# Machine Learning Cheat Sheet

## General

### Dataset and Definitions

$\mathcal{D}$ is a set of training examples, the n-th Training Example ($n = 1, 2, ..., N$), of this set is:
$\mathbf{x_n} = \begin{bmatrix} x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix}$

We write all N training examples in a Matrix:
$\mathbf{X} = [\mathbf{x_1}; \quad \mathbf{x_2}; \quad \dots; \quad \mathbf{x_N}]$

In supervised learning, you are also given a set of observations corresponding to the training examples:
$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_N \end{bmatrix}^{\mathbf{T}}$

We assume a *true* model behind the data, the observations are noisy versions of this ground truth:
$y = y_{true} + \epsilon$.

The final goal is to predict a $\hat{y}$ given any $\mathbf{x}$.

### Distributions

Gaussian distribution: $\mathcal{N}(X|\mu, \sigma^2)$
$\implies p(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$

Poisson distribution: $\mathcal{P}(X|\lambda)$
$\implies p(X = k) = \frac{\lambda^k}{k!} \exp(-\lambda)$

## Regression

Simple linear regression: $y_n \approx \beta_0 + \beta_1 x_{n1}$

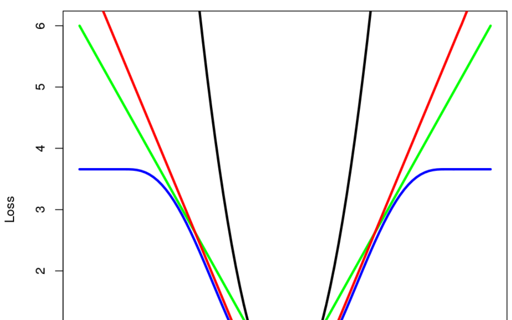
Multiple linear regression:
$y_n \approx f(\mathbf{x_n}) := \beta_0 + \beta_1 \mathbf{x_{n1}} + \beta_2 \mathbf{x_{n2}} + \dots + \beta_D \mathbf{x_{nD}}$

### Linear basis function model

$y_n = \beta_0 + \sum_{i=1}^{M} \beta_i \phi_i(\mathbf{x_n}) = \tilde{\phi}^{\mathbf{T}}(\mathbf{x_n^T})\boldsymbol{\beta}$. The optimal $\beta$ is given by $\beta = (\tilde{\Phi}^T\tilde{\Phi})^{-1}\tilde{\Phi}^T y$ where $\tilde{\Phi}$ is a matrix with N rows and the n-th row is $[1, \phi_1(x_n)^T, ..., \phi_M(x_n)^T]$.

Ridge regression: $\beta_{ridge} = (\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}} + \lambda \boldsymbol{I})^{-1}\tilde{\boldsymbol{\Phi}}^T \boldsymbol{y}$

### Cost functions




Cost function / Loss: $\mathcal{L}(\boldsymbol{\beta}) = \mathcal{L}(\mathcal{D}, \boldsymbol{\beta})$

Mean square error (MSE): $\frac{1}{2N}\sum_{n=1}^{N}[y_n - f(\mathbf{x_i})]^2$

MSE Matrix Formulation: $\frac{1}{2N}(\mathbf{y} - \mathbf{X}\beta)^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\beta)$

Mean absolute error (MAE): $\frac{1}{2N}\sum_{n=1}^{N}|y_n - f(\mathbf{x_i})|$

Huber loss: $\mathcal{L}_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$

Root mean square error (RMSE): $\sqrt{2 * \text{MSE}}$

Epsilon insensitive (used for SVMs):
$\mathcal{L}_\epsilon(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| \le \epsilon, \\ |y - \hat{y}| - \epsilon, & \text{otherwise.} \end{cases}$

## Grid Search

Complexity: $\mathcal{O}(M^D N D)$, where $M$ is the number of grids in one dimension.

## Gradient Descent

General rule: $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha \frac{\partial \mathcal{L}(\boldsymbol{\beta}^{(k)})}{\partial \boldsymbol{\beta}}$

Complexity: $\mathcal{O}(IND)$ where $I$ is the number of iterations we take.

Big questions are how to get a good $\alpha$.

The gradient for MSE comes out as:
$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = -\frac{1}{N}\tilde{X}^T(\boldsymbol{y} - \tilde{X}\boldsymbol{\beta})$

## Least squares

Complexity: $\mathcal{O}(ND^2 + D^3)$ The gradient of the MSE set to zero is called the normal equation:

$$-2\mathbf{y}^{\mathbf{T}}\mathbf{X} + 2\mathbf{X}^{\mathbf{T}}\mathbf{X}\beta = \mathbf{0}$$

We can solve this for $y$ directly (by Matrix manipulation) $\beta = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y$

## Classification

Logistic Function $\sigma = \frac{exp(x)}{1+exp(x)}$

Classification with linear regression: Use $y = 0$ as class $\mathcal{C}_\infty$ and $y = 1$ as class $\mathcal{C}_\in$ and then decide a newly estimated $y$ belongs to $\mathcal{C}_\infty$ if $y < 0.5$.

### Logistic Regression

$\tilde{\mathbf{X}}^T[\sigma(\tilde{\mathbf{X}}\beta) - y] = 0$

TODO: Generalized Linear model

### Cost functions

Root Mean square error (RMSE):
$\sqrt{\frac{1}{N}\sum_{n=1}^{N}[y_n - \hat{p_n}]^2}$

0-1 Loss: $\frac{1}{N}\sum_{n=1}^{N}\delta(y_n, \hat{y_n})$

logLoss:
$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(\hat{p_n}) + (1 - y_n)\log(1 - \hat{p_n})$

## Maximum Likelihood

The Likelihood Function maps the model parameters to the probability distribution of $\mathbf{y}$:
$\mathcal{L}_{lik}$ : parameter space $\to [0; 1] \quad \beta \mapsto \mathbf{p}(\mathbf{y} \mid \beta)$ An underlying $p$ is assumed before. If the observed $y$ are IID, $p(\mathbf{y} \mid \beta) = \prod_{\mathbf{n}} \mathbf{p}(\mathbf{y_n} \mid \beta)$.
$\mathcal{L}_{lik}$ can be viewed as just another cost function. Maximum likelihood then simply choses the parameters $\beta$ such that observed data is most likely.
$\beta = \arg\max_{\text{all}\beta} L(\beta)$
Assuming different $p$ is basically what makes this so flexible. We can chose e.g.:

| | |
|---|---|
| Gaussian $p$ | $\mathcal{L}_{lik} \widehat{=} \mathcal{L}_{MSE}$ |
| Poisson $p$ | $\mathcal{L}_{lik} \widehat{=} \mathcal{L}_{MAE}$ |

## Bayesian methods

Bayes rule: $p(A, B) = p(A|B)p(B) = p(B|A)p(A)$

The **prior** $p(\mathbf{f}|\mathbf{X})$ encodes our prior belief about the "true" model $\mathbf{f}$. The **likelihood** $p(\mathbf{y}|\mathbf{f})$ measures the probability of our (possibly noisy) observations given the prior.

Least-squares tries to find model parameters $\beta$ which maximize the likelihood. Ridge regression maximizes the **posterior** $p(\beta|\mathbf{y})$

### Graphical Models

TODO: Bayes Net: Directed acyclic graph

TODO: Belief propagation

Graph between the observations and the variables is a bi-partite graph.

## Kernel

Basically, Kernels are a mean to measure distance, or "similarity" of two vectors. We define:
$(\mathbf{K})_{\mathbf{i,j}} = \kappa(\mathbf{x_i}, \mathbf{x_j}) = \tilde{\phi}(\mathbf{x_i})^{\mathbf{T}}\tilde{\phi}(\mathbf{x_j})$.
The $\phi$ are not that important in the end, because we only use the Kernel as is. Sometimes it's even impossible to write them down explicitly.

| | |
|---|---|
| Linear | $\kappa(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$ |
| Polynomial | $\kappa(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}^{\mathbf{T}}\mathbf{x_j} + \mathbf{c})^{\mathbf{d}}$ |
| RBF | $\kappa(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\lVert\mathbf{x_i} - \mathbf{x_j}\rVert^2}{2\sigma^2}\right)$ |

## Neural Networks

TODO: Intuition and notation, backpropagation, regularization techniques

## Support Vector Machines

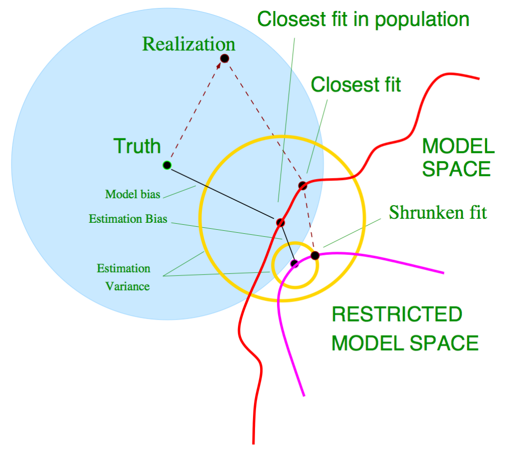
Search for the hyperplane separating the data such that the gap is biggest. It minimizes the following cost function:
$\mathcal{L}_{SVM}(\beta) = \sum_{\mathbf{n=1}}^{\mathbf{N}}[\mathbf{1} - \mathbf{y_n}\tilde{\phi}_\mathbf{n}\beta]_+ + \frac{\lambda}{2}\sum_{\mathbf{j=1}}^{\mathbf{M}}\beta_\mathbf{j}^2$
This is convex but not differentiable.

## Concepts

### Occam's Razor

It states that among competing hypotheses, the one with the fewest assumptions should be selected. Other, more complicated solutions may ultimately prove correct, butin the absence of certaintythe fewer assumptions that are made, the better.

### Bias-Variance Decomposition




Bias-variance come directly out of the test error:

$$\begin{aligned}
\overline{teErr} &= E[(\text{observation} - \text{prediction})^2] = E[(y - \hat{y})^2] \\
&= E[(y - y_{true} + y_{true} - \hat{y})^2] \\
&= \underbrace{E[(y - y_{true})^2]}_{\text{var of measurement}} + E[(y_{true} - \hat{y})^2] \\
&= \sigma^2 + E[(y_{true} - E[\hat{y}] + E[\hat{y}] - \hat{y})^2] \\
&= \sigma^2 + \underbrace{E[(y_{true} - E[\hat{y}])^2]}_{\text{pred bias}^2} + \underbrace{E[(E[\hat{y}] - \hat{y})^2]}_{\text{pred variance}}
\end{aligned}$$

| | bias | variance |
|---|---|---|
| regularization | + | - |
| reduce model complexity | + | - |
| more data | - | |

### Consistency

An estimator is said to be consistent, if it eventually recovers the true parameters that generated the data as the sample size goes to infinity. Of course, this only makes sense if the data actually comes from the specified model, which is not usually the case. Nevertheless, it can be a useful theoretical property.

## Convexity

f is called convex f: $\forall x_1, x_2 \in X, \forall t \in [0, 1]$ :
$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$.

k-fold cross-validation, definition of Test-Error, Train-Error

TODO: statistical/computational tradeoff

Definitions of: Efficiency, Optimality, Identifiability