: $x$-values have uniform distribution in $[-1, 1]$, $y$-values are generated from $y_i = x_i^2 + \nu$, where $\nu$ is normal noise with standard deviation 0.2. From left to right, the polynomial degree in the kernel (14.21) increases from 1 to 4; from top to bottom, the first 3 eigenvectors are shown, in order of decreasing eigenvalue size (eigenvalues are normalized to sum to 1). The figures contain lines of constant principal component value (contour lines); in the linear case ($d=1$), these are orthogonal to the eigenvectors. We do not depict the eigenvectors, as in the general case, they belong in a higher-dimensional feature space. Note, finally, that for $d=1$, there are only 2 non zero eigenvectors, this number being equal to the dimension of the input space.

To provide insight into how PCA in $H$ behaves in input space, we describe a set of experiments with an artificial 2-D data set, using polynomial kernels, $k(x, x') = \langle x, x' \rangle^d$, (14.21)
Feature extraction

Eigenvalue=0.251

Eigenvalue=0.233

Eigenvalue=0.052

Eigenvalue=0.044

Eigenvalue=0.037

Eigenvalue=0.033

Eigenvalue=0.031

Eigenvalue=0.025

Eigenvalue=0.014

Eigenvalue=0.008

Eigenvalue=0.007

Eigenvalue=0.006

Eigenvalue=0.005

Eigenvalue=0.004

Eigenvalue=0.003

Eigenvalue=0.002

Figure 14.4

Two-dimensional toy example with three data clusters (Gaussian with standard deviation 0.1; depicted region, $[-1, 1] \times [-0.5, 1]$): first 16 nonlinear principal components extracted with $k(x, x') = \exp(-\|x - x'\|^2 / 0.1)$. Note that the first 2 principal components (top left), which possess the largest eigenvalues, nicely separate the three clusters. The components 3 – 5 split the clusters into halves. Similarly, components 6–8 split them again, in a manner orthogonal to the above splits. The higher components are more difficult to describe. They look for finer structure in the data set, identifying higher-order moments.
Mean Classifier
‘Trivial’ classifier

- Represent each class by mean in feature space
- Classify along direction of maximum discrepancy between classes
- Trivial to ‘train’
• Class mean

\[ \mu_+ = \frac{1}{m_+} \sum_{i:y_i=1} \phi(x_i) \text{ and } \mu_- = \frac{1}{m_-} \sum_{i:y_i=-1} \phi(x_i) \]

• Classifier

\[ f(x) = \langle \mu_+ - \mu_-, \phi(x) \rangle = \sum_i \frac{y_i}{m y_i} k(x_i, x) \]
More kernel methods

- Canonical Correlation analysis
- Two sample test
  - Mean in feature space is sufficient to fully represent a distribution
  - Compare them by computing distance
- Independence test
  - Compare joint and product of marginals
- Structured feature extraction
  - Find directions of high significance and low function complexity
Conditional Models
Gaussian Processes
Weight & height
Weight & height

assume Gaussian correlation
\[ p(\text{weight} | \text{height}) = \frac{p(\text{height}, \text{weight})}{p(\text{height})} \propto p(\text{height}, \text{weight}) \]
\[ p(x_2 | x_1) \propto \exp \left( -\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^\top \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix}^{-1} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right) \]

keep linear and quadratic terms of exponent
Correlated Observations
Assume that the random variables $t \in \mathbb{R}^n, t' \in \mathbb{R}^{n'}$ are jointly normal with mean $(\mu, \mu')$ and covariance matrix $K$

$$p(t, t') \propto \exp \left( -\frac{1}{2} \begin{bmatrix} t - \mu \\ t' - \mu' \end{bmatrix}^\top \begin{bmatrix} K_{tt} & K_{tt'} \\ K_{tt'}^{\top} & K_{tt'} \end{bmatrix}^{-1} \begin{bmatrix} t - \mu \\ t' - \mu' \end{bmatrix} \right).$$

Inference
Given $t$, estimate $t'$ via $p(t'|t)$. Translation into machine learning language: we learn $t'$ from $t$.

Practical Solution
Since $t'|t \sim \mathcal{N}(\mu', \tilde{K})$, we only need to collect all terms in $p(t, t')$ depending on $t'$ by matrix inversion, hence

$$\tilde{K} = K_{tt'} - K_{tt'}^{\top} K_{tt}^{-1} K_{tt'}$$
and
$$\tilde{\mu} = \mu' + K_{tt'}^{\top} \left[ K_{tt}^{-1} (t - \mu) \right]$$
independent of $t'$. 
Gaussian Process

Key Idea
Instead of a fixed set of random variables \( t, t' \) we assume a stochastic process \( t : \mathcal{X} \rightarrow \mathbb{R} \), e.g. \( \mathcal{X} = \mathbb{R}^n \).
Previously we had \( \mathcal{X} = \{ \text{age, height, weight, \ldots} \} \).

Definition of a Gaussian Process
A stochastic process \( t : \mathcal{X} \rightarrow \mathbb{R} \), where all \( (t(x_1), \ldots, t(x_m)) \) are normally distributed.

Parameters of a GP
\[
\begin{align*}
\text{Mean} & \quad \mu(x) := \mathbb{E}[t(x)] \\
\text{Covariance Function} & \quad k(x, x') := \text{Cov}(t(x), t(x'))
\end{align*}
\]

Simplifying Assumption
We assume knowledge of \( k(x, x') \) and set \( \mu = 0 \).
Kernels ...

Covariance Function
- Function of two arguments
- Leads to matrix with nonnegative eigenvalues
- Describes correlation between pairs of observations

Kernel
- Function of two arguments
- Leads to matrix with nonnegative eigenvalues
- Similarity measure between pairs of observations

Lucky Guess
- We suspect that kernels and covariance functions are the same . . .
The connection

Gaussian Process on Parameters

\[ t \sim \mathcal{N}(\mu, K) \text{ where } K_{ij} = k(x_i, x_j) \]

Linear Model in Feature Space

\[ t(x) = \langle \Phi(x), w \rangle + \mu(x) \text{ where } w \sim \mathcal{N}(0, 1) \]

The covariance between \( t(x) \) and \( t(x') \) is then given by

\[ E_w [\langle \Phi(x), w \rangle \langle w, \Phi(x') \rangle] = \langle \Phi(x), \Phi(x') \rangle = k(x, x') \]

Conclusion

A small weight vector in “feature space”, as commonly used in SVM amounts to observing \( t \) with high \( p(t) \).

\[ \text{Log prior } \quad - \log p(t) \quad \longleftrightarrow \quad \text{Margin } \|w\|^2 \]

Will get back to this later again.
Regression
Joint Gaussian Model

- Random variables \((t, t')\) are drawn from GP
- Observe a subset \(t\) of them
- Predict the rest using
  \[
  \tilde{K} = K_{t't'} - K_{tt'} K_{tt}^{-1} K_{tt'} \quad \text{and} \quad \tilde{\mu} = \mu' + K_{tt'} [K_{tt}^{-1}(t - \mu)]
  \]
- Linear expansion (precompute things)
- Predictive uncertainty is data independent
  Good for experimental design
- Predictive uncertainty is data independent
- Predictive variance vanishes if \(K\) is rank deficient
**Observation**

Any function $k$ leading to a symmetric matrix with non-negative eigenvalues is a valid covariance function.

**Necessary and sufficient condition (Mercer’s Theorem)**

$k$ needs to be a nonnegative integral kernel.

**Examples of kernels** $k(x, x')$

- **Linear**
  \[ \langle x, x' \rangle \]

- **Laplacian RBF**
  \[ \exp \left( -\lambda \| x - x' \| \right) \]

- **Gaussian RBF**
  \[ \exp \left( -\lambda \| x - x' \|^2 \right) \]

- **Polynomial**
  \[ (\langle x, x' \rangle + c)^d, \quad c \geq 0, \quad d \in \mathbb{N} \]

- **B-Spline**
  \[ B_{2n+1}(x - x') \]

- **Cond. Expectation**
  \[ \mathbf{E}_c[p(x | c)p(x' | c)] \]
Linear ‘GP regression’

**Linear kernel:** $k(x, x') = \langle x, x' \rangle$

- Kernel matrix $X^\top X$
- Mean and covariance

\[
\tilde{K} = X'\!\!^\top X' - X'\!\!^\top X (X^\top X)^{-1} X^\top X' = X'\!\!^\top (1 - P_X) X'.
\]
\[
\tilde{µ} = X'\!\!^\top \left[X (X^\top X)^{-1} t \right]
\]

- $\tilde{µ}$ is a **linear function of** $X'$.

**Problem**

- The covariance matrix $X^\top X$ has at most rank $n$.
- After $n$ observations ($x \in \mathbb{R}^n$) the variance vanishes. This is **not realistic**.
- “Flat pancake” or “cigar” distribution.
Degenerate Covariance
Additive Noise

Indirect Model

Instead of observing $t(x)$ we observe $y = t(x) + \xi$, where $\xi$ is a nuisance term. This yields

$$p(Y|X) = \int \prod_{i=1}^{m} p(y_i|t_i)p(t|X)dt$$

where we can now find a maximum a posteriori solution for $t$ by maximizing the integrand (we will use this later).

Additive Normal Noise

- If $\xi \sim \mathcal{N}(0, \sigma^2)$ then $y$ is the sum of two Gaussian random variables.
- Means and variances add up.

$$y \sim \mathcal{N}(\mu, K + \sigma^21).$$
Predictive mean $k(x, X) \top (K(X, X) + \sigma^2 1)^{-1} y$
Variance

\[ \text{Variance} \]
Putting it all together
Putting it all together
Covariance Matrices

- Additive noise

\[ K = K_{\text{kernel}} + \sigma^2 \mathbf{1} \]

- Predictive mean and variance

\[ \tilde{K} = K_{tt} - K_{tt'} K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = K_{tt'} K_{tt}^{-1} t \]

Pointwise prediction

\[ K_{tt} = K + \sigma^2 \mathbf{1} \]
\[ K_{tt'} = k(x, x) + \sigma^2 \]
\[ K_{tt'} = (k(x_1, x), \ldots, k(x_m, x)) \]

Plug this into the mean and covariance equations.
Gaussian Process Conditional Models
Exponential Families
• Density function

\[ p(x; \theta) = \exp(\langle \phi(x), \theta \rangle - g(\theta)) \]

where \( g(\theta) = \log \sum_{x'} \exp(\langle \phi(x'), \theta \rangle) \)
Exponential Families

• Density function

\[ p(x; \theta) = \exp \left( \langle \phi(x), \theta \rangle - g(\theta) \right) \]

where \( g(\theta) = \log \sum_{x'} \exp \left( \langle \phi(x'), \theta \rangle \right) \)

• Log partition function generates cumulants

\[ \partial_{\theta} g(\theta) = \mathbb{E} [\phi(x)] \]
\[ \partial_{\theta}^2 g(\theta) = \text{Var} [\phi(x)] \]
Exponential Families

• Density function

\[ p(x; \theta) = \exp \left( \langle \phi(x), \theta \rangle - g(\theta) \right) \]

where \( g(\theta) = \log \sum_{x'} \exp \left( \langle \phi(x'), \theta \rangle \right) \)

• Log partition function generates cumulants

\[ \partial_\theta g(\theta) = \mathbf{E} \left[ \phi(x) \right] \]
\[ \partial_\theta^2 g(\theta) = \text{Var} \left[ \phi(x) \right] \]

• \( g \) is convex (second derivative is p.s.d.)
Conditional Exponential Families

\[ p(y|x; \theta) = \exp (\langle \phi(x, y), \theta \rangle - g(\theta|x)) \]

where \( g(\theta|x) = \log \sum_{y'} \exp (\langle \phi(x, y'), \theta \rangle) \)

\[ \partial_\theta g(\theta|x) = \mathbf{E} [\phi(x, y)|x] \]

\[ \partial^2_\theta g(\theta|x) = \text{Var} [\phi(x, y)|x] \]
• **Density function**

\[
p(y|x; \theta) = \exp \left( \langle \phi(x, y), \theta \rangle - g(\theta|x) \right)
\]

where \( g(\theta|x) = \log \sum_{y'} \exp \left( \langle \phi(x, y'), \theta \rangle \right) \)

\[
\partial_\theta g(\theta|x) = \mathbf{E} [\phi(x, y)|x]
\]

\[
\partial^2_\theta g(\theta|x) = \text{Var} [\phi(x, y)|x]
\]
Conditional Exponential Families

• Density function

\[ p(y|x; \theta) = \exp \left( \langle \phi(x, y), \theta \rangle - g(\theta|x) \right) \]

where \( g(\theta|x) = \log \sum_{y'} \exp \left( \langle \phi(x, y'), \theta \rangle \right) \)

• Log partition function generates cumulants

\[ \partial_\theta g(\theta|x) = \mathbb{E} [\phi(x, y)|x] \]

\[ \partial_\theta^2 g(\theta|x) = \text{Var} [\phi(x, y)|x] \]
Conditional Exponential Families

- **Density function**

  \[ p(y|x; \theta) = \exp \left( \langle \phi(x, y), \theta \rangle - g(\theta|x) \right) \]

  where \( g(\theta|x) = \log \sum_{y'} \exp \left( \langle \phi(x, y'), \theta \rangle \right) \)

- **Log partition function generates cumulants**

  \[ \partial_\theta g(\theta|x) = \mathbb{E} [\phi(x, y)|x] \]

  \[ \partial^2_\theta g(\theta|x) = \text{Var} [\phi(x, y)|x] \]

- **g is convex (second derivative is p.s.d.)**
Key Idea

- Gaussian Process indexed by \((x, y)\)
- Binary \(y\) yields classification
- Set for \(y\) yields multiclass
- Integer \(y\) yields Poisson regression
- Scalar \(y\) yields heteroscedastic regression
- Sequence for \(y\) yields CRF
- ... and lots more ...

- The GP is in the latent variables (Regression is special case where we can integrate)
Conditional GP Model

- **Data likelihood**
  \[ p(y|x, t(x)) := e^{t(x,y) - g(t(x))} \]
  \[ \text{where } g(t(x)) = \sum_y e^{t(x,y)} \]

- **Prior**
  \[ t \sim \mathcal{N}(\mu, K) \]

- **Posterior distribution**
  \[ p(t|X, Y) \propto \exp \left( \sum_i t(x_i, y_i) - g(t(x_i)) - \frac{1}{2} t^\top K^{-1} t \right) \]

- **Maximize with respect to t for MAP estimate**
Logistic Regression
Binomial Model

• Binary label space \{-1, 1\}
• We can center \(t(x,y)\) as \(y \cdot t(x)\) (constant offset doesn’t change model)
• Log-likelihood

\[
- \log p(y|t) = \log [e^t + e^{-t}] - yt = \log [1 + e^{-2yt}]
\]

• After rescaling by 2 this is the logistic loss
• MAP estimation problem

\[
\min_t \frac{1}{2} t^\top K^{-1} t + \sum_{i=1}^{m} \log [1 + e^{-y_i t_i}]
\]
More loss functions

- **Logistic** \( \log \left[ 1 + e^{-f(x)} \right] \)

- **Huberized loss**
  \[
  \begin{cases}
  0 & \text{if } f(x) > 1 \\
  \frac{1}{2}(1 - f(x))^2 & \text{if } f(x) \in [0, 1] \\
  \frac{1}{2} - f(x) & \text{if } f(x) < 0 
  \end{cases}
  \]

- **Soft margin**
  \( \max(0, 1 - f(x)) \)
Clean Data
Noisy Data
Heteroscedastic Estimation
Motivation

• GP Regression has variance estimate independent of observed data
• Assumes that we know variance globally beforehand
• **This is nonsense!**
• Estimate mean and variance jointly
• Easily possible in an exponential family model

Le, Canu, Smola, 2005
Recall - Normal distributions

Engineer’s favorite

\[ p(x) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{1}{2\sigma^2} (x - \mu)^2 \right) \quad \text{where } x \in \mathbb{R} =: \mathcal{X} \]

Massaging the math

\[ p(x) = \exp \left( \langle (x, -0.5x^2), \theta \rangle - \left( \frac{\mu^2}{2\sigma^2} + \frac{1}{2} \log(2\pi \sigma^2) \right) \right) \]

Using the substitution \( \theta_2 := \sigma^{-2} \) and \( \theta_1 := \mu \sigma^{-2} \) yields

\[ g(\theta) = \frac{1}{2} \left[ \theta_1^2 \theta_2^{-1} + \log 2\pi - \log \theta_2 \right] \]
Basic Idea

Sufficient Statistic

We pick \( \phi(x, y) = (y\phi_1(x), y^2\phi_2(x)) \), that is

\[
k((x, y), (x', y')) = k_1(x, x')yy' + k_2(x, x')y^2y'^2 \quad \text{where } y, y' \in \mathbb{R}
\]

Hence estimate mean and variance simultaneously.

Optimization Problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} \left[ -\frac{1}{4} \left[ \sum_{j=1}^{m} \alpha_{1j}k_1(x_i, x_j) \right] \right] - \frac{1}{2} \sum_{j=1}^{m} \alpha_{2j}k_2(x_i, x_j) + \sum_{j=1}^{m} \left[ y_j^\top \alpha_{1j}k_1(x_i, x_j) + (y_j^\top \alpha_{2j}y_j)k_2(x_i, x_j) \right] + \frac{1}{2\sigma^2} \sum_{i,j} \alpha_{1i}^\top \alpha_{1j}k_1(x_i, x_j) + \text{tr} \left[ \alpha_{2i}^\top \alpha_{2j}^\top k_2(x_i, x_j) \right]
\end{align*}
\]

subject to \( 0 > \sum_{i=1}^{m} \alpha_{2i}k(x_i, x_j) \)

- The problem is convex
- The log-determinant from the normalization of the Gaussian acts as a \textbf{barrier function}, i.e. a nice SDP.
## Computational Issues

### Newton Method with CG Solver
Use Newton method to compute update direction, CG solver instead of inverting Hessian.

### Lazy Evaluation
Never build explicit Hessian.

### Reduced Rank
Use incomplete Cholesky factorization for low-rank approximation.

### Result

<table>
<thead>
<tr>
<th>$m$</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1k</th>
<th>2k</th>
<th>5k</th>
<th>10k</th>
<th>20k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Hessian</td>
<td>8</td>
<td>18</td>
<td>90</td>
<td>607</td>
<td>3551</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hessian vector</td>
<td>9</td>
<td>15</td>
<td>38</td>
<td>115</td>
<td>752</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reduced rank</td>
<td>7</td>
<td>7</td>
<td>12</td>
<td>30</td>
<td>54</td>
<td>179</td>
<td>368</td>
<td>727</td>
</tr>
</tbody>
</table>

This yields scaling of $O(m^{2.1})$, $O(m^{1.4})$, and $O(m^{0.95})$. 
Standard GP
Heteroscedastic GP mean
Heteroscedastic GP variance

![3D plot of variance against longitude and latitude](image)

- Variance is plotted against longitude (in m) and latitude (in m).
- The x-axis represents longitude ranging from -2 to 3 × 10^5.
- The y-axis represents latitude ranging from -1 to 3 × 10^5.
- The z-axis represents variance, ranging from 0 to 5 × 10^4.

This plot illustrates the heteroscedasticity in the GP model, showing how variance changes across different locations.
(Generalized) Linear Models

- Kernel trick
  - Simple kernels
  - Kernel PCA
  - Mean Classifier
- Support Vectors
  - Support Vector Machine classification
  - Regression
  - Logistic regression
  - Novelty detection
- Gaussian Process Estimation
  - Regression
  - Classification
  - Heteroscedastic Regression
Further reading

• Ramp loss consistency

• Ranking and structured estimation
  http://users.cecs.anu.edu.au/~chteo/pub/LeSmoChaTeo09.pdf

• Invariances and convexity
  http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11755

• Ramp loss for structured estimation

• Structured estimation (with margin rescaling)
  http://ttic.uchicago.edu/~altun/pubs/AltHofTso06.pdf

• Structured estimation (without margin rescaling)
  http://www.seas.upenn.edu/~taskar/pubs/icml05.pdf

• Ben Taskar’s tutorial
  http://www.seas.upenn.edu/~taskar/nips07tut/nips07tut.ppt
Further reading

• SVM Tutorial (regression)

• SVM Tutorial (classification)

•Introductory chapter of Kernel book

• Introductory chapter of structured estimation book
  http://alex.smola.org/teaching/berkeley2012/slides/se_chapter2.pdf

• Kernel PCA
  http://dl.acm.org/citation.cfm?id=295919.295960