Midterm Review

Topics we covered

Machine Learning

Optimization

- Basics of optimization
  - Convexity
  - Unconstrained: GD, SGD
  - Constrained: Lagrange, KKT
  - Duality
- Linear Methods
  - Perceptrons
  - Support Vector Machines
  - Kernels

Statistics

- Basics of probability
  - Tail bounds
  - Density Estimation
  - Exponential Families
- Graphical Models

Systems!

Basics of Machine Learning

- Supervised/Unsupervised Learning?
  - Classification, Regression, Clustering
- Training error/Test error?
- Model Complexity: Overfitting/Underfitting
- True error – Bayes Optimal Error

Bias-Variance Tradeoff

- When estimating a quantity θ, we evaluate the performance of an estimator by computing its risk – expected value of a loss function
  - $\mathrm{R}\left(\theta, \hat{\theta}\right) = E\ L(\theta, \hat{\theta})$, where L could be
    - Mean Squared Error Loss
    - 0/1 Loss
    - Hinge Loss (used for SVMs)
- Bias-Variance Decomposition: $Y = f(x) + \varepsilon$

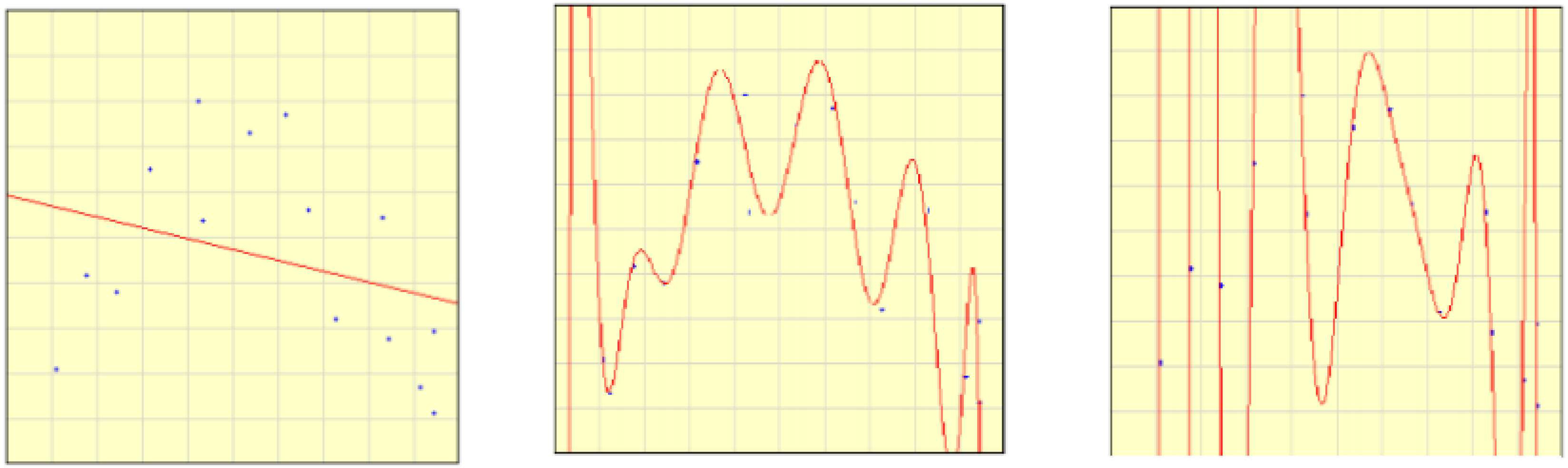
$$Err(x) = E\left[f(x) - \hat{f}(x)^2\right]$$

$$= \ (E\left[\hat{f}(x)\right] - f(x))^2 + E\left[\hat{f}(x) - E\left[\hat{f}(x)\right]\right]^2 + \sigma_\varepsilon{}^2$$

Bias Variance

Bias-Variance Tradeoff

- The choice of hypothesis class introduces a learning bias
  - More complex class: less bias and more variance.



Copied from: Junier Oliva

Training error

- Given a dataset
- Chose a loss function (L_2 for regression for example)

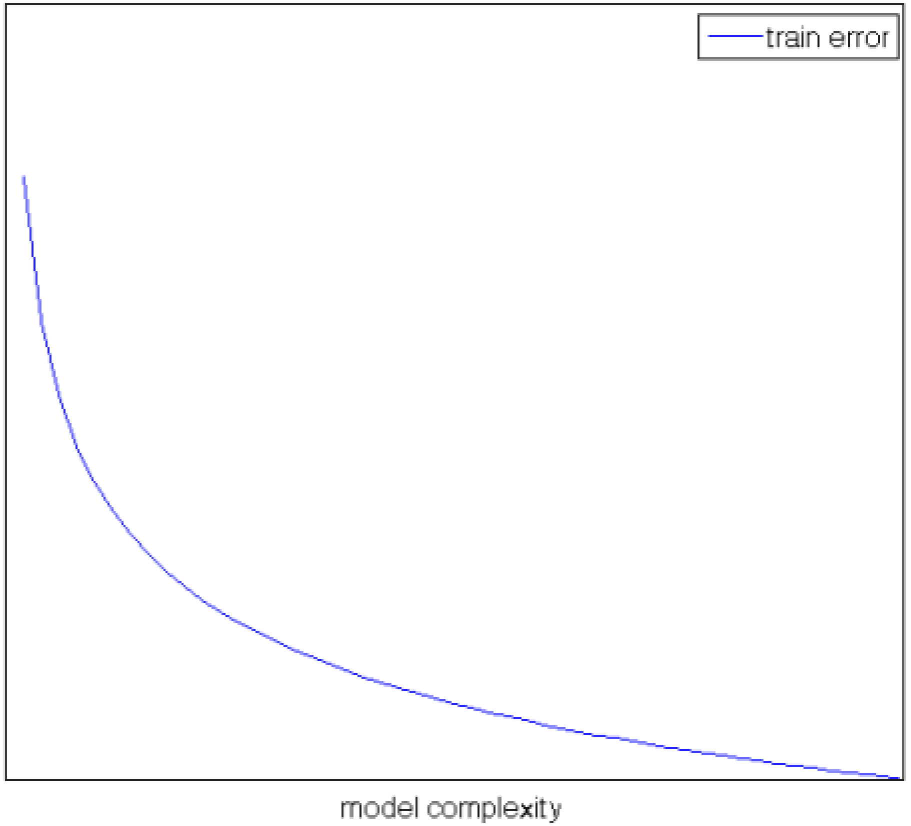
  - Training set error:

$$error_{train} = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \Big(I(y_i \neq h(x)) \Big)$$

$$error_{train} = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \Big(y_i - w.\mathbf{x_i} \Big)^2$$

Copied from: Junier Oliva

Training error as a function of complexity



Copied from: Junier Oliva

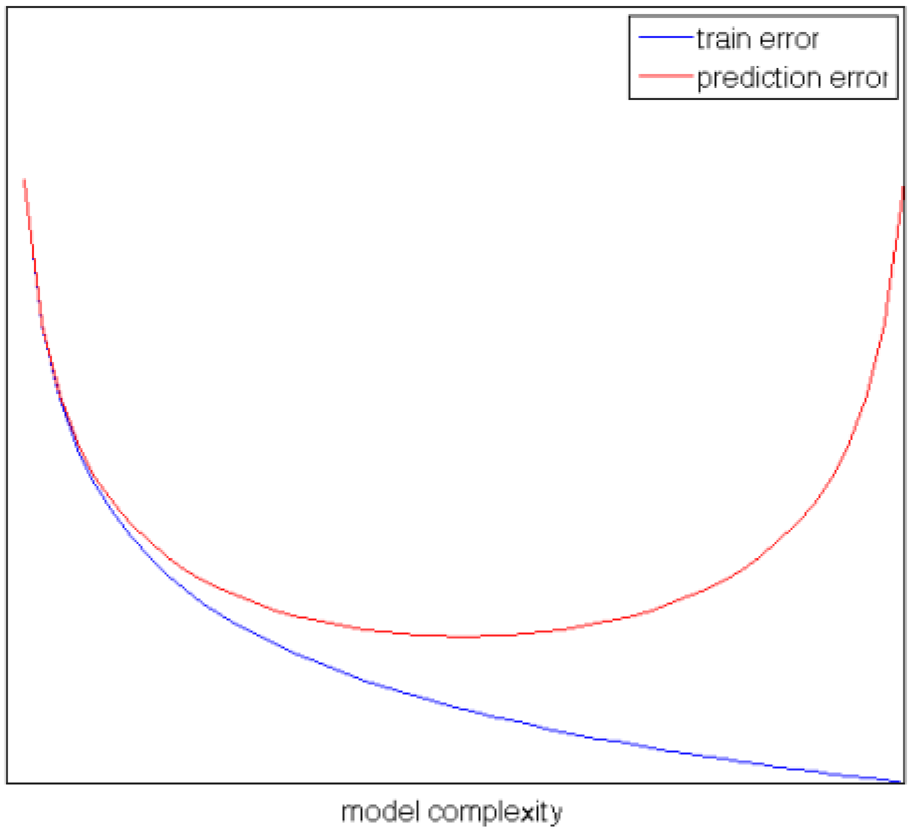
Prediction error

- Training error is not necessary a good measure
- We care about the error over all inputs points:

$$error_{true} = E_x\Big(I(y \neq h(x))\Big)$$

Copied from: Junier Oliva

Prediction error as a function of complexity



Copied from: Junier Oliva

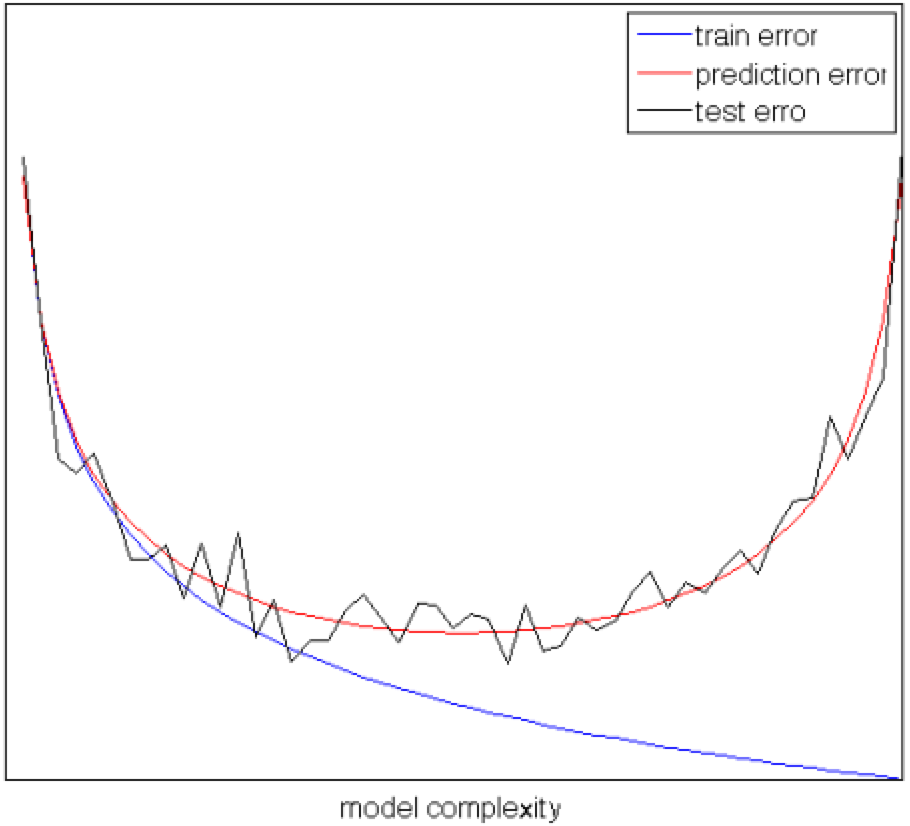
Train-test

- In practice:
  - Randomly divide the dataset into test and train.
  - Use training data to optimize parameters.
  - Test error:

$$error_{test} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \Big(I(y_i \neq h(x_i)) \Big)$$

Copied from: Junier Oliva

Test error as a function of complexity



Copied from: Junier Oliva

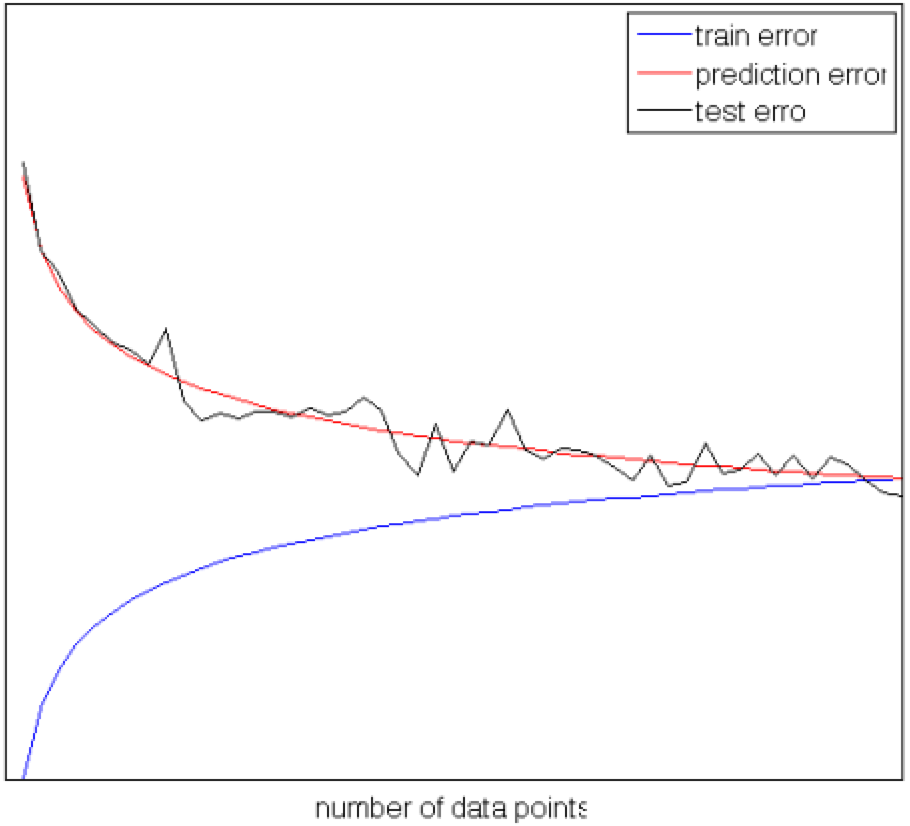
Overfitting

- Overfitting happens when we obtain a model h when there exist another solution h' such that:

$$[error_{train}(h) < error_{train}(h')] \wedge [error_{true}(h) > error_{true}(h')]$$

Copied from: Junier Oliva

Error as a function of data size for fixed complexity



Copied from: Junier Oliva

Regression

- Optimization Problem

$$f(x) = \langle a, x \rangle + b = \langle w, (x, 1) \rangle$$

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \frac{1}{2} (\langle w, \bar{x}_i \rangle - y_i)^2$$

- Solving it

$$0 = \sum_{i=1}^{m} \bar{x}_i (\langle w, \bar{x}_i \rangle - y_i) \Longleftrightarrow \left[\sum_{i=1}^{m} \bar{x}_i \bar{x}_i^\top \right] w = \sum_{i=1}^{m} y_i \bar{x}_i$$

only requires a matrix inversion.

- **Optimization Problem**

$$f(x) = \langle w, \phi(x) \rangle$$

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \frac{1}{2} (\langle w, \phi(x_i) \rangle - y_i)^2$$

- **Solving it**

$$\sum_{i=1}^{m} \phi(x_i) (\langle w, \phi(x_i) \rangle - y_i) \Longleftrightarrow \left[\sum_{i=1}^{m} \phi(x_i) \phi(x_i)^\top \right] w = \sum_{i=1}^{m} y_i \phi(x_i)$$

only requires a matrix inversion.

Optimization

1. Convexity
   - Convex Sets
   - Convex Functions
2. Unconstrained Convex Optimization
   - First-order Methods
   - Newton's Method
3. Constrained Optimization
   - Primal and dual problems
   - KKT conditions

Copied from: Xuezhi Wang

Convex Sets

- Definition
  For $x, x' \in X$ it follows that $\lambda x + (1 - \lambda)x' \in X$ for $\lambda \in [0, 1]$
- Examples
  - Empty set $\emptyset$, single point $\{x_0\}$, the whole space $\mathbb{R}^n$
  - Hyperplane: $\{x \mid a^\top x = b\}$, halfspaces $\{x \mid a^\top x \leq b\}$
  - Euclidean balls: $\{x \mid ||x - x_c||_2 \leq r\}$
  - Positive semidefinite matrices: $\mathbf{S}_+^n = \{A \in \mathbf{S}^n | A \succeq 0\}$ ($\mathbf{S}^n$ is the set of symmetric $n \times n$ matrices)
- Convex Set C, D
  - Translation $\{x + b \mid x \in C\}$
  - Scaling $\{\lambda x \mid x \in C\}$
  - Affine function $\{Ax + b \mid x \in C\}$
  - Intersection $C \cap D$
  - Set sum $C + D = \{x + y \mid x \in C, y \in D\}$

Copied from: Xuezhi Wang

Convex Functions

dom f is convex, $\lambda \in [0, 1]$
$\lambda f(x) + (1 - \lambda) f(y) \geq f(\lambda x + (1 - \lambda) y)$

- **First-order condition**: if f is differentiable,

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x)$$

- **Second-order condition**: if f is twice differentiable,

$$\nabla^2 f(x) \succeq 0$$

- **Strictly convex**: $\nabla^2 f(x) \succ 0$
  Strongly convex: $\nabla^2 f(x) \succeq dI$ with $d > 0$

Copied from: Xuezhi Wang

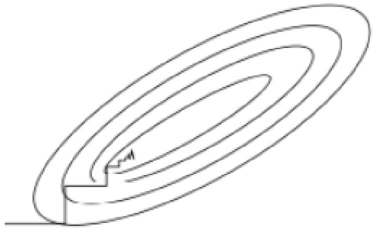
Convex Functions – Examples

- Exponential. e^{ax} convex on $\mathbb{R}$, any $a \in \mathbb{R}$
- Powers. x^a convex on $\mathbb{R}_{++}$ when $a \geq 1$ or $a \leq 0$, and concave for $0 \leq a \leq 1$.
- Powers of absolute value. $|x|^p$ for $p \geq 1$, convex on $\mathbb{R}$.
- Logarithm. $\log x$ concave on $\mathbb{R}_{++}$.
- Norms. Every norm on $\mathbb{R}^n$ is convex.
- $f(x) = \max\{x_1, ..., x_n\}$ convex on $\mathbb{R}^n$
- Log-sum-exp. $f(x) = \log(e^{x_1} + ... + e^{x_n})$ convex on $\mathbb{R}^n$.

Useful Observations

- A function is convex if and only if its epigraph is a convex set.
- Below-Sets of Convex Functions is a convex set
- Convex functions cannot have local minima

Gradient Descent



given a starting point $x \in \mathbf{dom} f$.
repeat
1. $\Delta x := -\nabla f(x)$
2. Choose step size t via exact or backtracking line search.
3. update. $x := x + t\Delta x$.

Until stopping criterion is satisfied.

- Key idea
  - Gradient points into descent direction
  - Locally gradient is good approximation of objective function

Copied from: Xuezhi Wang

Newton's Method

Goal: $\phi : \mathbb{R} \rightarrow \mathbb{R}$

$$\phi(x^*) = 0$$

$$x^* = ?$$

Linear Approximation (1st order Taylor approx**):**

$$\underbrace{\phi(\underbrace{x + \Delta x}_{x^*})}_{\phi(x^*) = 0} = \phi(x) + \phi'(x)\Delta x + \underbrace{o(|\Delta x|)}_{\text{NEGLIGABLE}}$$

Therefore,

$$0 \approx \phi(x) + \phi'(x)\Delta x$$

$$x^* - x = \Delta x = -\frac{\phi(x)}{\phi'(x)}$$

$$x_{k+1} = x_k - \frac{\phi(x)}{\phi'(x)}$$

Copied from: Prof Barnabas

Newton's Method

$f : \mathbb{R}^n \to \mathbb{R}$, f is differentiable.

$$\min_{x \in \mathbb{R}^n} f(x)$$

We need to find the roots of $\nabla f(x) = 0_n$

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n$$

Newton system: $\nabla f(x) + \nabla^2 f(x) \Delta x = 0_n$

Newton step: $\Delta x = x_{k+1} - x_k = -[\nabla^2 f(x)]^{-1} \nabla f(x)$

Iterate until convergence, or max number of iterations exceeded

Copied from: Prof Barnabas

Duality

Primal problem:

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{subject to } h_i(x) \leq 0, i = 1, \ldots, m$$

Lagrangian:

$$L(x, u) = f(x) + \sum_{i=1}^{m} u_i h_i(x)$$

where $u \in \mathbb{R}^m$ and $u \geq 0$.
Lagrange dual function:

$$g(u) = \min_{x \in \mathbb{R}^n} L(x, u)$$

Duality

Dual problem:

$$\max_u g(u)$$
$$\text{subject to } u \geq 0$$

- Dual problem is a convex optimization problem, since g is always concave (even if primal problem is not convex)

- The primal and dual optimal values always satisfy weak duality: $f^* \geq g^*$

- **Slater's condition**: for convex primal, if there is an x such that $h_1(x) < 0, ..., h_m(x) < 0$ and $l_1(x) = 0, ..., l_r(x) = 0$ then strong duality holds: $f^* = g^*$. Or equivalently Karlin's or strict constraint qualification.

KKT Conditions

If x^*, u^*, v^* are primal and dual solutions, with zero duality gap (strong duality holds), then x^*, u^*, v^* satisfy the KKT conditions:

- stationarity: $0 \in \partial f(x^*) + \sum u_i^* \partial h_i(x^*)$
- complementary slackness: $u_i^* h_i(x^*) = 0$ for all i
- primal feasibility: $h_i(x^*) \leq 0$ for all i
- dual feasibility: $u_i^* \geq 0$ for all i

Perceptrons

initialize $w = 0$ and $b = 0$
repeat
  if $y_i [\langle w, x_i \rangle + b] \leq 0$ **then**
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$

Convergence of Perceptrons

- If there exists some (w^*, b^*) with unit length and
$$y_i \left[\langle x_i, w^* \rangle + b^*\right] \geq \rho \text{ for all } i$$
then the perceptron converges to a linear separator after a number of steps bounded by
$$\left({b^*}^2 + 1\right)\left(r^2 + 1\right)\rho^{-2} \text{ where } \|x_i\| \leq r$$
- Dimensionality independent
- Order independent (i.e. also worst case)
- Scales with 'difficulty' of problem

Back to Optimization

- A typical machine learning problem has a penalty/regularizer + loss form

$$\min_w F(w) = g(w) + \frac{1}{n}\sum_{i=1}^{n} f(w; y_i, x_i),$$

  $x_i, w \in \mathbb{R}^p$, $y_i \in \mathbb{R}$, both g and f are convex
- Today we only consider differentiable f, and let $g = 0$ for simplicity
- For example, let $f(w; y_i, x_i) = -\log p(y_i|x_i, w)$, we are trying to maximize the log likelihood, which is

$$\max_w \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, w)$$

Gradient Descent

- choose initial $w^{(0)}$, repeat

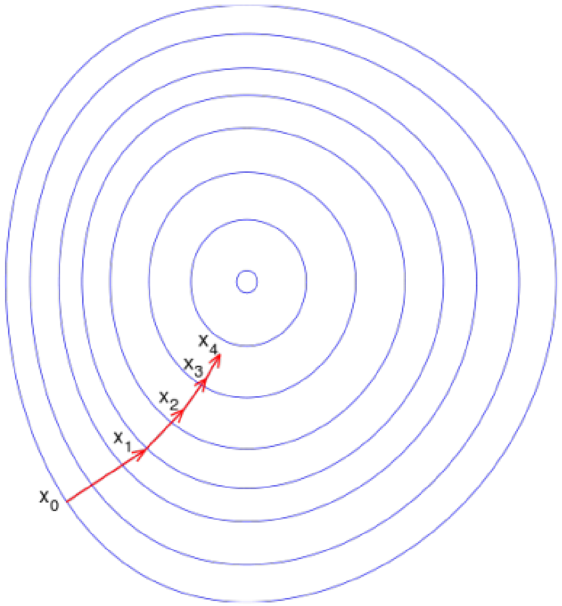
$$w^{(t+1)} = w^{(t)} - \eta_t \cdot \nabla F(w^{(t)})$$

  until stop
- η_t is the learning rate, and

$$\nabla F(w^{(t)}) = \frac{1}{n} \sum_i \nabla_w f(w^{(t)}; y_i, x_i)$$

- How to stop? $\|w^{(t+1)} - w^{(t)}\| \leq \epsilon$ or $\|\nabla F(w^{(t)})\| \leq \epsilon$

Two dimensional example:



Stochastic Gradient Descent

- We name $\frac{1}{n}\sum_i f(w; y_i, x_i)$ the empirical loss, the thing we hope to minimize is the expected loss

$$f(w) = \mathbb{E}_{y_i, x_i} f(w; y_i, x_i)$$

- Suppose we receive an infinite stream of samples (y_t, x_t) from the distribution, one way to optimize the objective is

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla_w f(w^{(t)}; y_t, x_t)$$

- On practice, we simulate the stream by randomly pick up (y_t, x_t) from the samples we have
- Comparing the average gradient of GD $\frac{1}{n}\sum_i \nabla_w f(w^{(t)}; y_i, x_i)$

SGD and Perceptron

- Recall Perceptron: initialize w, repeat

$$w = w + \begin{cases} y_i x_i & \text{if } y_i \langle w, x_i \rangle < 0 \\ 0 & \text{otherwise} \end{cases}$$

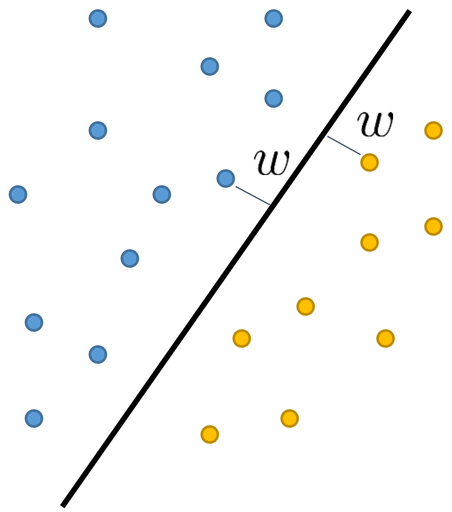
- Fix learning rate $\eta = 1$, let $f(w; y, x) = \max(0, -y_i \langle w, x_i \rangle)$, then

$$\nabla_w f(w; y, x) = \begin{cases} -y_i x_i & \text{if } y_i \langle w, x_i \rangle < 0 \\ 0 & \text{otherwise} \end{cases}$$

  we derive Perceptron from SGD

SVM Primal

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Hard Margin

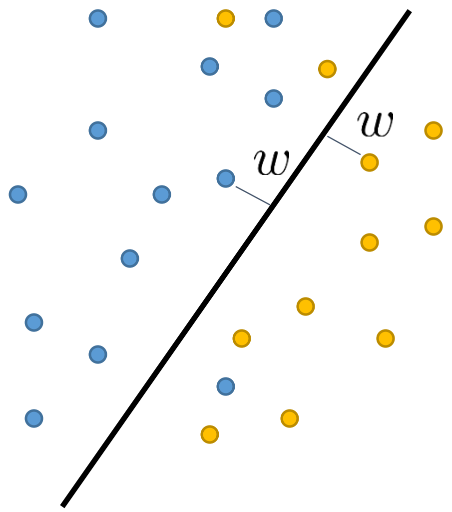
$$\min_{w,b} ||w||^2$$

$$\text{subject to } (\langle w, x_i \rangle + b) y_i \geq 1$$

Copied from: Junier Oliva

SVM Primal

Find maximum margin hyper-plane $$f(x) = \langle w, x \rangle + b = 0$$



Soft Margin

$$\min_{w,b} ||w||^2 + C \sum_i \xi_i$$

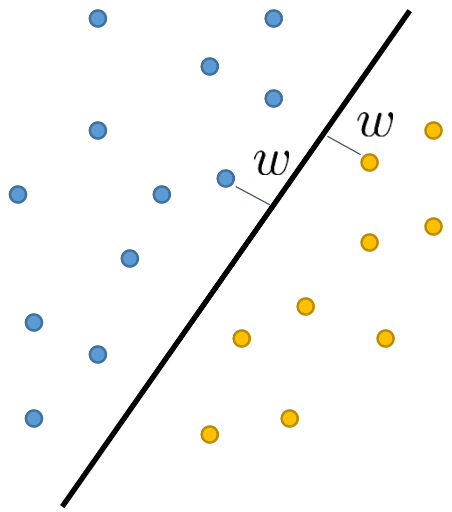
$$\text{subject to } (\langle w, x_i \rangle + b) y_i \geq 1$$

$$\xi_i \geq 0$$

Copied from: Junier Oliva

SVM Dual

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the hard margin SVM

$$\mathcal{L}(w, \alpha) = \frac{1}{2}\langle w, w \rangle - \sum_i \alpha_i \Big[(\langle w, x_i \rangle + b) y_i - 1 \Big]$$

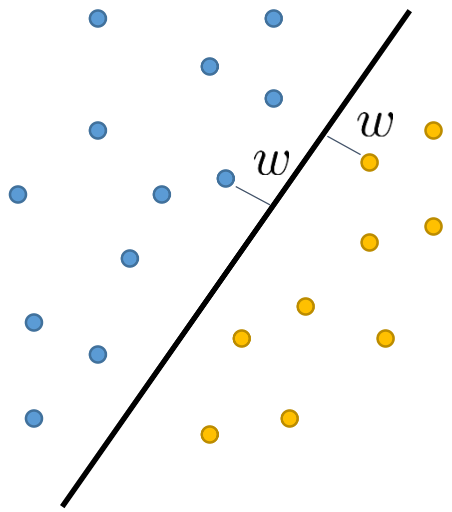
$$\alpha_i \geq 0$$

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \quad \rightarrow \quad w = \sum_i \alpha_i y_i x_i$$

Copied from: Junier Oliva

SVM Dual

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the hard margin SVM

$$\mathcal{L}(w, \alpha) = \frac{1}{2}\langle w, w \rangle - \sum_i \alpha_i \Big[(\langle w, x_i \rangle + b) y_i - 1 \Big]$$

$$\alpha_j \geq 0$$

Substituting α for w

$$w = \sum_i \alpha_i y_j x_j$$

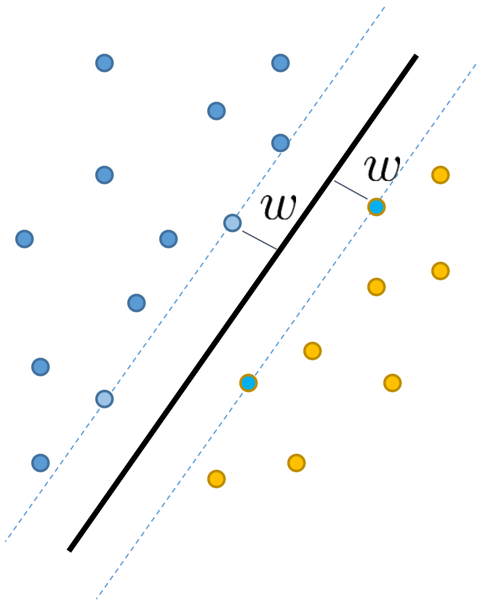
$$\langle w, w \rangle = \sum_{i,j} \langle \alpha_i y_i x_i, \alpha_j y_j x_j \rangle$$

$$= \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

Copied from: Junier Oliva

SVM Dual

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the hard margin SVM

$$\mathcal{L}(w, \alpha) = \frac{1}{2}\langle w, w \rangle - \sum_i \alpha_i \left[(\langle w, x_i \rangle + b) y_i - 1 \right]$$

$$\alpha_j \geq 0$$

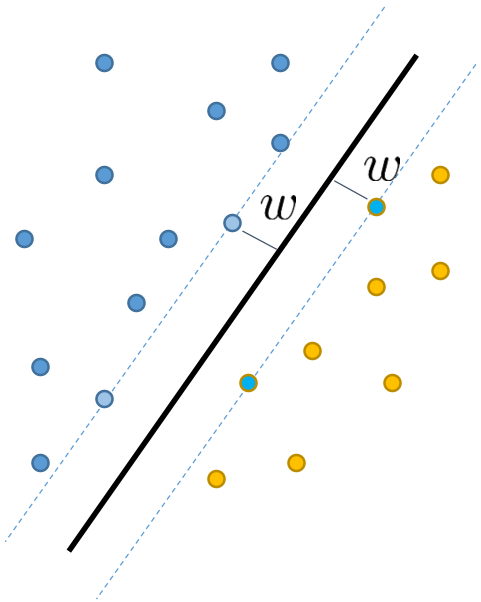
The constraints are active for the support vectors

$$\forall k \text{ s.t. } a_k > 0 \qquad b = y_k - \langle w, x_k \rangle$$

Copied from: Junier Oliva

SVM Dual

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the hard margin SVM

$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$

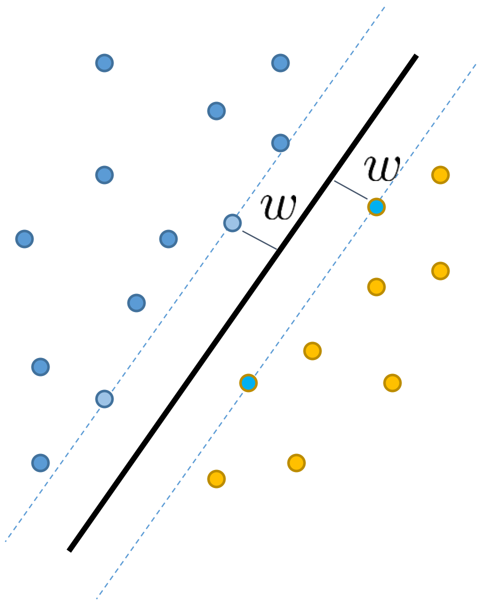
$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Copied from: Junier Oliva

SVM – Computing w

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the hard margin SVM

$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

solve, get α_i

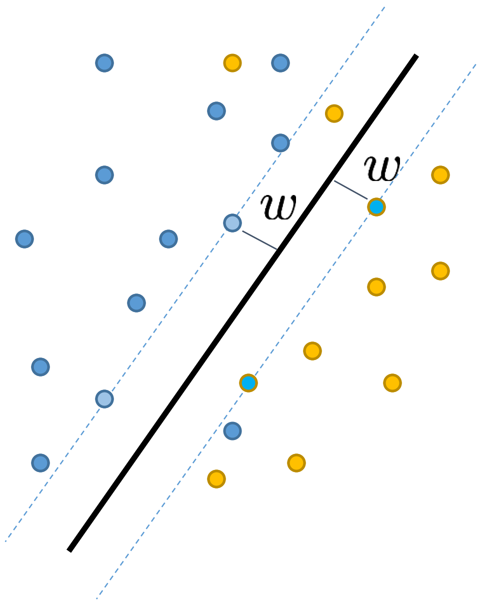
$$w = \sum_i \alpha_i y_i x_i$$

$$b = y_k - \langle w, x_k \rangle \qquad \forall k \text{ for which } \alpha_k > 0$$

Copied from: Junier Oliva

SVM – Computing w

Find maximum margin hyper-plane $f(x) = \langle w, x \rangle + b = 0$



Dual for the soft margin SVM

$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$

$$\sum_i \alpha_i y_i = 0$$

only difference from the separable case → $C \geq \alpha_i \geq 0$

solve, get α_i

$$w = \sum_i \alpha_i y_i x_i$$

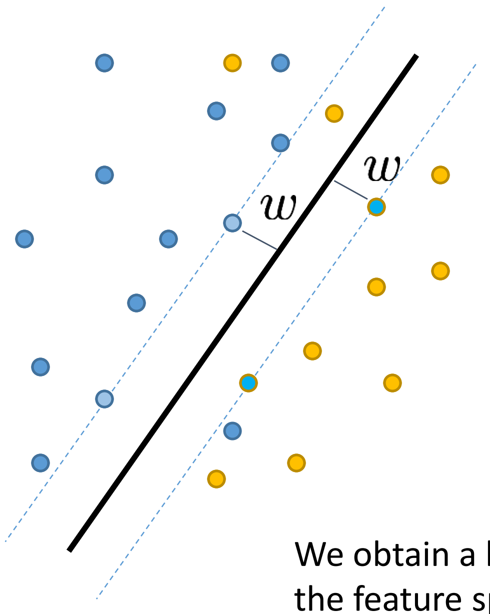
$$b = y_k - \langle w, x_k \rangle \qquad \forall k \text{ for which } C > \alpha_k > 0$$

Copied from: Junier Oliva

SVM – the feature map

Find maximum margin hyper-plane $f(x) = \langle w, \Phi(x) \rangle + b = 0$

But data is not linearly separable ☹



$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle + \sum_i \alpha_i$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

We obtain a linear separator in the feature space.

!! $M >> m$

$\Phi(x)$ is expensive to compute!

feature map

inputs

features

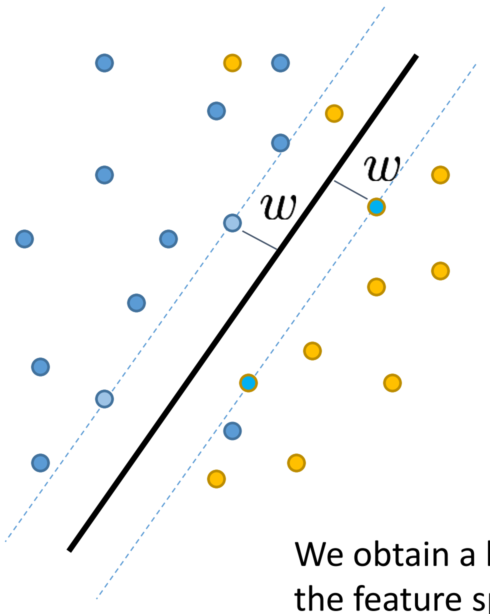
$$x \rightarrow \Phi(x)$$

$$\in R^m \qquad \in R^M$$

Copied from: Junier Oliva

Introducing the kernel

The dual formulation no longer depends on w, only on a dot product!



$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle + \sum_i \alpha_i$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

We obtain a linear separator in the feature space.

!! $M >> m$

$\Phi(x)$ is expensive to compute!

But we don't have to!

What we need is the dot product:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

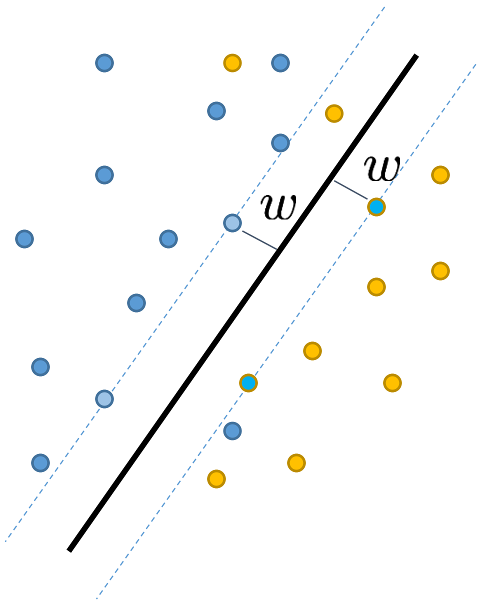
Let's call this a kernel

- 2-variable function
- can be written as a dot product

Copied from: Junier Oliva

Kernel SVM

The dual formulation no longer depends on w, only on a dot product!



$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_i \alpha_i$$

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

closed form

This is the famous 'kernel trick'.

- never compute the feature map
- learn using the closed form K
- constant time for HD dot products

Copied from: Junier Oliva

Kernel SVM –Run time

What happens when we need to classify some x_0?

Recall that w depends on α

$$
\begin{aligned}
w &= \sum_i \alpha_i y_i \Phi(x_i) \\
b &= y_k - \langle w, \Phi(x_k) \rangle \\
&\quad \forall k \text{ s.t. } C > \alpha_k > 0
\end{aligned}
$$

Our classifier for x_0 uses w

$$sign(\langle w, \Phi(x_0) \rangle + b)$$

Copied from: Junier Oliva

Kernel SVM –Run time

What happens when we need to classify some x_0?

Recall that w depends on α

$$
\begin{aligned}
w &= \sum_i \alpha_i y_i \Phi(x_i) \\
b &= y_k - \langle w, \Phi(x_k) \rangle \\
&\quad \forall k \text{ s.t. } C > \alpha_k > 0
\end{aligned}
$$

Our classifier for x_0 uses w

$$sign(\langle w, \Phi(x_0) \rangle + b)$$

Who needs w when we've got dot products?

$$
\begin{aligned}
\langle w, \Phi(x_0) \rangle &= \sum_i \alpha_i y_i K(x_0, x_i) \\
b &= y_k - \sum_i \alpha_i y_i K(x_k, x_i) \\
&\quad k \rightarrow \text{ support vectors}
\end{aligned}
$$

Copied from: Junier Oliva

Kernel SVM Recap

Pick kernel

Solve the optimization to get α

$$\max_{\alpha} \quad -\frac{1}{2}\sum_i \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_i \alpha_i$$

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

Compute b using the support vectors

$$b \quad = \quad y_k - \sum_i \alpha_i y_i K(x_k, x_i)$$

Classify as

$$sign\Big(\sum_i \alpha_i y_i K(x_0, x_i) + b\Big)$$

Copied from: Junier Oliva

Reminder on Kernels

- Remember Kernels are nothing but implicit feature maps

$$\phi : \mathcal{X} \to \mathbb{R}^d$$

- Gram Matrix
  - of a set of vectors $x_1 \ldots x_n$ in the inner product space defined by the kernel K
  - $G_{ij} = K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \qquad \forall i, j \in 1 \ldots n$
- Gram Matrix is always positive definite

Bayes Rule

- Joint Probability

$$\Pr(X, Y) = \Pr(X|Y)\Pr(Y) = \Pr(Y|X)\Pr(X)$$

- Bayes Rule

$$\Pr(X|Y) = \frac{\Pr(Y|X) \cdot \Pr(X)}{\Pr(Y)}$$

- Hypothesis testing
- Reverse conditioning

Law of Large Numbers

- Random variables x_i with mean $\mu = \mathbf{E}[x_i]$
- Empirical average $\hat{\mu}_n := \frac{1}{n}\sum_{i=1}^{n} x_i$

- Weak Law of Large Numbers

$$\lim_{n\to\infty} \Pr\left(|\hat{\mu}_n - \mu| \leq \epsilon\right) = 1 \text{ for any } \epsilon > 0$$

- Strong Law of Large Numbers

$$\Pr\left(\lim_{n\to\infty} \hat{\mu}_n = \mu\right) = 1$$

this means convergence in probability

Central Limit Theorem

- Independent random variables x_i with mean μ_i and standard deviation σ_i
- The random variable

$$z_n := \left[\sum_{i=1}^{n} \sigma_i^2\right]^{-\frac{1}{2}} \left[\sum_{i=1}^{n} x_i - \mu_i\right]$$

converges to a Normal Distribution $\mathcal{N}(0, 1)$

- Special case - IID random variables & average

$$\frac{\sqrt{n}}{\sigma}\left[\frac{1}{n}\sum_{i=1}^{n} x_i - \mu\right] \to \mathcal{N}(0, 1)$$

$O\left(n^{-\frac{1}{2}}\right)$ convergence

Tail Bounds

Markov Inequality: If X is any nonnegative integrable random variable and $a > 0$, then

$$\Pr(X > a) \leq \frac{\mathbb{E}[X]}{a}$$

Chebyshev Inequality: If X is any random variable with mean μ and variance σ^2. Then for any $\epsilon > 0$, we have

$$\Pr(|X - \mu| > \epsilon) \leq \frac{\sigma^2}{\epsilon^2}$$

More Tail Bounds

The Chernoff Bound: Suppose $Y_1, ..., Y_r$ are i.i.d. random variables, taking values in $\{0, 1\}$. Let $p = E[Y_i]$ and > 0. Then

$$\Pr\left(\sum_i Y_i > nq\right) \leq \exp(-rD(q||p))$$

Hoeffding's Inequality: Suppose $Y_1, ..., Y_r$ are i.i.d. random variables, taking values in $(a_i, b_i\}$. Then

$$\Pr\left(\left|\sum_i (Y_i - \mathbb{E}[Y_i]\right| > t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^r (b_i - a_i)^2}\right)$$

Union Bound: set of events $A_1, A_2, A_3, ...$, we have

$$\Pr\left(\bigcup_i A\right) \leq \sum_i \Pr(A_i)$$


A
B
2 for $40
5 for $25
TODAY ONLY!
$10 OFF $50+
PLUS
FREE SHIPPING ON $100+
Details and offer codes below.
FREE
SPRING'S BEAUTY ESSENTIALS
with any $40 beauty purchase
find us on facebook

A/B testing