Outline

Risk Minimization and Learning Problems:
   Classification, Regression, Novelty Detection

Feature Spaces, Preprocessing, and Kernels:
   Representer Theorem, Examples of Kernels, Testing Kernels

Sequential Minimal Optimization:
   Subset Minimization Strategy, Choice Heuristics, Stopping Rules

Online Learning:
   Stochastic Gradient Descent, Examples

Applications: Optical Character Recognition, Jet Engines, Chairs, Bioinformatics,
   Network Intrusion Detection, Spam Filtering, Text Categorization
Classification

**Data:** Pairs of observations \((x_i, y_i)\) generated from some distribution \(P(x, y)\), e.g., (blood status, cancer), (credit transaction information, fraud), (sound profile of jet engine, defect)

**Task:** Predict \(y\) given \(x\) at a new location.

Modification: find a function \(f(x)\) that does the task.
Minimize $\frac{1}{2}\|w\|^2$ subject to $y_i(\langle w, x_i \rangle + b) \geq 1$ for all $i$. 

Note: 

\[
\begin{align*}
\langle w, x_1 \rangle + b &= +1 \\
\langle w, x_2 \rangle + b &= -1 \\
\Rightarrow \langle w, (x_1 - x_2) \rangle &= 2 \\
\Rightarrow \frac{\langle w, (x_1 - x_2) \rangle}{\|w\|} &= \frac{2}{\|w\|}
\end{align*}
\]
Regression

Data:
Pairs of observations $(x_i, y_i)$ generated from some $P(x, y)$, e.g.,
(market index, SP100)
(fab parameters, yield)
(user profile, price)

Task:
Predict $y$, given $x$. 
\( \varepsilon \)-insensitive Linear Regressor

Optimization Problem:

A) \( \frac{1}{2} \| w \|^2 \) subject to \(-\varepsilon \leq y_i - (\langle w, x_i \rangle + b) \leq \varepsilon\)

B) \( \frac{1}{2} \| w \|^2 + \sum_{i=1}^{m} \xi_i + \xi_i^* \) subject to \(- (\varepsilon + \xi_i) \leq y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i^*\)
Novelty Detection

**Data:**
Observations \((x_i, y_i)\) 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.
Maximum Distance Hyperplane

**Idea:** Find hyperplane 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:**

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

**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:** We construct a **Lagrange Function** \( L \) by subtracting the constraints, multiplied by **Lagrange multipliers** \((\alpha_i \text{ and } \eta_i)\), from the **Primal Objective Function**. \( L \) has a **saddlepoint** at the optimal solution.

\[
L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (\langle w, x_i \rangle - 1 + \xi_i) - \sum_{i=1}^{m} \eta_i \xi_i \text{ where } \alpha_i, \eta_i \geq 0
\]

For instance, if \( \xi_i < 0 \) we could increase \( L \) without bound via \( \eta_i \).
The Dual Optimization Problem II

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 we substitute the two optimality conditions back into \( L \).

Dual Problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i \\
\text{subject to} & \quad \alpha_i \in [0, C]
\end{align*}
\]

All this is only possible due to the convexity of the primal problem.
The $\nu$-Trick

**Problem:** Depending on how we choose $C$, the number of points selected as lying on the “wrong” side of the hyperplane $H := \{x|\langle w, x \rangle = 1\}$ will vary. But we would like to specify a certain fraction $\nu$ beforehand.

**Solution:** Use adaptive hyperplane that separates data from the origin, i.e. find $H := \{x|\langle w, x \rangle = \rho\}$ where the threshold $\rho$ is adaptive.

**Primal Problem:** minimize $\frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$

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

**Dual Problem:** minimize $\frac{1}{2} \sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle$

subject to $\alpha_i \in [0, 1]$ and $\sum_{i=1}^{m} \alpha_i = \nu m$. 
Optimization Problems with the $\nu$-Trick

**Classification:**

minimize \[ \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

subject to \( \alpha_i \in [0, 1], \sum_{i=1}^{m} \alpha_i = \nu m \) and \( \sum_{i=1}^{m} \alpha_i y_i = 0 \).

function expansion \( f(x) = \sum_{i=1}^{m} \alpha_i y_i \langle x_i, x \rangle + b \)

**Regression:**

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

subject to \( \alpha_i, \alpha_i^* \in [0, C] \) and \( \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) = C \nu m \).

function expansion \( f(x) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b \)
Kernels and Nonlinearity

**Problem:** linear functions are often too simple to provide good estimators.

**Idea 1:** map to a higher dimensional feature space via $\Phi : x \rightarrow \Phi(x)$ and solve the problem there.

Replace every $\langle x, x' \rangle$ by $\langle \Phi(x), \Phi(x') \rangle$.

**Idea 2:** instead of computing $\Phi(x)$ explicitly use **kernel function**

$$k(x, x') := \langle \Phi(x), \Phi(x') \rangle.$$ 

**Strategy:** Replace every $\langle x, x' \rangle$ by $k(x, x')$. 
Example: Polynomial Kernels

**Quadratic Features in** $\mathbb{R}^2$: $\Phi(x) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

**Dot Product:**

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left( x_1^2, \sqrt{2}x_1x_2, x_2^2 \right) , \left( x_1'^2, \sqrt{2}x_1'x_2', x_2'^2 \right) \right\rangle = \langle x, x' \rangle^2.$$  

**Extension:** This dot product trick does not only work for 2nd order polynomials but for any order: $k(x, x') = \langle x, x' \rangle^d$. 

Alex Smola, Australian National University: Tutorial given at ISCAS 2001: Support Vector Learning — Concepts and Algorithms
Kernels and Mercer’s Theorem

**Question:** which functions $k(x, x')$ can we use as kernels?

**Short Answer:** all functions $k(x, x')$ generating symmetric matrices $K \in \mathbb{R}^{m \times m}$ with $K_{ij} := k(x_i, x_j)$ where all eigenvalues are nonnegative.

**Long Answer:** all $k(x, x')$ satisfying the conditions of Mercer’s Theorem:

Any $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that satisfies

$$\int k(x, x') f(x) f(x') dx dx' \geq 0$$

for all $f \in L_2(\mathcal{X})$ can be written as

$$k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x')$$

where $\phi_i(x)$ are the eigenfunctions of the integral operator.
Theorem: Any translation invariant symmetric \( k(x, x') = k(x - x') \) satisfies Mercer’s condition if it has a nonnegative Fourier transform.

The decay properties of the spectrum tell us the smoothness of \( k \).

Gaussian RBF Kernels: \( k(x, x') = \exp \left( -\frac{\|x - x'\|^2}{2\sigma^2} \right) \)

The Fourier transform is also a Gaussian, however with inverse width.

Laplacian RBF Kernels: \( k(x, x') = \exp \left( -\frac{\|x - x'\|}{\sigma} \right) \)

The Fourier spectrum decays less rapidly (damped harmonic oscillator in one dimension).

Indicator Function: \( k(x, x') = 1_{[-1,1]}(x - x') \) is not a Mercer kernel since its Fourier transform, the sinc function, has negative entries.
Translation Invariant Kernels

Gaussian Kernel

Laplacian Kernel

Indicator Kernel
Dot Product Kernels

**Theorem:** Any **dot product** kernel $k(x, x') = k(\langle x, x' \rangle)$ satisfies Mercer’s condition if it has a **nonnegative Taylor series expansion**.

The **decay properties of the Taylor expansion** tell us the **smoothness** of $k$.

**Polynomial Kernels:**

$$k(x, x') = (\langle x, x' \rangle + c)^d$$

This is a proper kernel only for $d \in \mathbb{N}$ and $c \geq 0$.

**tanh-Kernels:**

$$k(x, x') = \tanh (a \langle x, x' \rangle + b)$$ is never a proper kernel, for any $a, b \in \mathbb{R}$.

**Infinite Polynomials:**

$$k(x, x') = \frac{1}{1 - \langle x, x' \rangle}$$ for $\|x\|, \|x'\| \leq 1$

is a Mercer kernel with bad regularization properties (flat Taylor series).
Question: is there always a kernel expansion

\[ f(x) = \sum_i \alpha_i k(x_i, x) \]

if we minimize

\[ R_{\text{reg}}[w] = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} c(x_i, y_i, f(x_i)) \]

Answer (Representer Theorem): YES

Kimeldorf and Wahba, 1971; Cox and O’Sulivan, 1990; This holds for SVM, Gaussian Processes, Regularization Networks, . . .

Consequences: the number of kernels increases with the number of observations \( m \). Finding the optimal solutions will typically cost \( O(m^2) \) to \( O(m^3) \) operations (depending on the data).
### Optimization Techniques

**Interior Point Methods:** solve constraints for primal and dual optimization problem simultaneously and satisfy the Kuhn-Tucker conditions. This involves inverting $K$, i.e. $O(m^3)$.

**Sherman-Morrison-Woodbury Methods:** exploit the fact that $K$ does not really have full rank and invert a rank-$n$ approximation of $K$. This leads to $O(m \cdot n^2)$ algorithms.

**Chunking and Sequential Minimal Optimization:** take subproblems of the big quadratic minimization problem and optimize over a subset variables at a time. Smaller memory footprint and sometimes fast.

**Online Methods:** stochastic gradient descent on the objective function. Simple, approximate algorithm, small memory footprint.
Flashback: Singleclass Dual QP

Dual Quadratic Program for $\nu$-SVM:

\[
\text{minimize } \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j K_{ij}
\]

subject to $\alpha_i \in [0, 1]$ and $\sum_{i=1}^{m} \alpha_i = \nu m$.

Splitting: divide the $m$ patterns into a working set $S^w$ and a fixed set $S_f$. Minimizing the objective function over $S^w$ can only decrease it.

Subproblem on $S_w$:

\[
\text{minimize } \frac{1}{2} \sum_{i,j \in S_w} \alpha_i^w \alpha_j^w K_{ij}^w + \sum_{i \in S^w, j \in S^f} \alpha_i^w \alpha_j^f K_{ij}^{wf}
\]

subject to $\alpha_i^w \in [0, 1]$ and $\sum_{i \in S^w} \alpha_i^w = \nu m - \sum_{i \in S^f} \alpha_i^f$.
Subset Selection and Chunking

Simple Version: Start with small set, train, keep SVs, add in new patterns, repeat. This works great for almost noise free data, e.g. USPS and NIST OCR problems (Vapnik & Chervonenkis, 1965, AT&T OCR Group, 1990es).

Basic Idea: Take subset of variables, say $\alpha_i \ldots \alpha_I$, optimize over the latter while keeping the rest fixed. Then pick next set of variables, optimize, ... (Osuna, Freund, Girosi, 1997). Probably best implementation by Joachims (SVMLight). Works great on texts.

Common Problems: Chunking does not always converge in practice and speed of convergence is highly problem dependent. Which variables should you pick? Performance degrades with the number of additional linear constraints. Most problems occur if many variables are neither 0 nor $C$. 
Sequential Minimal Optimization

Basic Idea: Optimize only over pairs of variables — we need pairs to keep the equality constraints satisfied (Platt, 1998).

Advantage: Analytic solution of subproblems is possible (at least for linear and quadratic loss functions), simple one-dimensional convex minimization otherwise.

Scaling Behaviour: Very large problems can be solved at only $O(m)$ storage cost, provided we are willing to wait long enough (time scales with $O(m^\gamma)$ where $\gamma > 2$, depending on the choice of problems and modifications).

Problems: Some formulations are hard to deal with in SMO: many nonzero start variables, several constraints at the same time (as in $\nu$-SVM), special selection strategy for most modifications.
Explicit Solution for 2 Variables

Dual Quadratic Program in 2 Variables:

minimize \( \frac{1}{2} \sum_{i,j=1}^{2} \alpha_i \alpha_j K_{ij} + \sum_{i=1}^{2} \alpha_i v_i \)

subject to \( \alpha_i \in [0, 1] \) and \( \sum_{i=1}^{2} \alpha_i = \Delta \)

Here \( \Delta = \nu m - \sum_{i=3}^{m} \alpha_i \) and \( C_i = \sum_{j=3}^{m} \alpha_j K_{ij} \).

Reduce Problem to 1 Variable: Use \( \alpha_1 = \Delta - \alpha_2 \) and substitute into the restricted optimization problem. This is a quadratic function on an interval, hence we can find the solution \( \alpha_2^{\text{new}} = \frac{\Delta(K_{11} - K_{12}) - C_1 + C_2}{K_{11} + K_{22} - 2K_{12}} \) and have to restrict it such that \( \alpha_2^{\text{new}} \) and \( \Delta - \alpha_2^{\text{new}} \) are in \([0, 1]\).
Selection Strategy

**Problem:** which variables \( i, j \) should we select?

**Idea 1:** we only make progress if \( \alpha_i \) is not already satisfying the optimality conditions,

i.e. \( f(x_i) > \rho \) for \( \alpha_i = 0 \) or \( f(x_i) < \rho \) for \( \alpha_i = 1 \).

**Idea 2:** we can show that the change by optimizing over \( i, j \) depends on \( \frac{C_i - C_j}{K_{11} + K_{22} - 2K_{12}} \).

**Strategy, Part 1:** find \( i \) where \( C_i \) is large and where simultaneously \( \alpha_i \) does not satisfy the constraints.

**Strategy, Part 2:** find \( j \) where \( C_i - C_j \) is large and where also \( \alpha_j \) does not satisfy the constraints.
Stopping Rule

**Problem:** when have we optimized enough?

**Idea 1:** use a lower bound for the objective function and stop if the relative gap size is small enough.

**Idea 2:** recall that the KKT-conditions tell us the gap size (i.e., the terms we subtracted in the Lagrange function).

**Gap for Novelty Detection:**

\[ \text{Gap} = \sum_{i=1}^{m} \alpha_i \max(f(x_i) - \rho, 0) + (C - \alpha_i) \max(\rho - f(x_i), 0) \]

This also makes a good selection criterion for optimization over subsets in other chunking strategies. Similar methods can be used for classification and regression.
Online Learning: Motivation

Problem: Training complexity increases with sample size and number of basis functions. Typically training time scales with $O(m^{2+\gamma})$ and prediction time with $O(m)$.

Problem: Distributions may change over time and we want to have a time dependent predictor.

Problem: Iterative reduced set methods using projection are too expensive.

Design Goal: Mainly local observations should matter for a predictor. Limited time horizon

Design Goal: Cheap and simple update rules for added patterns.

Design Goal: Almost as good as batch learning.
Online Learning: HOWTO

Start with regularized risk functional

\[ R_{\text{reg}}[f] = E[c(x, y, f(x))] + \frac{\lambda}{2}\|f\|^2 \]

in the Reproducing Kernel Hilbert Space.

Perform stochastic gradient descent on \( R_{\text{reg}}[f] \), i.e. at step \( t \) (with \( x_t, y_t \)) replace

\[ E[c(x, y, f(x))] \approx c(x_t, y_t, f(x_t)). \]

and walk \( \lambda \) in the negative gradient direction \( f \rightarrow f - \Lambda \partial_f R_{\text{stoch}}[f, t] \).

Stochastic Gradient:

\[ \partial_f \left[ c(x_t, y_t, f(x_t)) + \frac{\lambda}{2}\|f\|^2 \right] = c'(x_t, y_t, f(x_t))k(x_t, \cdot) + \lambda f. \]
Update Rule

Update in Function Space:

\[ f \rightarrow f - \lambda \partial_f R_{\text{stoch}}[f, t] = (1 - \Lambda \lambda) f - \Lambda c'(x_t, y_t, f(x_t)) k(x_t, \cdot). \]

Update in Coefficient Space:

\[ \alpha_t = -\Lambda c'(x_t, y_t, f(x_t)) \]
\[ \alpha_i = (1 - \Lambda \lambda) \alpha_i \text{ for } i < t \]

We assume a kernel expansion \( f(x) = \sum_{i=1}^{t} \alpha_i k(x_i, x) \)

Finite Time Horizon: Coefficients decay over time. After \( t \) iterations we have

\[ \alpha_i \rightarrow (1 - \lambda \Lambda)^t \alpha_i \]

We can drop \( \alpha_i \) after \( t \) steps with an error of at most \( (1 - \lambda \Lambda)^t \sqrt{k(x_i, x_i)} \).
Update Rules for Novelty Detection

Loss Function:

\[ c(x, f) = \max(0, \rho - f(x)) - \nu \rho \]

Gradient:

\[
\begin{align*}
\partial_f c(x, f) &= \begin{cases} 
-k(x, \cdot) & \text{if } \rho > f(x) \\
0 & \text{otherwise}
\end{cases} \\
\partial_\rho c(x, f) &= \begin{cases} 
1 - \nu & \text{if } \rho > f(x) \\
-\nu & \text{otherwise}
\end{cases}
\end{align*}
\]

Explicit Algorithm:

\[
\begin{align*}
\alpha_t &= 0 \quad | \quad \rho = \rho + \lambda \nu \quad \text{for known patterns} \\
\alpha_t &= \lambda \quad | \quad \rho = \rho + \lambda(\nu - 1) \quad \text{otherwise}
\end{align*}
\]
Update Rules for Online Methods

**Classification:** Soft Margin

\[
\begin{align*}
\alpha_t &= 0 \quad | \rho = \rho + \lambda\nu \quad | b = b \quad | \text{for correct classification} \\
\alpha_t &= \lambda y_t \quad | \rho = \rho + \lambda(\nu - 1) \quad | b = b + \lambda y_t \quad | \text{otherwise}
\end{align*}
\]

**Regression & Time Series:** ε-Insensitive Loss

\[
\begin{align*}
\alpha_t &= 0 \quad | \varepsilon = \varepsilon + \lambda\nu \quad | b = b \quad | \text{for } |f(x_t) - y_t| < \varepsilon \\
\alpha_t &= \lambda \quad | \varepsilon = \varepsilon + \lambda(\nu - 1) \quad | b = b + \lambda \quad | \text{for } f(x_t) - y_t < -\varepsilon \\
\alpha_t &= -\lambda \quad | \varepsilon = \varepsilon + \lambda(\nu - 1) \quad | b = b + \lambda \quad | \text{otherwise}
\end{align*}
\]

**Regression:** Squared Loss

\[
\alpha_t = \lambda(y_t - f(x_t)) \quad \text{and} \quad b = b + \lambda(y_t - f(x_t))
\]
Worst Training Examples
Worst Test Examples
Further Applications

**Convergence:** Theoretical guarantees from the mistake framework (Kivinen, Herbster, etc.).

**Trimmed Estimators:** We can use the $\nu$ trick also for Huber’s robust loss function (not possible for SV batch learning). Simply compute the threshold $\varepsilon$ as in the $\varepsilon$-insensitive case.

**Nonlinear Control Theory:** Kernelization of linear filters. Time series prediction.

**Applications to Bayesian Settings:** Should work just find also for Gaussian Processes, Maximum Entropy settings, etc.
Summary

- Learning as risk minimization
- Feature spaces, kernels and regularization
- The $\nu$-Trick
- Chunking and SMO
- Online Learning
- Applications after the break …

For more information see

http://www.kernel-machines.org
Novelty Detection

**Problems:** fraudulent transactions in credit card database, failed phone connections, ambiguous records in database

**Goal 1:** Find unusual and rare examples in a dataset.

**Solution:** Use an indicator of areas where the density $p(x)$ is low.

- Technique: Single-Class SVM
  - Schölkopf, Platt, Shawe-Taylor, Smola and Williamson, 2000

**Goal 2:** Find mislabelled patterns in database.

**Solution:** Train a classifier and look at Support Vectors and training errors.

- Technique: Standard SVM
  - Guyon, Matic, and Vapnik, 1996
Database Cleaning on USPS

Random Images from the USPS Database

The worst images from the USPS Database (Gaussian Kernel and $\nu = 0.05$).
**Problem:** detect when a jet engine fails. Creating data from engine failure is way too expensive.

**Solution:** Use the spectrum (sound and vibrations) of “normal” working jet engines to calibrate the density estimator.

Hayton, Schölkopf, Tarassenko, Anuzis, 2000

**Performance:** SV novelty detector outperforms neural networks. Furthermore, we obtain meaningful examples of outliers.
Optical Character Recognition

Problem: Recognize handwritten characters, e.g. USPS database of 7,000 digits (16x16 pixels) or NIST database of 60,000 digits (28x28 pixels).

Solution: Train 10 binary classifiers, usually with polynomial kernels $k(x, x') = \langle x, x' \rangle^p$.

Problem: We have invariances in our data (translations of the data to left/right, scaling, slanting, stroke changes, etc.) which will not change the property of a digit.

Solution: Virtual Support Vectors Add transformed Support Vectors (only the boundary matters) into the training set. This leads to the world best classifier on the NIST database.
Elementary Particle Physics

**Goal:** Detect c-quarks from Hadronization experiments (= collision of electrons and positrons) at CERN. Typically almost all of the events will be u, d, s, or c (up, down, strange, charm) quarks. Top and bottom quarks are so rare that they can be ignored.

**Problem:** Physical backwards calculation from the physical sensor measurements (jet energy, momentum, number of jets, particle angles, number of muons, etc) in the calorimeters is too expensive to carry out — we need a numerical shortcut. Moreover, we may reject a large fraction of points.

**Support Vectors:** Use Gaussian rbf kernels (the physicists decreed that the variables were meaningful). Moreover, reject points close to the margin (|f(x)| corresponds to the distance from margin).

The algorithm beat Neural Networks on any dataset made available to us.