# Machine Learning Cheat Sheet

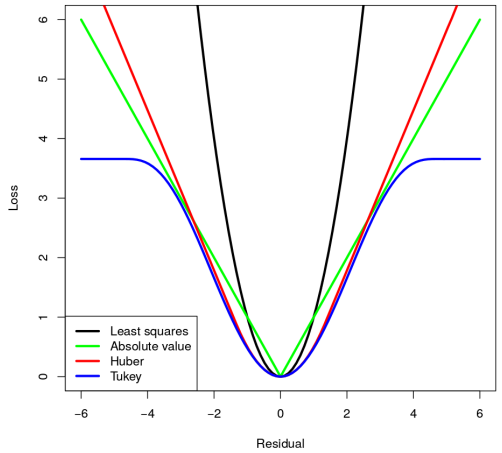
## Regression

We have a set of $N$ training examples of dimensionality $D$, e.g:
$x_n = \begin{bmatrix} x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix}^T$
We often put the examples into one matrix with extra column of 1s:
$\widetilde{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ & & \dots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}$
(note that $x_n$ are rows, not columns)
The goal is to predict a $\hat{y}$ given $x$.
Simple linear regression: $y_n \approx \beta_0 + \beta_1 x_{n1}$
Multiple dimension linear regression:
$y_n \approx f(x^*) := \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_D x_D^*$

### Linear basis function model

$y_n = \beta_0 + \sum_{i=1}^{M} \beta_i \phi_i(\mathbf{x_n}) = \widetilde{\phi}^{\mathbf{T}}(\mathbf{x_n^T})\boldsymbol{\beta}$. The optimal $\beta$ is given by $\beta = (\widetilde{\Phi}^T\widetilde{\Phi})^{-1}\widetilde{\Phi}^T y$ where $\widetilde{\Phi}$ 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: $\beta_{ridge} = (\widetilde{\boldsymbol{\Phi}}^T\widetilde{\boldsymbol{\Phi}} + \lambda \boldsymbol{I})^{-1}\widetilde{\boldsymbol{\Phi}}^T \boldsymbol{y}$

### Cost functions




Cost function / Loss: $\mathcal{L}(\boldsymbol{\beta}) = \mathcal{L}(\mathcal{D}, \boldsymbol{\beta})$
Mean square error (MSE): $\frac{1}{2N}\sum_{n=1}^{N}[y_n - f(\mathbf{x_i})]^2$
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):
$\mathcal{L}_\epsilon(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| \le \epsilon, \\ |y - \hat{y}| - \epsilon, & \text{otherwise.} \end{cases}$

TODO: statistical/computational tradeoff

## Gradient Descent

General rule: $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha\frac{\partial \mathcal{L}(\boldsymbol{\beta}^{(k)})}{\partial \boldsymbol{\beta}}$
How to get a good $\alpha$ is a hard question (too small gives slow time, too big may not converge).
The gradient for MSE comes out as:
$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = -\frac{1}{N}\widetilde{X}^T(\boldsymbol{y} - \widetilde{X}\boldsymbol{\beta})$

## Normal equations

$\beta = (\widetilde{X}^T\widetilde{X})^{-1}\widetilde{X}^T y$
Works for $\widetilde{X}$ having $rank = D$ (so that inverse is defined)

## 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

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

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

## 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.

## Math

convexity:
$\forall_{x_1,x_2}\forall_{t\in[0,1]} f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2)$
Jensen's inequality (log is **concave**):
$log(\frac{\sum_{i=1}^{n} x_i}{n}) \ge \frac{\sum_{i=1}^{n} log(x_i)}{n}$
Hessian is positive semidefinite $\Rightarrow$ function is convex.
Positive semidefinite matrix $M \Leftrightarrow$ Gram matrix (inner product) $\Leftrightarrow \forall_x x^T M x \ge 0$.

## Difficult words

**consistent** estimator converges to the true value as we increase the number of data to infinity.
**unidentifiable** model has many global minima due to symmetry.

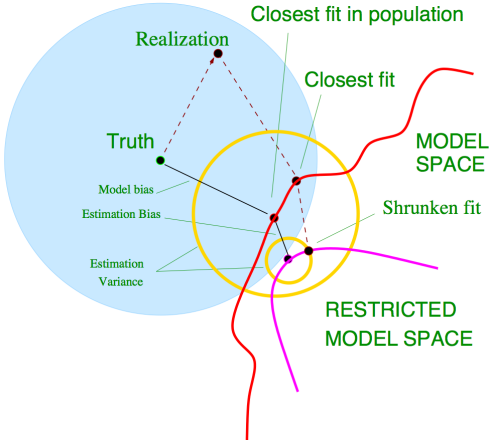
## Distributions

Gaussian: $\mathcal{N}(X|\mu, \sigma^2)$
$\Longrightarrow p(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$
Poisson: $\mathcal{P}(X|\lambda)$
$\Longrightarrow p(X = k) = \frac{\lambda^k}{k!}\exp(-\lambda)$

## Complexities

Grid search: $\mathcal{O}(M^D ND)$, where $M$ is the number of test points in one dimension.
Gradient descent: $\mathcal{O}(IND)$ where $I$ is the number of iterations we make.
Least-squares (normal equations): $\mathcal{O}(ND^2 + D^3)$

## Bias-Variance Decomposition




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

## Maximum Likelihood

The Likelihood Function maps the model parameters to the probability distribution of $\mathbf{y}$:
$\mathcal{L}_{lik}$ : parameter space $\to [0; 1] \quad \beta \mapsto \mathbf{p(y \mid \beta)}$ An underlying $p$ is assumed before. If the observed $y$ are IID, $p(\mathbf{y} \mid \beta) = \prod_{\mathbf{n}} \mathbf{p(y_n} \mid \beta)$.
$\mathcal{L}_{lik}$ 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 | $\kappa(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$ |

## 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:
$\mathcal{L}_{SVM}(\beta) = \sum_{\mathbf{n=1}}^{\mathbf{N}}[\mathbf{1} - \mathbf{y_n}\tilde{\phi}_{\mathbf{n}}\beta]_+ + \frac{\lambda}{\mathbf{2}}\sum_{\mathbf{j=1}}^{\mathbf{M}}\beta_{\mathbf{j}}^{\mathbf{2}}$
This is convex but not differentiable.