Gabriele Dragotto
gabriele.dragotto@polymtl.ca

FITTING, REG AND TRAINING

OVERFITTING +VARIANCE -BIAS } SHOULD BALANCE
UNDERFITTING +BIAS -VARIANCE

$$\mathbb{E}[\hat{y}-\tilde{f}(x)] = [\mathbb{E}(\tilde{f}(x)) - \tilde{f}(x)]^2 + \mathbb{E}(\tilde{f}(x)^2) - \mathbb{E}(\tilde{f}(x))^2 + \sigma^2$$

ACC $= \frac{\#\text{CORR CLASS}}{\#\text{SAMPLES}}$ // SENS/RECALL $= \frac{\#\text{TRUE POS}}{\#\text{POSITIVES}}$ $E_R = \frac{\#FP + \#FN}{\#\text{SAMPLES}}$

SPEC $= \frac{\#\text{TRUE NEG}}{\#\text{NEG}}$ // PRECISION $= \frac{\#\text{TRUE POS}}{\#\text{DECL POS}}$ $F_1 = \frac{2PR}{P+R}$

L1: $|w|$ - NON LINEAR - 0 SOME $+\lambda \cdot \text{SIGN}(w)$ L2: $\|w\|^2$ - SMOOTH $+2\lambda \bar{w}$

RM COND: 1. $\sum_0^{\infty} \alpha(k) = \infty$ 2. $\sum_0^{\infty} \alpha^2(k) < \infty$

LINEAR REGRESS.

$f_{\bar{w}_i} = \bar{X}\bar{w}_i$

$MSE = (\bar{y} - \bar{X}\bar{w})^T(y - \bar{X}w)$

EXACT $\frac{\partial Err(w)}{\partial w} = -2X^T(y - Xw)$ INV: mm^2

$= \bar{w} = (X^TX)^{-1}X^Ty$ MULT: m^3

GDS $\bar{w}_{k+1} = \bar{w}_k + 2\alpha_k X^T(y - Xw)$ $\Theta(mm^2 + m^3)$

GENERATIVE DISCRIMINATIVE

$P(y=1|\bar{x}) = \frac{P(\bar{x}|y=1)P(y=1)}{P(\bar{x})}$

// DISCRIMINATIVE LREG, KNN, NN
// GENERATIVE LDA, QDA, NB, HMM

LDA $m(m+m)$ SAME Σ FOR K CLASSES, m FEATURES
NBAYES $(K-1)+mK$ Σ DIFF BUT DIAGONAL
QDA $m(mK+m)$ Σ DIFF

LOGISTIC REGRESSION

LOG-ODDS $a = \sum_i \bar{w}_i x_i = \text{LOG}\, p/(1-p)$

ODDS $\sigma = b^a = b^{\sum w_i x_i}$

LIN. FN. $\theta(\bar{w}^T\bar{x}) = \sigma(a) = \frac{1}{1+e^{-w^Tx}} = 1/(1+e^{-a})$ APPROXIMATES THE BOUNDARY LINEARLY

LIKELIH. $L(D) = \prod^m \sigma(w^Tx)^{y_i} \cdot (1-\sigma(w^Tx))^{y_i}$ $P(y_i=1)$ GIVEN FEATURE

LOG(L(D)) $\text{LOG}L(D) = \sum_i y_i \cdot \text{LOG}(\sigma(w^Tx)) + (1-y_i)\text{LOG}(1-\sigma$ $H(d) = -\text{LOG}\,L(D)$

GDS $\frac{\partial H(\bar{w})}{\partial(\bar{w})} = -\sum_{i=1}^{m} \bar{x}_i(y_i - \sigma(\bar{w}^T\bar{x}_i))$ WITH $\bar{w}_{k+1} = \bar{w}_k + \alpha \frac{\partial H(\bar{w})}{\partial(\bar{w})}$ ONE $O(m$ FULL $O(m^3+m$

GAUSSIAN L $P(y_i|\bar{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^Tx_i)^2}{2\sigma^2}}$

LOG-L $\text{LOG}(L(D)) = \sum_i^m -\text{LOG}(\sqrt{2\pi\sigma^2}) - \frac{(y_i - w^Tx_i)^2}{2\sigma^2}$ $\frac{\partial\sigma(z)}{\partial z} = \sigma(z)(1-$

LDA, QDA $\text{COV} = \Sigma = \sum_k \sum_{j=1}^{m} \frac{(\bar{x}_k - \mu_k)(\bar{x}_i - \mu_k)^T}{N_0 + N_1 - 2} \to O(m^2)$ $\nabla\|x\|_2^2 = 2x$

LDA BOUNDARY: $\text{LOG}\left(\frac{P(\bar{x}|y=1)}{P(\bar{x}|y=0)}\right) = \underbrace{-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0}_{b} + \underbrace{x^T\Sigma^{-1}(\mu_1 - \mu_0)}_{\bar{w}^T\bar{x}}$

NAIVE BAYES

// FEATURES CONDITIONALLY INDEPENDENT //

$P(x_i|Y) = P(x_j|Y, x_i)\ \forall i,j \Rightarrow P(x|y) \propto \prod_i (x_i|y) \cdot P(y)$ $O(m)$ VS $O(2^m)$

$\theta_1 = P(y=1)$ ALWAYS ADDITIVE! SMOOTHING: $\theta_{j,i} = \frac{\#y_i, x_j + 1}{\#y_i + 2}$

$\theta_{j,1} = P(x_j=1, y=1)$ $\theta_{j,0} = P(x_j=1, y=0)$

$\text{LOG}(L(D))$ $\text{LOG}\frac{P(y=1|\bar{x})}{P(y=0|\bar{x})} = \text{LOG}\frac{P(y=1)}{P(y=0)} + \sum_{j=1}^{m} \frac{P(x_j|y=1)}{P(x_j|y=1)}$ $P(x_i) = P(x_i|y_1) + P(x_i|y_0)$

SWAP $P(x_i|K) = P(K) \cdot P(x_i|K)$

DTREES

$I(E) = \text{LOG}\, 1/P(E)$ $H(Y|X) = \sum_x P(X=x)H(Y|X=x)$

$H(S) = \sum_x P_x I(S_x) = -\sum_x P_x \text{LOG}P_x$ CLASS. MAX INFO GAIN
REGR MIN STD.DEV

FEATURES

$TF_i = \frac{\text{OCC. WORD}_i}{\text{WORD IN DOC}}$ PCA $\underset{W,V}{\text{ARGMIN}}\left(\sum_i^m \|X - XWV^T\|^2\right)$

$IDF_i = \text{LOG}\frac{\#\text{DOC}}{\#\text{DOC WITH}_i}$ $W^{m \times m}$ $m \times m$ FIRST m EIGENVECTOR BY EIGENVALUE ORTHO COLUMNS 1TH DIR WITH

SVM

PERCEPTON $h_{\bar{w}}(\bar{x}) = \text{SIGN}(\bar{w}^T\bar{x})$ — ALWAYS CONVERGES IFF LINEARLY SEPARABLE

$\text{ERR}(\bar{w}) = \sum_i \{0 \text{ IFF } y_i \bar{w}^T \bar{x}_i \geq 0;\ 1 \text{ ELSE}\}$

HARD-SVM $O(m^3)$

S-VECT $\bar{w}\bar{x} + b$

MARGIN $S = 2/\|\bar{w}\|$

$\text{MIN } \frac{1}{2}\|w\|^2$

S.T. $y_i \bar{w}^T x_i \geq 1 \quad \forall i = 1 \cdots m \ \alpha$

$d_{\bar{w},b}(x) = \frac{w^T x + b}{\sqrt{w_1^2 + w_2^2}}$

LAGRANGIAN $L(\bar{w}, \bar{\alpha}) = \frac{1}{2}\|\bar{w}\|^2 + \sum \alpha_i (1 - y_i(\bar{w}^T \bar{x}))$

$M = 2 d_{\bar{w}b}(x)$

$\frac{\partial L}{\partial \bar{w}} = \bar{w} - \sum_m \alpha_i \bar{x}_i y_i$ ← WEIGHTS!

$\text{XOR}_{x_1, x_2} = x_1 x_2 + \bar{x}_1 \bar{x}_2$

OUTPUT $h(\bar{w}) = \text{SIGN}(w^T \bar{x} + b)$

SOFT-SVM $L_{0-\infty} \to L_{0-1}$ HINGE LOSS $\mathcal{E} = L_{HIN}(\bar{w}^T \bar{x}, y_i) = \text{MAX}\{1 - \bar{w}^T x_i, 0)$

$\Rightarrow \underset{\mathcal{E}, \bar{w}}{\text{MIN}} \ C \cdot \sum_i \mathcal{E}_i + \frac{1}{2}\|w\|^2 \ \alpha$

$\alpha = 0$ GREAT! ☺

$\mathcal{E} = 0$ ON MARGIN

$\mathcal{E} \in (0,1)$ IN MARGIN

$\mathcal{E} = 1$ BOUNDARY

$\mathcal{E} > 1$ MISS

S.T. $y_i \bar{w}^T x_i \geq 1 - \mathcal{E}_i \quad \forall i = 1 \cdots m \ \beta$

$\mathcal{E}_i \geq 0 \quad \forall i$

$G_K = e^{\frac{-\|\bar{x} - \bar{z}\|^2}{2\sigma^2}}$

$S_K = \tanh(C_1 \bar{x} \bar{z} + C_2)$

KERNEL-T MAP INTO HIGHER DIMENTIONAL SPACE

ENSAMBLE

BOOTSTRAPPING K MODELS ON K SUB TRAINSET. −VAR +BIAS (DTREE, KNN)

RANDOM FOREST FOR (NODE) m RANDOM VAR BEST TEST

EXT. RANDOM. TR. FOR (TESTS) (m RANDOM AND RAN TEST) BEST TEST

BOOSTING INCREMENTALLY TRAIN −BIAS $L_W = \alpha_i = \frac{1}{2} \text{LOG}\left(\frac{1 - \mathcal{E}_i}{\mathcal{E}_i}\right)$

STACKING METACLASS ON TOP OF DIFFERENT APPROACHES

NEURALNET

FF-NN: PARAM = I * O + bias $\quad h_i(\bar{x}) = \sigma(\bar{w}_i x + b) \ \forall i$

2 HIDDEN LAYERS DO THE MAGIC!

$\sigma(x) = \frac{e^x}{1 + e^x} \quad \frac{\partial \sigma(\bar{x})}{\sigma(\bar{x})} = \sigma(\bar{x})(1 - \sigma(\bar{x}))$

$E_{OUT} = (\bar{y} - y)\bar{h}_{LAST} = \sigma_o \bar{h}_{LAST}$

$E_{HIDD} = \sigma_{h,j} \bar{x}$

SIGM, TANH SATURATE

RELU GOOD ☺

SOFTPLUS SMOTHER BUT COMP.

SOFTMAX MULTICLASS

MOMENTUM $M = \beta \Delta_{i-1} \bar{w} \Rightarrow \Delta_i \bar{w} = \frac{\alpha \partial}{\partial w} + M$

CNN

INVARIANT TO TRANSFORMATION. STRIDE = SPACING

CONV $(m \cdot m \cdot p \cdot k)$ FILTER $(m \times m)$

p INPUT k OUTPUT

RNN

TEMPORALITY + SHARED WEIGHTS

END OF SEQ SENTIMENT HIDDEN-S-REC ELMAN

END OF STEP LANGUAGE OUTPUTS-REC JORDAN

JORDAN $\bar{h}_T = (\bar{w} o_{t-1} + \bar{U} \bar{x}_t + \bar{b})$

ELMAN $\bar{h}_T = (\bar{w} h_{T-1} + \bar{U} \bar{x}_t + \bar{b})$

$\bar{o}_t = \phi(\bar{V} \bar{h}_T + \bar{c})$

BPTT WITH GRADIENT CLIPPING OR TRUNCATED

$\bar{W} = \bar{Q} \bar{D} \bar{Q}^T \Rightarrow h_T = \bar{Q} D^d \bar{Q}^T h_0$

LSTM

FIX VANISHING

HS-TRACK SELECT INFO FROM PAST X NEXT

CELL-STATE INFO FOR NEXT S

FORGET | GATE | AMOUNT TO KEEP

INPUT | | WHICH TO KEEP

OUTPUT | | WHAT TO BROADCAST

ENCODER CONTEXT VECTOR

DECODER +ATTENTION!

TEACHER FORCING FEED RIGHT ANSWER $y(t)$ AT $t+1$

M/VAR NAVE BAYES

$P(x_j | y = k) = \frac{1}{\sqrt{2\pi \sigma_{j,k}^2}} e^{-\frac{(x_j - \mu_{j,k})^2}{2\sigma_{j,k}^2}} \quad \text{LOG}(P(x_j | y = k)) \propto \sum_j^m - \frac{(x_j - \mu_{j,k})^2}{2\sigma_{j,k}^2} - \text{LOG}(\sigma_{j,k})$

$\text{LOG}\left(\frac{P(y=1|x)}{P(y=0|x)}\right) \propto \text{LOG}(P(y=1)) + \sum_{j=1}^m \left(-\frac{(x_j + \mu_{j,1})^2}{\sigma_{j,1}^2} - \text{LOG}(\sigma_{j,1})\right) + \text{LOG}(P(y=0)) + \sum_{j=1}^m \left(-\frac{(x_j + \mu_{j,0})}{\sigma_{j,0}^2} - \text{LOG}(\sigma_{j,0})\right)$

NB-NORMAL $P(y = k | \bar{x}_i) = \frac{P(y = k) \cdot \prod_i P(x_i = \%)}{P(x_i = \%)}$

// $P(x_i = \%) = P(x_i = \% | y = 1) + P(x_i = \% | y = 0)$

// $P(x_i = \% | y = k) = P(x_i = \% | y = k) \cdot P(y = k)$

ODDS: $\frac{P(y = 1 | x_i)}{P(y = 0 | x_i)}$

Canada Excellence Research Chair
Data Science for Real-Time Decision-Making

Applied Machine Learning

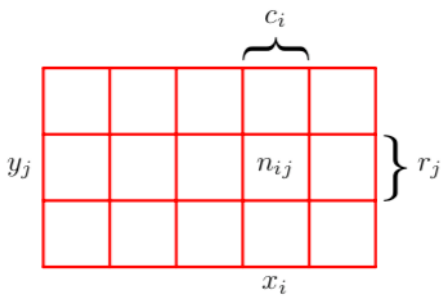
COMP551 @ McGill University

Gabriele Dragotto
gabriele.dragotto@polymtl.ca

Statistical tools

Definition 1. *The* ***joint probability*** *for* 2 *discrete random variables* X *and* Y *is defined as* $P(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$

Figure 1: Example of distribution for 2 variables. From Bishop (2006)



Definition 2. *The* ***conditional probability*** *of* $Y = y_j$ *given* $X = x_i$ *is* $P(Y = y_j|X = x_i) = \frac{n_{ij}}{c_i}$

Definition 3. *The* ***Bayes theorem*** *for* 2 *states that* $P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)}$

Definition 4. *The* ***sum rule*** *for* 2 *random variables* X *and* Y *states that* $p(X) = \sum_Y p(X, Y)$

Definition 5. *The* ***product rule*** *for* 2 *random variables* X *and* Y *states that* $p(X, Y) = p(Y|X) \cdot P(X)$ *where* $p(Y|X)$ *is the joint probability of* Y *given* X.

With the Bayes Theorem, the former definition implies Equation (1)

$$p(Y|X) = \frac{p(X|Y) \cdot p(Y)}{p(X)} \quad \text{and} \quad p(X) = \sum_Y p(X|Y) \cdot p(Y) \tag{1}$$

Definition 6. *The* ***Kolmogorov product rule*** *for* 2 *random variables* X *and* Y *states that* $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$.

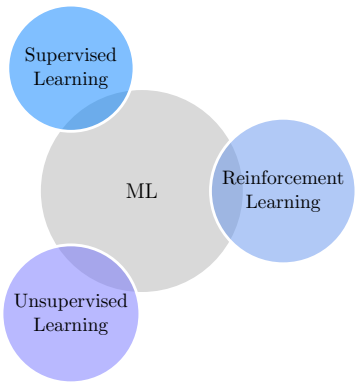
If the random variable is continue, the introduced concepts cha:

$$\int_{-\infty}^{+\infty} p(x)dx = 1 \quad p(x) \geq 0 \; \forall x \tag{2}$$

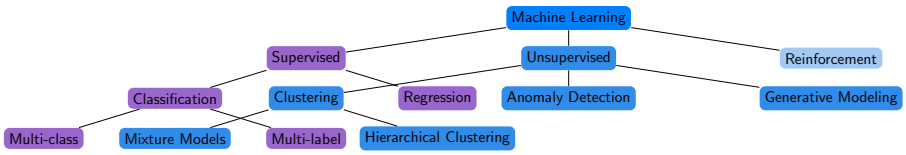
$$P(x \leq z) = \int_{-\infty}^{z} p(x)dx \quad P'(x) = p(x) \tag{3}$$

1 Introduction

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, **improves** with experience E
>
> Class Notes by Hamilton (2019)



In a broader scope, *AI* can be seen as the field of Computer Science aiming to automate actions, decision and learning process typical of human beings. A sub field of *AI* is **machine learning**, which implements in machines the capability of extracting knowledge from data (Goodfellow et al., 2016). This approach relies heavily on the representation of the data it recieves (eg, regression only analyses some features). The subfield of representation learning concerns with an approach capable of guessing the right representation to use in order to produce the output.
Finally, deep learning aims to build lower level representation of data, and hence builds up complex representations from simpler ones. This former approach



1.1 Fitting, regularization and validation

Definition 7. *A **training set** is a formatted set of data. Each column is named variable, feature or attribute and each row constitutes a training example or instance. The desired outcome is named target or output variable.*

Definition 8. *An **overfitted** model is a statistical model that contains more parameters than can be justified by the data. The model has a **lower true error** on a different hypotesys than the chosen one, hence high variance/low bias.*

Counter with: model comparison, cross-validation, regularization, early stopping, pruning, Bayesian priors, or dropout. Reduce variance at the cost of some bias.

Definition 9. ***Underfitting** occurs when a statistical model or machine learning algorithm cannot adequately capture the underlying structure of the data. The model has then low variance/high bias.*

1.2 Errors

Assuming a machine learning algorithm predicts $y = f(x) + \epsilon$ where $\epsilon \sim (\mu = 0, \sigma^2)$, then the error can be decomposed in *bias* and *variance*:

$$\text{Error} \quad \mathbb{E}[y - \tilde{f}(\bar{x})] = (Bias[\tilde{f}(\bar{x})])^2 + Var[\tilde{f}(\bar{x})] + \sigma^2 \tag{4}$$

$$\text{Bias} \quad \mathbb{E}[\tilde{f}(\bar{x})] - \tilde{f}(\bar{x}) \tag{5}$$

$$\text{Variance} \quad \mathbb{E}[\tilde{f}(\bar{x})^2] - \mathbb{E}[\tilde{f}(\bar{x})]^2 \tag{6}$$

$$\tag{7}$$

Definition 10. ***Bias** occurs when a statistical model or machine learning algorithm cannot capture relations between input and output variables. underfitting.*

Definition 11. ***Variance** models the sensitivity between small changes in the input. overfitting.*

$$\text{Accuracy} \quad A = \frac{\#CorrectlyClassified}{\#Samples} \tag{8}$$

$$\text{Sensitivity/Recall} \quad R = \frac{\#TruePositives}{\#Positives} \tag{9}$$

$$\text{Specificity} \quad S_c = \frac{\#TrueNegatives}{\#Negatives} \tag{10}$$

$$\text{Precision} \quad P = \frac{\#TruePositives}{\#DeclaredPositives} \tag{11}$$

$$\text{Error rate} \quad E_r = \frac{\#FalseNegatives + \#FalsePositives}{\#Samples} \tag{12}$$

$$F_1 \text{ score} \quad F_1 = 2\frac{P \cdot R}{P + R} \tag{13}$$

TruePositives	FalseNegatives
FalsePositives	TrueNegatives

Table 1: The confusion matrix structure

1.2.1 Cross Validation

Definition 12. ***k-fold*** *cross validation splits the training-set in k partitions, training on $k-1$ and verifying on the remaining one. It increases the computational time by a factor of k.*

Definition 13. ***Leave-out-one*** *eliminates one row on the training set, and evaluates the error on it. The error estimation averages on all the iterations.*

1.3 Regularization

Definition 14. ***Lasso penalization*** *- or L1 regularization - add to the loss-function a penalty term proportional to the absolute value of estimated weights. non linear . Sets some coefficients to 0.*

Definition 15. ***Ridge regression*** *- or L2 regularization - add to the loss-function a penalty term proportional to the square of estimated weights. Smooths some coefficients.*

2 Linear Regression

The *I.I.D* assumption states the following:

Definition 16. *A set of random variables is independent and identically distributed if each random variable has the* ***same probability distribution*** *as the others and all are* ***mutually independent****.*

The following equations are the fundamental:

$$\text{Function} \quad f_{\bar{w}_i}(\bar{X}) = \bar{X}\bar{w}_i \tag{14}$$

$$\text{Fn. MSE Error} \quad f^*_{\bar{w}}(\bar{X}) = \arg\min_i \left[(\bar{y} - \bar{X}\bar{w}_i)^T(\bar{y} - \bar{X}\bar{w}_i)\right] \tag{15}$$

2.1 Exact approach

We minimize the gradient of the MSE representation.

$$\text{Gradient MSE} \quad \frac{\delta Err(\bar{w})}{\delta \bar{w}} = -2X^T(y - X\bar{w}) \tag{16}$$

$$\Rightarrow \quad X^T(y - X\bar{w}) = 0 \tag{17}$$

$$\Rightarrow \quad X^TY = X^TX\bar{w} \tag{18}$$

$$\Rightarrow \quad \bar{\mathbf{w}} = (\mathbf{X^TX})^{-1}\mathbf{X^TY} \tag{19}$$

Operation		**Cost**
Matrix Inversion	1	nm^2
Matrix multiplications	3	m^3
Total	**4**	$\Theta(nm^2 + m^3)$

Some issues:

- Numerical stability
- Singular X: features are dependent (no full column rank).
  fix? *combine, apply functions, interaction terms, etc...*

2.2 Gradient descent

Weights are updated step by step depending on the MSE gradient representation. Assuming $Err(\bar{w}_0) > Err(\bar{w}_1) > ...Err(\bar{w}_{step})$, until $|\bar{w}_{k+1} -$

$\bar{w}_k| > \epsilon$, then:

$$\text{Updated Weights} \quad \bar{w}_{k+1} = \bar{w}_k - \alpha_k \frac{\delta Err(\bar{w}_k)}{\delta \bar{w}_k} \tag{20}$$

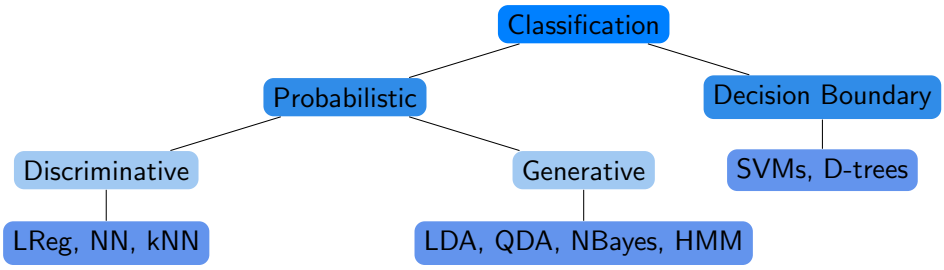
$$\Rightarrow \quad \bar{w}_{k+1} = \bar{w}_k + 2\alpha_k X^T(y - X\bar{w}) \tag{21}$$

Where the function (parameter) α_k is the learning rate at the $k-th$ step.

- Large α: gradient might **not converge** $\Rightarrow \alpha_k \to 0$ if $k \to 0$.
- Small α: might loop in **local minima** $\Rightarrow \alpha_k \to 0$ if $k \to 0$.

Claim 1. *Robbins-Monroe conditions are sufficient to ensure convergence of the $\bar{w}_k$ to a local minimum of the error function.*
RMC: $\sum_{k=0}^{\infty} \alpha_k = \infty$ *and* $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$

For instance, $\alpha(k) = 1/(k+1)$ satisfies the RMC.



$$P(y=1|\bar{x}) = \frac{P(\bar{x}|y=1)P(y=1)}{P(\bar{x})}$$

Estimating how likely are the features given the output, and the independent probability of the output class.

- **Discriminative learning**: $P(y|\bar{x})$ models the boundary between classes.
- **Generative learning**: $P(\bar{x}|y)$ models the distribution of each class.

Model	**Cost**	**Assumption**
LDA	$m(m+n)$	Same covariance for k classes, and m features
Naive Bayes	$(k-1)+mk$	Different covariances but diagonal matrix
QDA	$m(mk+n)$	Different covariances for k classes

3 Logistic Regression

Definition 17. *The **logit** - or log-odd ratio - is defined as the logarithm of the odds* $1/(1-p)$*. Therefore* $logit(p) = \log \frac{p}{1-p}$

The following equations are the fundamental:

$$\text{Log-odds} \quad a = \sum_i w_i x_i = \log \frac{p}{1-p} \tag{22}$$

$$\text{Odds} \quad o = b^a = b^{\sum_i w_i x_i} \tag{23}$$

$$\text{Lin. Logistic Fn.} \quad \sigma(\bar{w}^T \bar{x}) = \sigma(a) = \frac{1}{1+e^{-\bar{w}^T \bar{x}}} = \frac{1}{1+e^{-a}} \tag{24}$$

Where the decision boundary approximates **the log-odds with a linear function of the features** $\bar{w}^T \bar{x}$.
$\sigma(\bar{w}^T \bar{x})$ is the proability of $y_i = 1$ given the $\bar{x}$ input vector. The goal is to maximize the likelihood, without incurring in numerical instability.

$$\text{Likelihood} \quad L(D) = \prod_{i=1}^{n} \sigma(\bar{w}^T \bar{x})^{y_i} \cdot (1 - \sigma(\bar{w}^T \bar{x}))^{1-y_i} \tag{25}$$

$$\text{log-Likelihood} \quad \log L(D) = \sum_{i=1}^{n} y_i \log(\sigma(\bar{w}^T \bar{x})) + (1-y_i) \log(1-\sigma(\bar{w}^T \bar{x})) \tag{26}$$

$$\text{Cross Entropy Loss} \quad H(d) = -\log L(D) \tag{27}$$

Therefore, maximising likelihood corresponds to minimizing the cross entropy. It is easy to note that the inner term of $L(D)$:

1. $= \sigma(\bar{w}^T \bar{x})$ iff $y_i = 1$
2. $= 1 - \sigma(\bar{w}^T \bar{x})$ iff $y_i = 0$

3.1 Gradient descent

We minimize the gradient of the cross-entropy loss representation. One step is $O(nm)$, vs $O(m^3 + nm^2)$ for exact solution.

$$\text{Gradient CE} \quad \frac{\delta H(\bar{w})}{\delta \bar{w}} \tag{28}$$

$$\text{with} \quad \frac{\delta \log(\sigma)}{\delta \bar{w}} = \frac{1}{\sigma}, \quad \frac{\delta \sigma}{\delta \bar{w}} = \sigma(1-\sigma), \frac{\delta \bar{w}^T x}{\delta \bar{w}} = x, \tag{29}$$

$$\frac{\delta(1-\sigma)}{\delta \bar{w}} = -\sigma(1-\sigma) \tag{30}$$

$$\Rightarrow \quad \frac{\delta \mathbf{H}(\bar{\mathbf{w}})}{\delta \bar{\mathbf{w}}} = -\sum_{\mathbf{i=1}}^{\mathbf{n}} \bar{\mathbf{x}}_\mathbf{i}(\mathbf{y_i} - \sigma(\bar{\mathbf{w}}^\mathbf{T}\bar{\mathbf{x}}_\mathbf{i})) \tag{31}$$

Therefore, the update rule

$$\text{Updated Weights} \quad \bar{w}_{k+1} = \bar{w}_k + \alpha_k(-\frac{\delta H(\bar{w})}{\delta \bar{w}}) \tag{32}$$

3.2 Probabilistic interpretation

The logistic regression can be interpreted through a gaussian distribution of the likelihood.

$$\text{Gaussian Likelihood} \quad P(y_i|\bar{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \bar{w}^T \bar{x}_i)^2}{2\sigma^2}} \tag{33}$$

$$\text{Gaussian log-ikelihood} \quad \log L(D) = \sum_{i=1}^{n} -\log(\sqrt{2\pi\sigma^2}) - \frac{(y_i - \bar{w}^T \bar{x}_i)^2}{2\sigma^2} \tag{34}$$

$$\text{Squared loss} \quad (y_i - \bar{w}^T \bar{x}_i)^2 \tag{35}$$

Therefore, maximising log-likelihood corresponds to minimizing the squared loss. Train models using theoretically grounded loss functions but evaluate using interpretable measures.

4 LDA and QDA

LDA approaches the problem by assuming that the conditional probability density functions $p(\bar{x}|y_i = \%)$ are **gaussian** functions with mean μ_i and same covariance Σ, while QDA assumes different covarances. The Bayes

optimal predicts points as being from the second class if the log-odd ratios is **greater than 0**.

$$P(\bar{x}|y) \quad \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{(\bar{x}-\mu)^T(\bar{x}-\mu)}{2\Sigma}} \tag{36}$$

$$\text{Covariance} \quad \Sigma = \sum_{k=0}^{1}\sum_{j=1}^{n} \frac{(\bar{x}_i-\mu_k)^T(\bar{x}_i-\mu_k)}{N_0+N_1-2} \tag{37}$$

QDA has **more parameters** to estimate, but greater flexibility to estimate the target function. Estimating Σ is expensive: $O(m^2)$!
The LDA has a linear boundary, namely $w_0 + \bar{x}^T\bar{w}$.

$$\log \frac{P(\bar{x}|y=1)}{P(\bar{x}|y=0)} = \tag{38}$$

$$\log \frac{P(y=1)}{P(x=1)} - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 + x^T\Sigma^{-1}(\mu_1-\mu_0) \tag{39}$$

4.1 Naive Bayes

Definition 18. *The strong (naive) assumption states assumes that the features are* ***conditionally*** *independent given the output y. Therefore:*
$P(x_j|y) = P(x_j|y, x_k) \quad \forall\, j,k \quad \Rightarrow P(x|y) = \prod_i (x_i|y)$

A Naive Bayes classifier with **binary** features has to estimate just $O(m)$ parameters compared to $O(2^m)$. Useful when the number of features is high: linear time parameters and exact form optimization. Hence, no correlation between features is taken into account.

$$\Theta_1 \quad P(y=1) \tag{40}$$

$$\Theta_{j,1} \quad P(x_j=1|y=1) \tag{41}$$

$$\Theta_{j,0} \quad P(x_j=1|y=0) \tag{42}$$

$$L(\Theta_1|y) \quad \Theta_1^y(1-\Theta_1)^{1-y} \tag{43}$$

$$\text{log-Likelihood} \quad log_L(D) = \log \frac{P(y=1|\bar{x})}{P(y=0|\bar{x})} = \tag{44}$$

$$\Rightarrow \quad \log \frac{P(y=1)}{P(y=0)} + \log \frac{\prod_{j=1}^m P(x_j|y=1)}{\prod_{j=1}^m P(x_j|y=0)} \tag{45}$$

$$\Rightarrow \quad \log \frac{P(y=1)}{P(y=0)} + \sum_{j=1}^{m} \log \frac{P(x_j|y=1)}{P(x_j|y=0)} \tag{46}$$

Therefore, **each feature contributes independently** to the classification. In order to reduce variance, one can exploit additive smoothing.

Definition 19. *The **additive smoothing** - or Laplace smoothing - modifies estimators as:* $\hat{\theta}_i = \frac{x_i+\alpha}{N+\alpha d} \quad \forall i$ *where* $d = p_i^{-1}$*, namely the probability of class* y_i*.*

For instance, the add one smoothing would be: $\Theta_{j,i} = \frac{\#y_i=1 \wedge x_j=1+1}{\#y_i=1+2}$.

Gaussian Naive Bayes assumes Σ is distinct between classes and diagonal, and $P(x|y)$ is assumed to be a multivariate Gaussian.

5 Decision Trees

Divide the space of features in a detailed way. Learning a tree means learning tests (with categorical/binary outcomes) over each branch. Usually: grow and prune.

- Easy to represent: Boolean functions
- Hard to represent: Parity function $O(2^m)$, Majority function

$$\text{Information} \quad I(E) = \log_2 \frac{1}{P(E)} \tag{47}$$

$$\text{Entropy} \quad H(S) = \sum_i p_i I(s_i) = -\sum_i p_i \log p_i \tag{48}$$

$$\text{Conditional Entropy} \quad H(y|x) = \sum_i p(x=i) H(y|x=i) \tag{49}$$

Claim 2. *The entropy can be interpreted as: average amount of information per symbol, uncertainity before the outcome, number of bits for the symbol.*

- Classification: maximize information gain.
- Regression: minimize standard deviation.

In order to avoid overfitting:

- post-pruning: prune the tree once it is fully grown.
  For each node: prune iff **improves validation accuracy**. Replace decision with majority rule.
- early-stopping: stop the training when no information gain.

6 Feature Design

6.1 NL Features

Some language processing standard measures:

$$\text{tf} \quad tf_i = \frac{word_i}{\#words} \quad \text{in document } d_k \tag{50}$$

$$\text{idf} \quad idf_i = \log \frac{\#documents}{\#documents \text{ with } i} \quad \text{in corpus} \tag{51}$$

$$\text{tf-idf} \quad tf - idf_i = tf_i \cdot idf_i \tag{52}$$

6.2 PCA

Principal component analysis - or truncated SVD - projects into a lower dimensional space a given set of features. Assume the original dimension as $\mathbb{R}^m$ and the final one $\mathbb{R}^n$ with $n < m$.

$$\text{PCA-problem} \quad \arg\min_{W,U}(\sum_i^n ||X - XWU^T||^2) \tag{53}$$

Where:

- $W^{m \times n}$: compression matrix with the first n eigenvectors sorted by eigenvalues. Columns are orthogonal, and $i - th$ column is the $i - th$ direction with $i - th$ maximal variance.
- $U^{n \times m}$: decompression matrix.

7 Instance learning

Compute a **domain specific** *distance* metric between points in the training set and the ones being tested. The decision boundaries are given by the Voronoi diagram ran on the training data. lazy learning, wait for the query to generalize.

With the $k - NN$ the prediction is the majority/mean of k-nearest points. Try to use *gaussian* distances.

8 Support Vector Machines

8.1 Percepton

Basic linear classifier. Its error estimates how much $w^t x$ is far away from being correct. The training data is linearly separable iff there is no training error.

$$\text{Percepton} \quad h_w(\bar{x}) = sign(\bar{w}^T \bar{x}) \tag{54}$$

$$\text{Percepton Err} \quad Err(\bar{w}) = \sum_i \{0 \iff y_i \bar{w}^T x_i \geq 0; \text{ else } 1\} \tag{55}$$

$$\tag{56}$$

Theorem 8.1. *If the training data is linearly separable, the percepton will converge to* 0 *error in a finite number of steps.* ♡ *(Bishop, 2006)*

8.2 SVM

Definition 20. *A linear SVM is a perceptron with a vector $\bar{w}$ so that the margin is maximized.*

The SVM builds the separating hyperplane between two classes of the input space. $\vec{w} \cdot \vec{x} - b = 0$ is the separation hyperplane.

$$\text{Separation} \quad S = \frac{2}{||\bar{w}||} \tag{57}$$

The optimization model, that can be trained at most in $O(n^3)$.

$$\text{minimize}_{\bar{w}} \quad \frac{1}{2}||\bar{w}||^2 \tag{58}$$

$$\text{subject to} \quad y_i \bar{w}^T \bar{x}_i \geq 1 \quad \forall i = 1, .., n \tag{59}$$

$$||\bar{w}|| = \frac{1}{M} \tag{60}$$

The lagrangian multipliers $\bar{\alpha}$ of the dual are the support vectors. The weight vector is a linear combination of the support vectors.

Definition 21. *A support vector is a point laying on the decision boundary.*

$$\text{Lagrangian Form.} \quad L(\bar{w}, \bar{\alpha}) = \frac{1}{2}||\bar{w}||^2 + \sum_i \alpha_i(1 - y_i(\bar{w}^T\bar{x})) \tag{61}$$

$$\text{Derivate} \quad \frac{\delta L(\bar{w}, \bar{\alpha})}{\delta \bar{w}} = \bar{w} - \sum_i \alpha_i \bar{x}_i y_i \tag{62}$$

$$\text{Weights} \quad \bar{w} = \sum_i \alpha_i \bar{x}_i y_i \tag{63}$$

$$\text{Margin} \quad M = M(\bar{\alpha}_i) = \frac{\bar{w}^T\bar{\alpha}_i}{||\bar{w}||} = \frac{1}{||\bar{w}||} \tag{64}$$

$$\text{Output} \quad h_{\bar{w}}(\bar{x}) = sign(\sum_i^n \alpha_i y_i(x_i \cdot x)) = sign(\bar{w}^T\bar{x} + b) \tag{65}$$

The output of the classifier is given by the dotproduct of the sample x with the support vectors x_i.

8.3 Non-linearly separable class

If the training set is not linearly separable, either:

- **Soften** the constraint: $L_{0-\infty} \to L_{0-1}$.
- **Kernel-trick**: use non-linear kernels.

8.3.1 Soft SVM with Hinge loss

Approximate the missclassification penalty with a linear function .

$$\text{Hinge Loss} \quad \xi = L_{hin}(\bar{w}^T\bar{x}_i, y_i) = \max\{1 - \bar{w}^T\bar{x}_i, 0\} \tag{66}$$

Therefore optimize the following:

$$\text{minimize}_{\bar{w},\xi} \quad C\sum_i \xi_i + \frac{1}{2}||\bar{w}||^2 \tag{67}$$

$$\text{subject to} \quad y_i\bar{w}^T\bar{x}_i \geq 1 - \xi_i \quad \forall i = 1, .., n \quad (\textbf{Lagr} : \bar{\alpha}) \tag{68}$$

$$\xi_i \geq 0 \quad \forall i = 1, .., n \quad (\textbf{Lagr} : \bar{\beta}) \tag{69}$$

With $C \to \infty$ it gives an hard-SVM.

- $\alpha = 0$: points outside margin.

- $\xi = 0$: points on/outside the margin.
- $\xi \in (0, 1)$: points within the margin (on with $\alpha_i > 0$).
- $\xi = 1$: points on decision line.
- $\xi > 1$: missclassified.

8.3.2 Kernel machines

The method allows to compute dot-products **without explicitly computing coordinates** in the new feature space. Optimizing with the dual requires just the computation of the kernel function.

$$\text{Gaussian Kernel} \quad K = (\bar{x}, \bar{z}) = e^{\frac{-||\bar{x}-\bar{z}||^2}{2\sigma^2}} \tag{70}$$

$$\text{Sigmoid Kernel} \quad K = (\bar{x}, \bar{z}) = \tanh(c_1 \bar{x} \cdot \bar{z} + c_2) \tag{71}$$

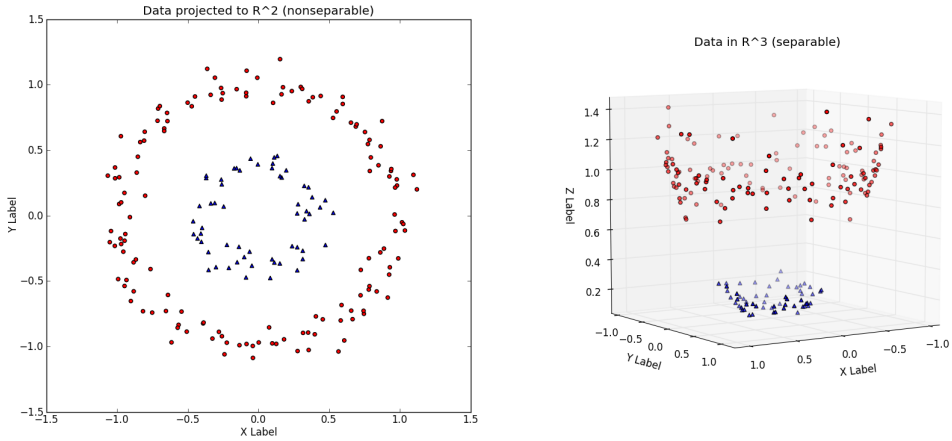


Figure 2: Kernel mapping $\mathbb{R}^2 \to \mathbb{R}^3$

9 Ensamble methods

- **Bootstrapping-Bagging**: K different models trained on *different sub training sets* (eg, sampling with replacement).
  Reduce variance and increase bias: **DTree, kNN**.

- **Boosting**: K different models incrementally trained.
  Reduce bias: **AdaBoost**.Empirically, low chance of overfitting.
  misclassified are dangerous: can be weighted too much.

- **Stacking**: different classifiers combined with a meta-classifier.
  Features: classifiers output.

9.1 Bagging

1 Random Forest

```
Train K different trees with k bootstraps.
for each node do
    Pick m random variables
    Determine test=max{InfoGain}
end for
Predict using ensamble
```

Each tree has high variance, but the ensemble uses averaging, which reduces variance.

2 Extremely randomized trees

```
Train K different trees with k bootstraps.
while desired depth do
    for number of tests do
        Pick m random variables
        Determine a random test
    end for
    Select test=max{InfoGain}
end while
Predict using ensamble
```

The smaller m is, the *more randomized* the trees are.

9.2 Boosting

Iterate training with focus on misclassified instances in the previous epoch.

With *AdaBoost*, the learner has a weight defined with:

3 Boosting

```
while error under threshold do
  for classifier do
    Train with more focus on misclassified instances
  end for
end while
Predict using ensamble
```

$$\text{Learner weight} \quad \alpha_i = \frac{1}{2}\log(\frac{1-\epsilon_i}{\epsilon_i}) \tag{72}$$

10 Neural Network

Generally can suffer from overtraining ($overfitting$) occurs when weights take on large magnitudes. Train with $P_{aram} = I \cdot O + b$ with respectively $Input$, $Output$ sizes and $bias$.

- **Feed-forward**: output of layer j is input of $j+1$.
  Fully connected: all units in j are input of all units in $j+1$.

$$\text{Sigmoid} \quad \sigma(x) = \frac{e^x}{1+e^x} \tag{73}$$

$$\text{Sigmoid derivate} \quad \frac{\delta\sigma(x)}{\delta x} = \sigma(x)(1-\sigma(x)) \tag{74}$$

$$\text{Hidden unit} \quad h_i = \sigma(\bar{w}^T\bar{x} + b) \quad \forall i \tag{75}$$

4 FF-NN

```
h̄^0 = 0
for i = 1, ...H do
  h̄^i = σ(W̄^i h̄^{i-1} + b̄^i)
end for
y_out = φ(W̄^out h̄^H + b̄^out)
```

$$\text{Error} \quad J = \frac{1}{2}(\tilde{y} - y)^2 \tag{76}$$

$$\text{Error derivate (out)} \quad \frac{\partial J}{\partial \bar{w}_{out}} = \frac{\partial J}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial \bar{w}_{out}} = (\tilde{y} - y)\bar{h}_{last} = \delta_o \bar{h}_{last} \tag{77}$$

$$\text{Error derivate (hidden)} \quad \frac{\partial J}{\partial \bar{w}_j} = \frac{\partial J}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial \bar{w}_j} = \delta_o \frac{\partial \tilde{y}}{\partial \bar{w}_j} = \delta_o \frac{\partial \tilde{y}}{\partial \bar{h}_j} \frac{\partial \bar{h}_j}{\partial \bar{w}_j} \tag{78}$$

$$\Rightarrow \quad \delta_o \bar{w}_{o,j} \sigma(\bar{w}^T \bar{x} + \bar{b})(1 - \sigma(\bar{w}^T \bar{x} + \bar{b}))\bar{x} \tag{79}$$

$$\Rightarrow \quad \delta_{h,j} \bar{x} \tag{80}$$

5 Stochastic GDS

while no convergence **do**
  Pick a training $\bar{x}$
  1 Feed $\bar{x}$ and get y
  2 Compute $\frac{\partial J}{\partial \bar{w}_{out}}$
  for Hidden unit i **do**
    3 Compute share of correction $\frac{\partial J}{\partial \bar{w}_j}$
  end for
  4 Update weights
    Hidden: $\bar{w}_j = \bar{w}_j - \alpha \frac{\partial J}{\partial \bar{w}_i}$
    Output: $\bar{w}_{out} = \bar{w}_{out} - \alpha \frac{\partial J}{\partial \bar{w}_{out}}$
end while

Claim 3. *Any function can be approximated to arbitrary accuracy by a network with 2 hidden layers.*

10.1 Activation function

Should be easy differentiable and **non-linear**.

- **Sigmoid, tanH**: easily saturate.
- **reLU**: strong empirical results.
- **softplus**: smoother than reLU but harder to train.
- **softmax**: generalizes multiclass tasks.

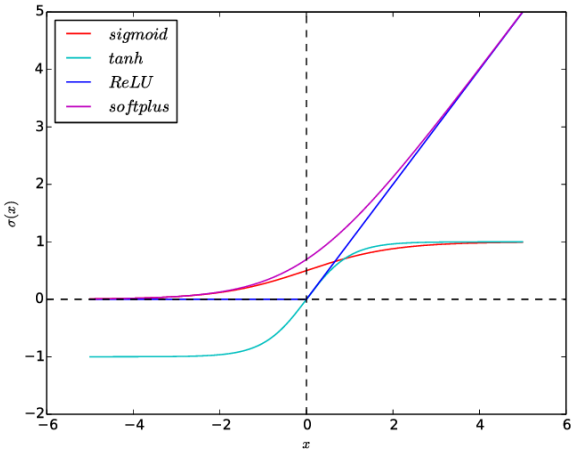


Figure 3: Most common activation functions

10.2 Backpropagation

Reverse-mode automatic differentiation ($RV - AD$) can efficiently compute the derivative of every node in a computation graph. Very sensible to learning rates. Empirically implemented in AdamOpt.

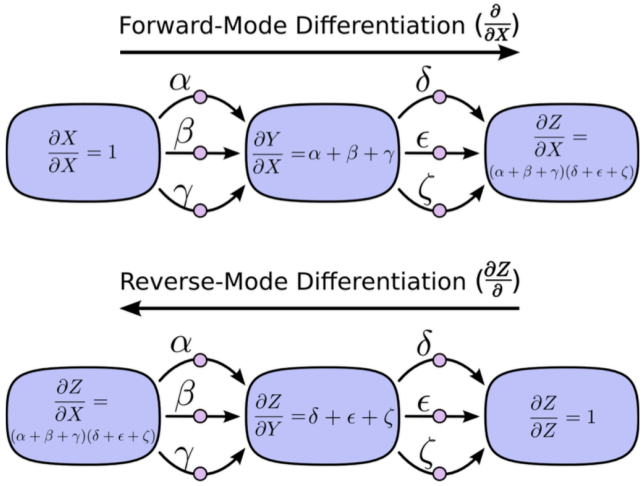


Figure 4: Differentiation paradigms from Hamilton (2019)

$$\text{Momentum} \quad M = \beta \Delta_{i-1} \bar{w} \tag{81}$$

$$\text{Momentum Update} \quad \Delta_i \bar{w} = \alpha \frac{\partial J}{\partial \bar{w}} + M \tag{82}$$

With momentum:

- **Pros**:
  - Avoid **small** local-minima.
  - Keep $\bar{w}$ **moving** when error is flat.
- **Cons**:
  - Avoid **global minima**.
  - It's **an additional** parameter.

11 Convonutional Neural Network

> Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.
>
> (Wikipedia, 2019)

Each layer transforms an input 3D **tensor** to an output 3D tensor using a differentiable function.

- Very high-dimensional inputs
- Multiple layers of inputs
- Invariance: light, rotations, translations

Main working scheme:

- Local receptive field for each first hidden unit (all channels), and shared parameters for each receptive field. For each conv: $(n \cdot m \cdot l)k$ parameters, with $n \times m$ filter, k output, and l input sizes.
- Pooling: aggregates result of convonutional layers.

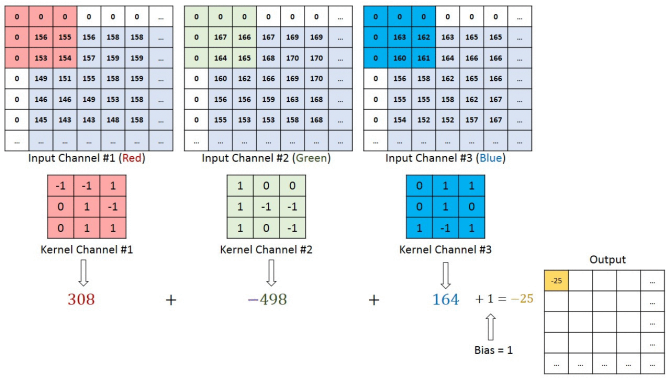


Figure 5: Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel. From TowardsDataScience (2019)

- Stride: spacing between receptive fields.

Definition 22. *The **dropout** independently sets each hidden unit activity to zero with probability p*

Definition 23. *The **Batch Normalization** normalizes the input layer by adjusting and scaling the activations.*

12 Recurrent Neural Network

RNN is a class of artificial neural network where connections between nodes form a directed graph along a **temporal sequence**. Weights are **shared** over time-steps. Output can be:

- End of the sequence: for instance, sentiment classification.
- End of time-step: for instance, generative language.

Recurrence can be based on:

- Hidden states: Elman RNN $\bar{h}_t = (\bar{W}\bar{h}_{t-1} + \bar{U}\bar{x}_t + \bar{b})$
- Outputs: Jordan RNN $\bar{h}_t = (\bar{W}\bar{o}_{t-1} + \bar{U}\bar{x}_t + \bar{b})$

$$\text{Output} \quad \bar{o}_t = \phi(\bar{V}\bar{h}_t + \bar{c}) \tag{83}$$

Network are trained with backpropagation through time ($BPTT$). Sometimes the gradient is **truncated** for large sequences (see Figure 7).

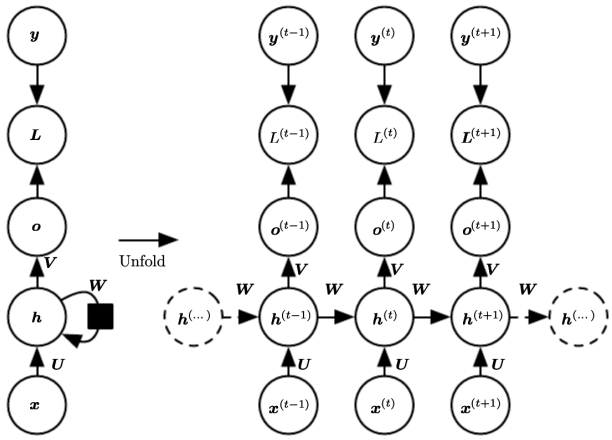


Figure 6: *RNN* scheme from Goodfellow et al. (2016)

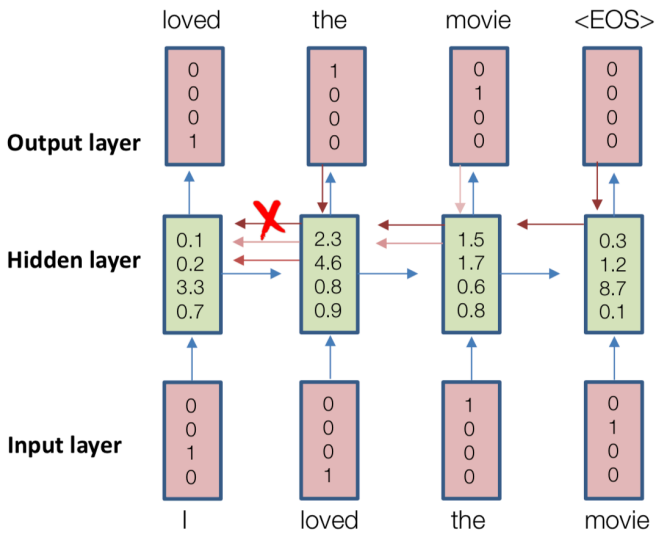


Figure 7: *RNN BPTT* from Hamilton (2019)

With **long term dependencies**: usually exploding/vanishing gradient: $\bar{h}_t = \bar{W}\bar{h}_{t-1}$ with $\bar{W} = \bar{Q}\bar{D}\bar{Q}^T$, and therefore $\bar{h}_t = \bar{Q}\bar{\mathbf{D}}^{\mathbf{d}}\bar{Q}^T\bar{h}_0$. Hence, use Gradient Clipping.

12.1 LSTM Units

Fix the **vanishing** grandient issue. Each cell is composed of **gates**, sigmoid layers to control information flow.

- Hidden state tracking: selects info from past for the *next* prediction.
- Cell State: selects info for future **predictions**.

Types of gates:

- Forget gate: amount of information to keep from previous cell.
- Input gate: what input is kept from previous cell.
- Output gate: what info from the cell is needed to predict.

12.2 Encoder-decoder RNN

Get different size output by **encoding** the input with a RNN - context vector-, and the output with a different RNN. The context vector can be enhanced with attention: namely, the decoder weights part of the context vector.

Definition 24. *The **teacher forcing** is a training procedure in which the model receives the ground truth output $y(t)$ as input at time $t+1$. (Goodfellow et al., 2016)*

References

Bishop, C. M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep learning. Vol. 1. MIT press Cambridge.

Hamilton, W., 2019. Comp551 lectures. University Lecture.

TowardsDataScience, 2019. Towardsdatascience. Online.

Wikipedia, 2019. Wikipedia.org. The Online Encyclopedia.