

Intro into Machine Learning

ML in Physics. Linear Models. Numerical optimization. Logistic Regression.
Figures of Merits. Overfitting. Model Selection. Regularization.

Fourth Machine Learning in High Energy Physics Summer School,
MLHEP 2018, August 6–12

Alexey Artemov^{1,2}

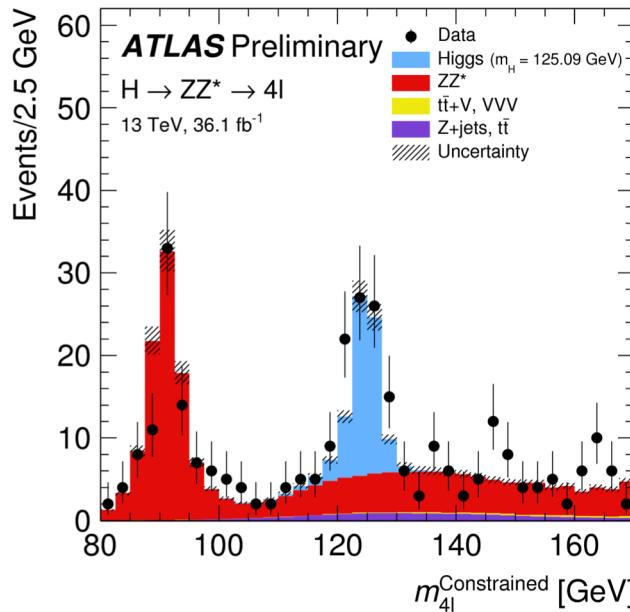
¹ Skoltech ² National Research University Higher School of Economics

Lecture overview

- › Machine learning in high energy physics
- › Linear models for regression
- › Numerical and stochastic optimization at a glance
- › Linear models for classification
- › Figures of merits
- › Overfitting: how to fool the linear regression
- › Regularization
- › Why machine learning works: a bit of learning theory
- › Bonus track

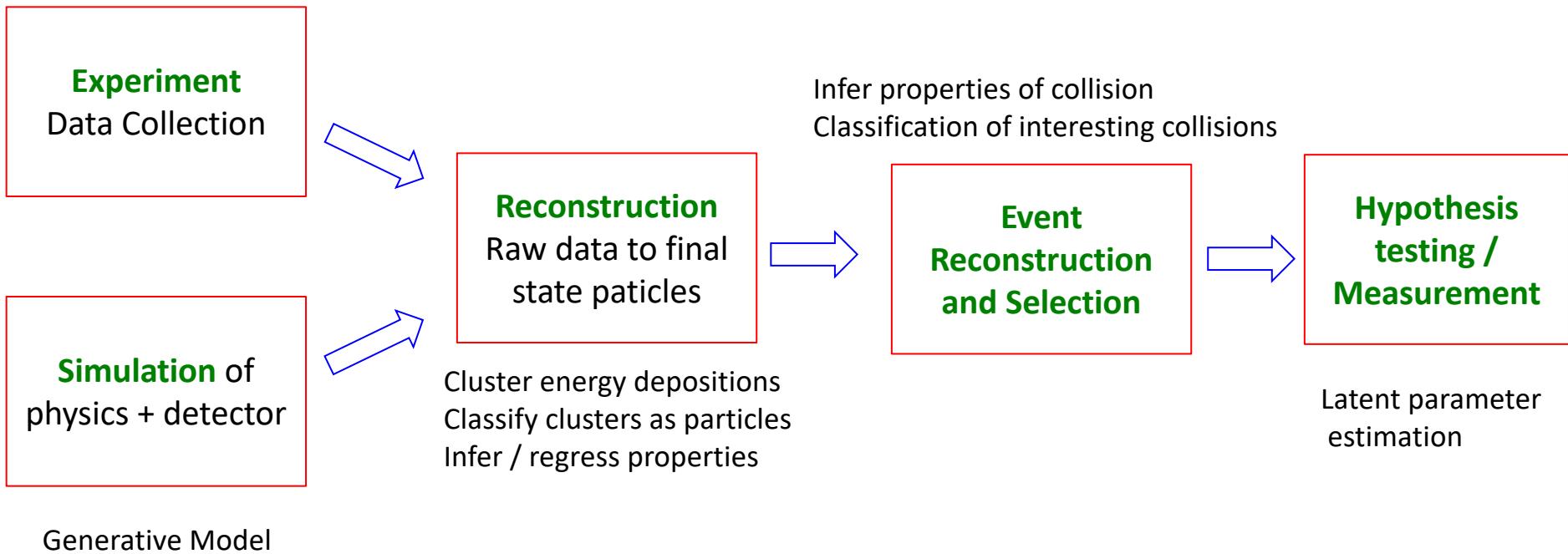
Machine learning in high
energy physics

Studying Collisions



- With a collection of collisions we can perform:
 - Hypothesis testing: new particle present?
 - Measurement: Inference of latent parameters, e.g. Higgs mass
- Extremely accurate simulations + knowledge of the data generating process (i.e. physics) to analyze our data!

Pipeline

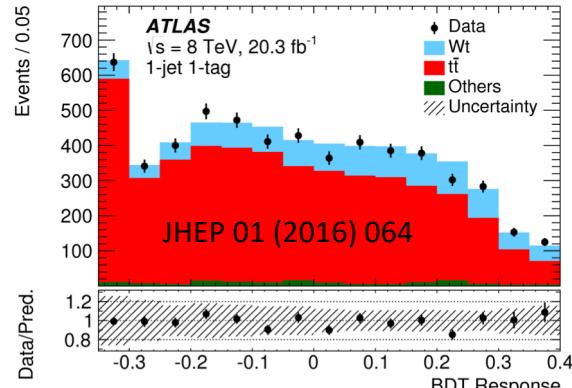


- Build on our knowledge of how the data is created
 - Use our simulation to design and study reconstruction algorithms, and to compare predictions with our experimental data
- Use Machine learning to improve (or rethink) the steps of this process?

Machine Learning Applied Widely in HEP

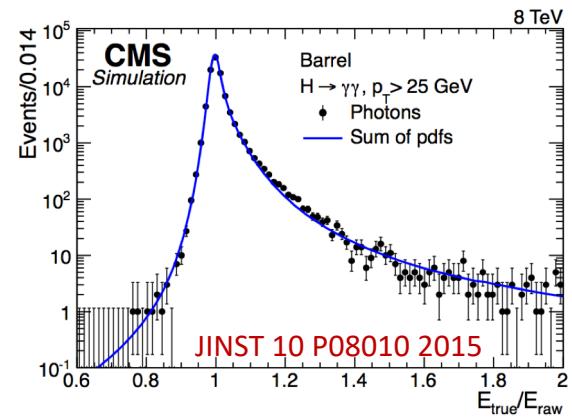
- **In analysis:**

- Classifying signal from background, especially in complex final states
- Reconstructing heavy particles and improving the energy / mass resolution



- **In reconstruction:**

- Improving detector level inputs to reconstruction
- Particle identification tasks
- Energy / direction calibration



- **In the trigger:**

- Quickly identifying complex final states

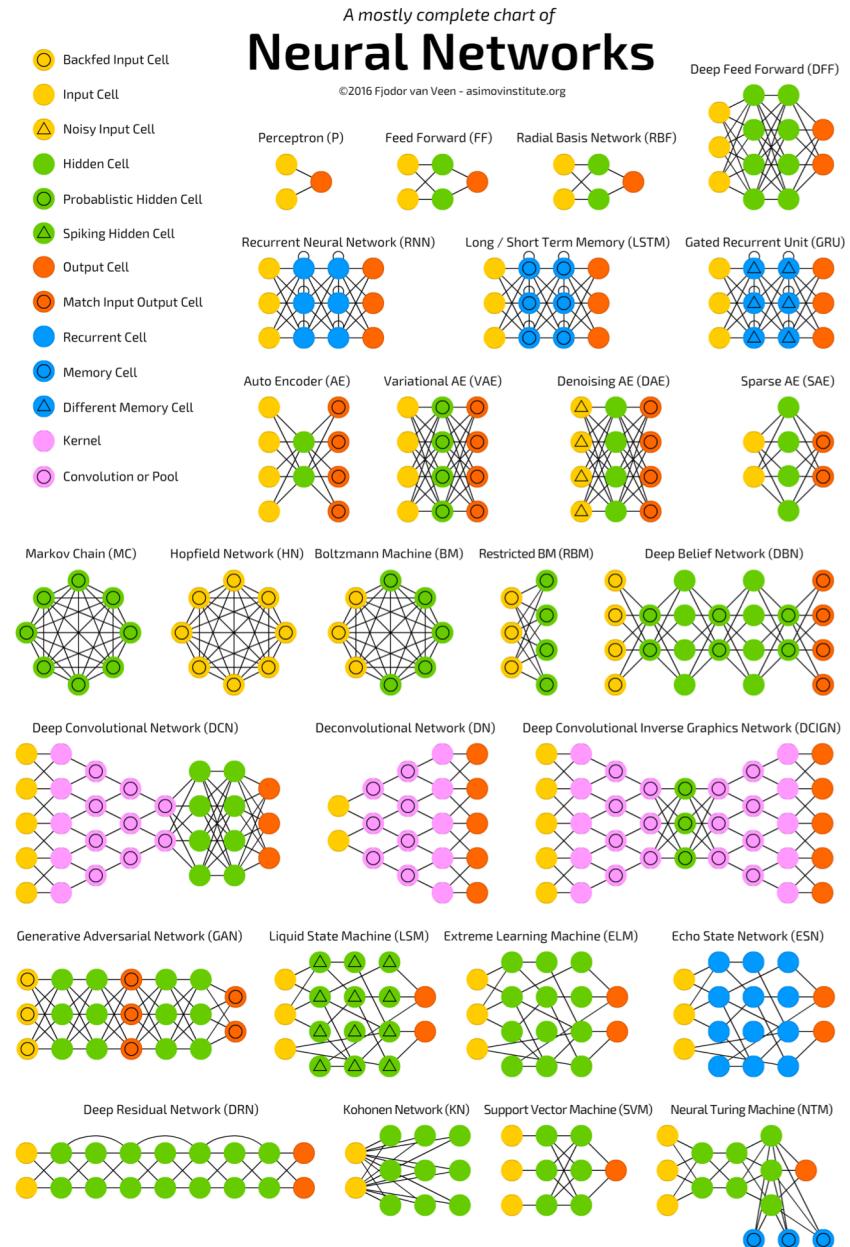
- **In computing:**

- Estimating dataset popularity, and determining needed number and location of dataset replicas

		Z/γ* → ττ				
		Tau Particle Flow				
		Diagonal fraction: 74.7%				
Reconstructed decay mode		3h [±] ≥ 1π ⁰	2.5	3.6	5.3	56.6
3h [±]		0.2	0.6	0.3	92.5	40.2
h [±] ≥ 2π ⁰		0.4	6.0	35.4	0.1	0.4
h [±] π ⁰		9.4	74.8	56.3	0.9	2.5
h [±]		89.7	16.0	4.3	1.2	0.3

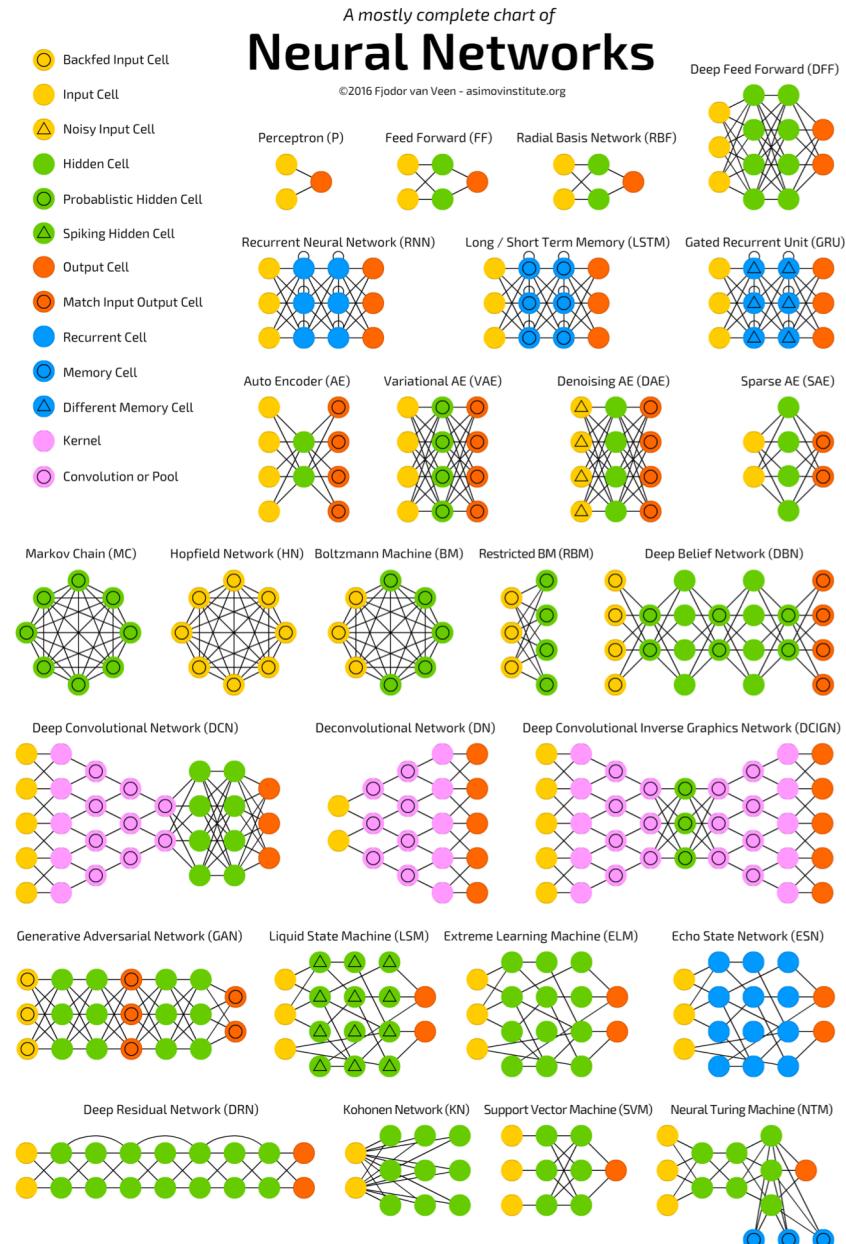
Deep Learning for HEP

- Moving **inductive bias** from feature engineering to machine learning (neural network) model design
 - Inductive bias \sim knowledge about the problem
 - Feature engineering \sim hand crafted variables
 - Model design \sim the data representation and the structure of the machine learning model / network



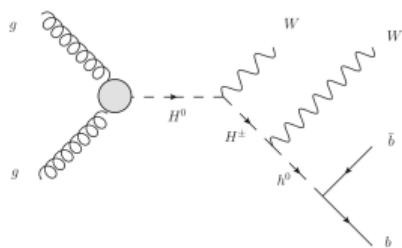
Deep Learning for HEP

- How do we move deep learning advancements into HEP?
 - Translate problems in HEP into problems in ML domain
 - Incorporate HEP domain knowledge: physical motivation for designing data representation, architectures, or new methods if needed
 - How do we extract what is learned?

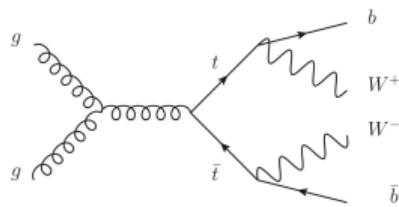


Search for the exotic particles in HEP

- > Starred in the HIGGS benchmark
- > The goal: classify signal vs. background processes



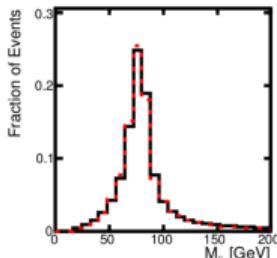
(a)



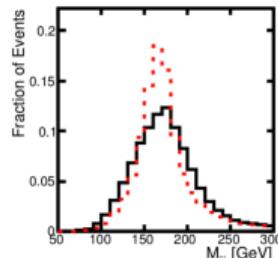
(b)

- > Features: jets 1 through 4 p_T s, lepton p_T , missing transverse momentum, etc. (28 features in total)
- > Dataset size: 11 mln events (<http://archive.ics.uci.edu/ml/datasets/HIGGS>)

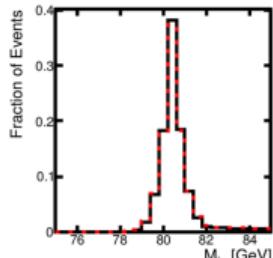
Alexey Artemov



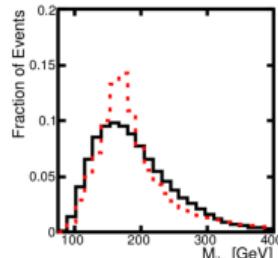
(a)



(b)



(c)



(d)

Expert knowledge and physical models VS machine learned models

- › Would one want to apply machine learning for . . .
 - › . . . Newton's mechanics?
 - › . . . suggesting music to radio listeners?
 - › . . . sorting integers?
 - › . . . controlling steel production?
 - › . . . sorting strawberries?
- › Some criteria for machine learning to be applied in a dependency recovery setting:
 - › Little prior knowledge of the dependency exists
 - › The dependency has a complex form too hard for manual examination
 - › A sample from the dependency of sufficiently large size is available

Machine learning as function approximation

- › An unknown distribution D generates **instances** $(\mathbf{x}_1, \mathbf{x}_2, \dots)$ independently
- › An unknown function $f : \mathbb{X} \rightarrow \mathbb{Y}$ generates **responses** (y_1, y_2, \dots) for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \dots$
- › The machine learning problem: choose a plausible **hypothesis** $h : \mathbb{X} \rightarrow \mathbb{Y}$ from the **hypothesis space** \mathbb{H}
- › The error of a hypothesis h is the deviation from the true f measured by the **loss function** (an example for regression):

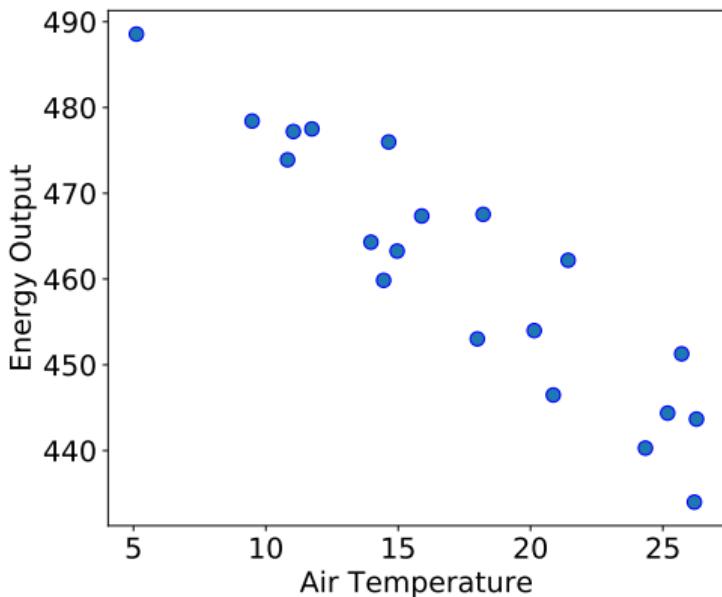
$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (f(\mathbf{x}_i) - h(\mathbf{x}_i))^2$$

- › **Learning:** the search for the optimal hypothesis $h \in \mathbb{H}$ w.r.t. the fixed loss function

Linear models for regression

Univariate linear regression

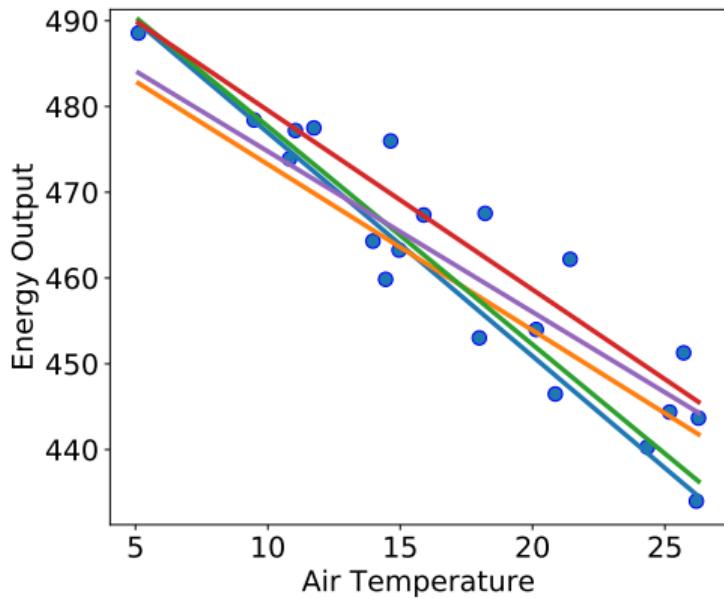
- › A single feature (**regressor**) x :
Air Temperature
- › A single **dependent variable** y :
Energy Output
- › Training set $X^\ell = \{(x_i, y_i)\}_{i=1}^{20}$
- › The regression model:
$$y_i = h(x_i; \mathbf{w}) + \varepsilon_i$$
- › Linear model: $y_i = w_1 x_i + w_0 + \varepsilon_i$
- › **The goal:** given X^ℓ , find $\mathbf{w} = (w_1, w_0)$



Univariate linear regression

- › Which fit to choose?
- › With the linear model being fixed, depends on the data and the loss function!
- › Mean square (L2) loss (MSE):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2$$



Some other evaluation metrics for regression

- › Mean square (L2) loss (MSE): $\text{MSE}(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2$
- › Root MSE: $\text{RMSE}(h, X^\ell) = \sqrt{\frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2}$
- › Coefficient of determination: $R^2(h, X^\ell) = 1 - \frac{\sum_{i=1}^{\ell} (y_i - h(x_i))^2}{\sum_{i=1}^{\ell} (y_i - \mu_y)^2}$
with $\mu_y = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$
- › Mean absolute error: $\text{MAE}(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} |y_i - h(x_i)|$

Univariate linear regression

- With the loss fixed, the linear problem reduces to optimization:

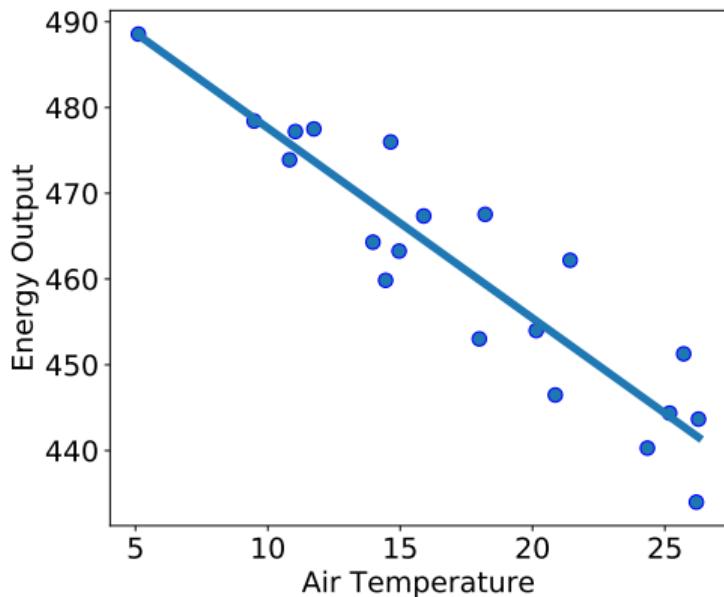
$$\frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - w_1 x_i - w_0)^2 \rightarrow \min_{(w_0, w_1) \in \mathbb{R}^2},$$

to which an analytical solution is available

$$\hat{w}_1 = \frac{\sum_{i=1}^{\ell} (x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{\ell} (x_i - \mu_x)^2},$$

$$\hat{w}_0 = \mu_y - \hat{w}_1 \mu_x$$

$$\text{with } \mu_x = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i, \quad \mu_y = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$$



Multivariate linear regression

- › Multiple features (regressors) $\mathbf{x}_i = (x_{1i}, \dots, x_{di})$ available for each y_i
- › The model:

$$y_1 = w_1 x_{11} + \dots + w_d x_{d1} + \varepsilon_1,$$

$$y_2 = w_1 x_{12} + \dots + w_d x_{d2} + \varepsilon_2,$$

...

$$y_\ell = w_1 x_{1\ell} + \dots + w_d x_{d\ell} + \varepsilon_\ell,$$

is often written in matrix-vector form as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_\ell \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{d1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1\ell} & x_{2\ell} & \dots & x_{d\ell} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_\ell \end{bmatrix} \quad \longleftrightarrow \quad \mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}$$

Multivariate linear regression: the solution

- › The problem: minimize MSE

$$Q(h, X^l) = \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^d w_k x_{ki} \right)^2 \equiv \| \mathbf{y} - \mathbf{Xw} \|^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › Solve analytically via computing the gradient

$$\nabla_{\mathbf{w}} \| \mathbf{y} - \mathbf{Xw} \|^2 = 2(\mathbf{y} - \mathbf{Xw})\mathbf{X} = 0$$

- › The solution

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Numerical and stochastic
optimization at a glance

A quick intro into the Numerical Optimization

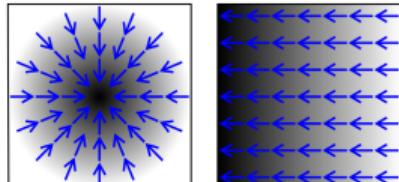
- Consider the **optimization problem** in \mathbb{R}^d

$$f(\mathbf{x}) \rightarrow \min_{\mathbf{x} \in \mathbb{R}^d}$$

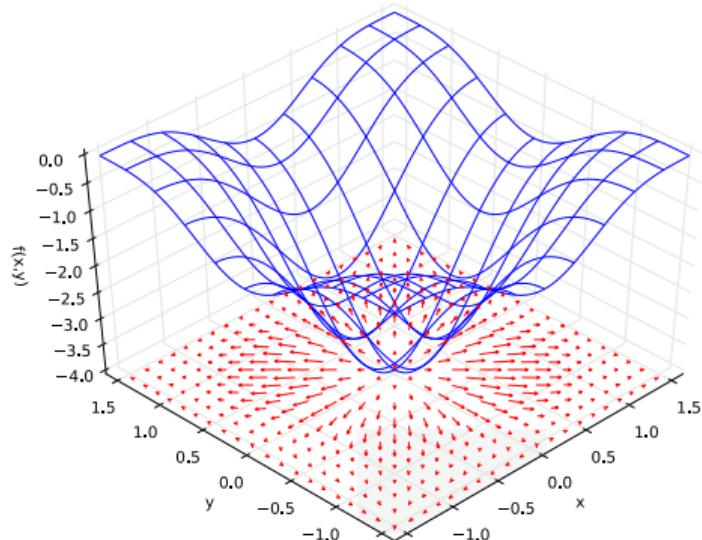
(such as $f(\mathbf{w}) \equiv \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^d w_k x_{ki} \right)^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$)

- In general, solved using the numerical methods such as the **gradient descent**
- Gradients:** directions in \mathbb{R}^d pointing towards steepest function increase

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \equiv \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)$$



Alexey Artemov



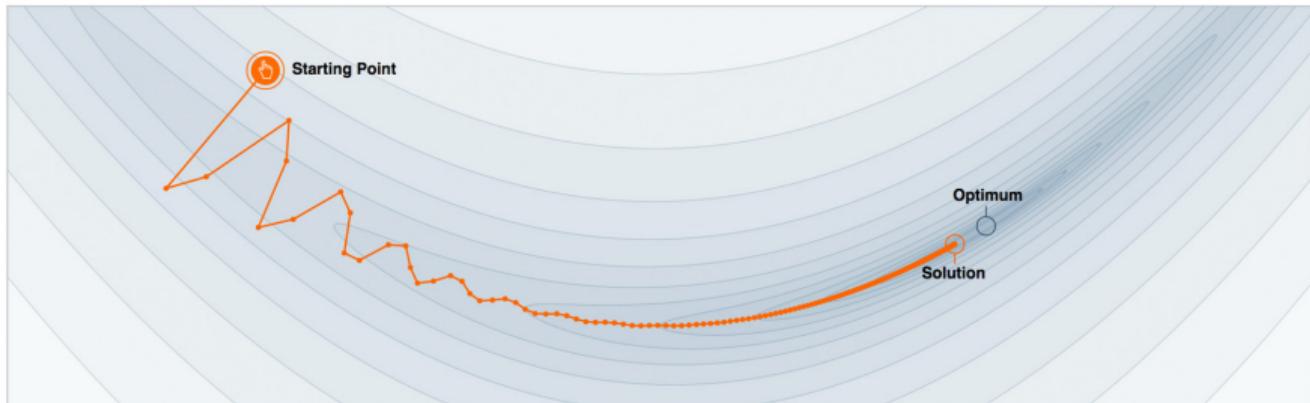
The gradient descent algorithm

- › The gradient descent procedure iterates from $\mathbf{x}^{(0)}$ as

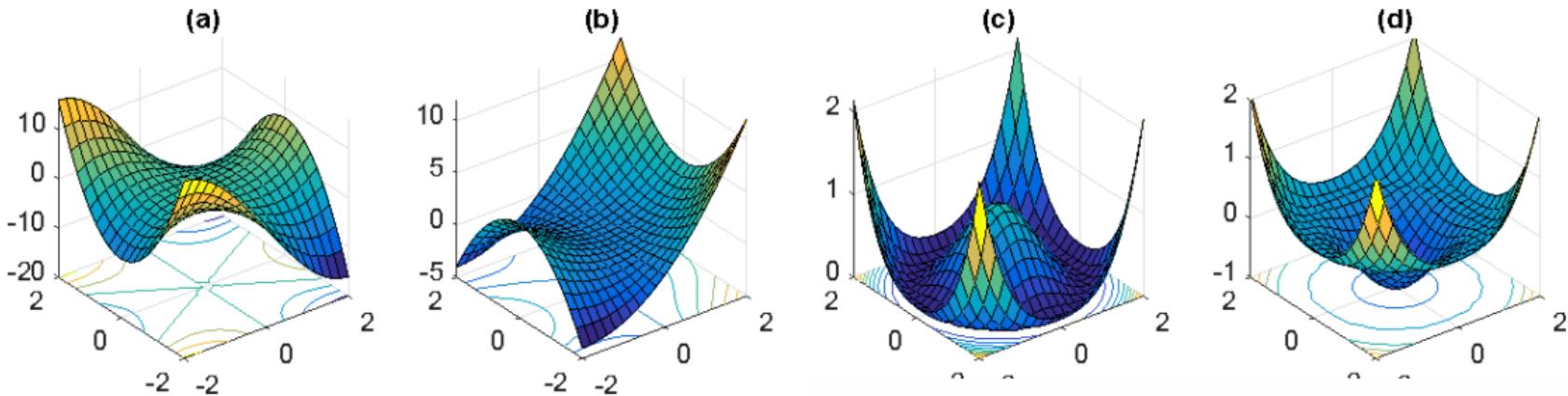
$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \alpha_k \nabla_{\mathbf{x}} f(\mathbf{x}^{(k-1)})$$

with α_k controlling the k th step size

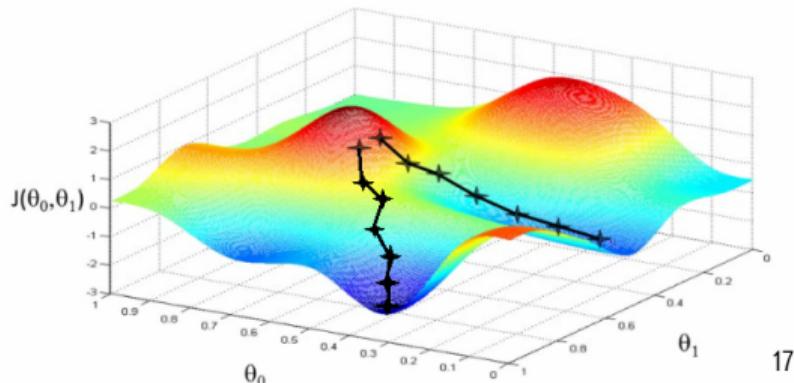
- › For smooth **convex** functions with a single minimum \mathbf{x}^* , k steps of gradient descent achieve accuracy $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) = \mathcal{O}(1/k)$



The gradient descent with non-convex targets



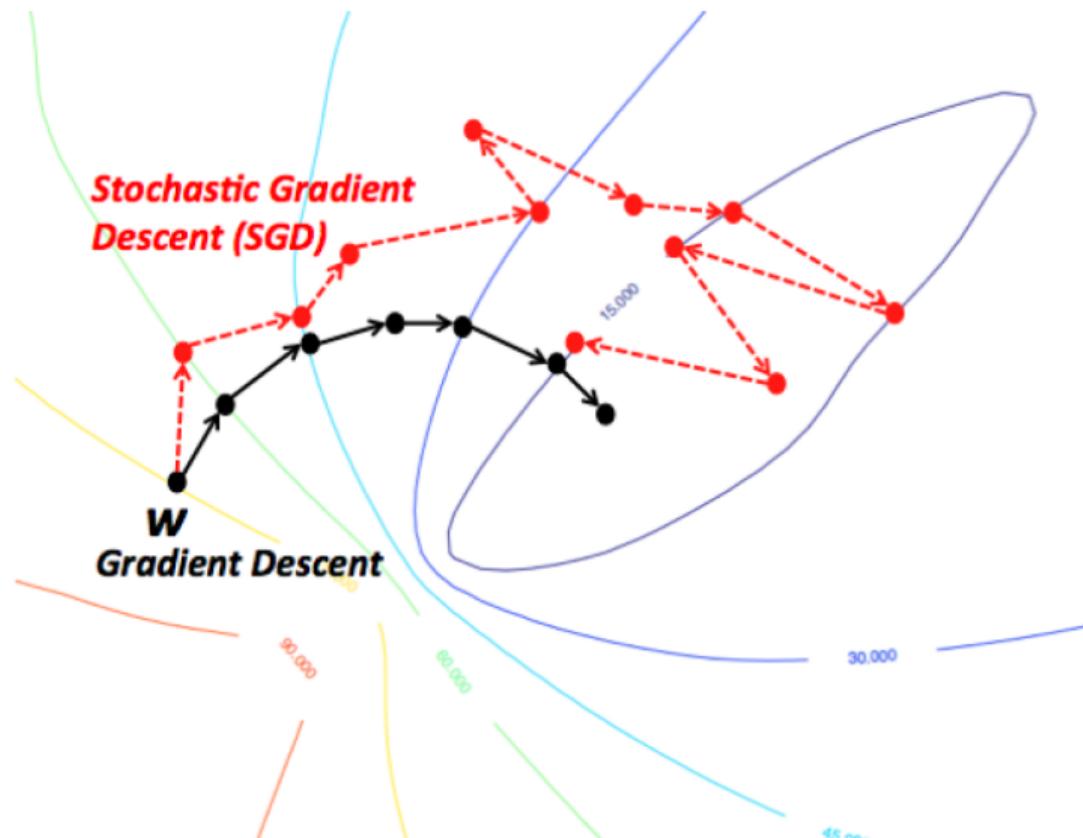
- › Trajectories of gradient descent over **non-convex functions** may (and will) not always end up in a single optimum



The stochastic gradient descent algorithm

- › Machine learning: many additive targets $f(\mathbf{w}) = \sum_{i=1}^{\ell} f_i(\mathbf{w})$,
computationally inefficient for large ℓ
- › Use subsamples for gradient estimation: the Stochastic Gradient Descent (SGD)
 1. Pick $i_k \in \{1, \dots, \ell\}$ at random;
 2. Compute $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$
- › For smooth convex functions with a single minimum \mathbf{x}^* , k steps of SGD achieve accuracy $f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*) = \mathcal{O}(1/\sqrt{k})$
- › Batching, variance reduction, momentum hacks available to improve the convergence rate to $\mathcal{O}(1/k)$

Gradient descent VS stochastic gradient descent



An example: SGD for multivariate linear regression

- › Initialize with some $\mathbf{w}^{(0)}$
- › Gradient in i_k th object is

$$\nabla_{\mathbf{x}} f_{i_k}(\mathbf{w}) = 2(y_{i_k} - \mathbf{x}_{i_k}^\top \mathbf{w}) \mathbf{x}_{i_k} \quad (\in \mathbb{R}^d)$$

- › Compute updates using SGD:

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$$

Linear models for classification

Linear models for classification

- › An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \dots)$
- › An unknown function $f : \mathbb{X} \rightarrow \mathbb{Y}$ generates labels (y_1, y_2, \dots) for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- › The classification problem: choose a plausible hypothesis (**classifier**) $h : \mathbb{X} \rightarrow \mathbb{Y}$ from the **hypothesis space** \mathbb{H}
- › The error of the classifier h is the probability (over D) that it will fail

$$Q(h, D) = \Pr_{\mathbf{x} \sim D}[f(\mathbf{x}) \neq h(\mathbf{x})]$$

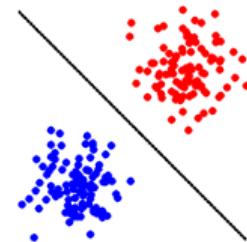
usually estimated by the **accuracy metric**

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$$

Linear models for classification

- › Linear model: $h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^d w_i x_i + w_0\right) = \text{sign}(\mathbf{w}^\top \mathbf{x} + w_0)$
- › The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq h(\mathbf{x}_i)] \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$



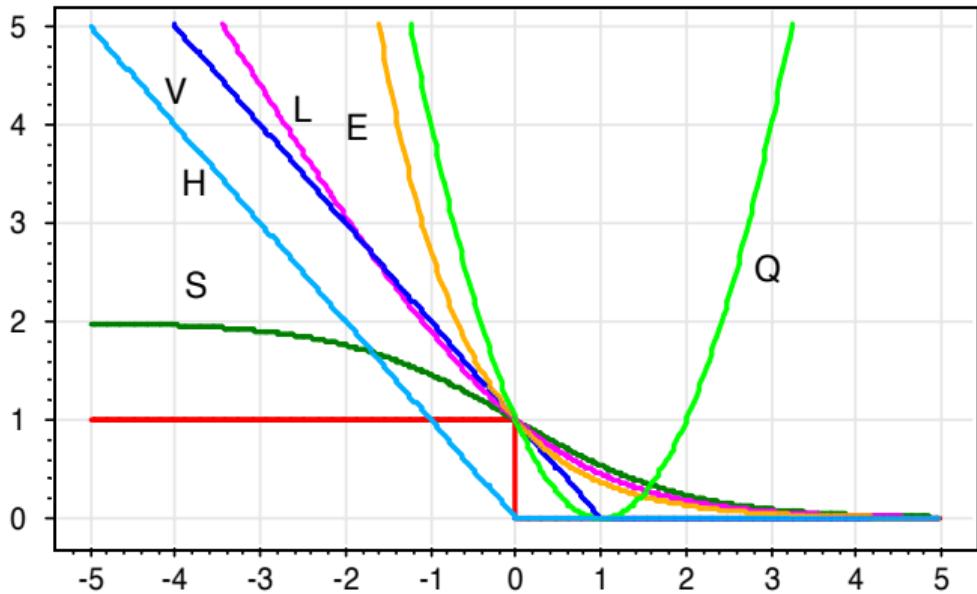
- (cannot optimize using gradient descent)
- › The solution: optimize a differentiable upper bound for $Q(h, X^\ell)$!
 - › $Q(h, X^\ell)$ can be written using $Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(M_i)$
 - where $L(M_i) = [M_i < 0] \equiv [y_i \mathbf{w}^\top \mathbf{x}_i < 0]$
 - › Upper-bounding $L(M)$ yields upper bounds for $Q(h, X^\ell)$

Linear models for classification: upper bounds

Multiple approximations to accuracy

- › $L_L(M) = \log(1 + e^{-M})$
- › $L_H(M) = \max(0, 1 - M)$
- › $L_P(M) = \max(0, -M)$
- › $L_E(M) = e^{-M}$
- › $L_S(M) = 2/(1 + e^M)$

and their respective optimization procedures give rise to various learning algorithms



The logistic regression model

- › Training set $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$ where $y_i \in \{-1, +1\}$
- › We seek an algorithm h such that $h(\mathbf{x}) = P(y = +1|\mathbf{x})$ (**model probability!**)
- › A probability that an instance (\mathbf{x}_i, y_i) is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

- › Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

is often written via **log-likelihood** (of which the negative is **log-loss**)

$$\log L(X^\ell) = \sum_{i=1}^{\ell} [y_i = +1] \log h(\mathbf{x}_i) + [y_i = -1] \log(1 - h(\mathbf{x}_i))$$

The logistic regression model

- › The choice of h : sigmoid function

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

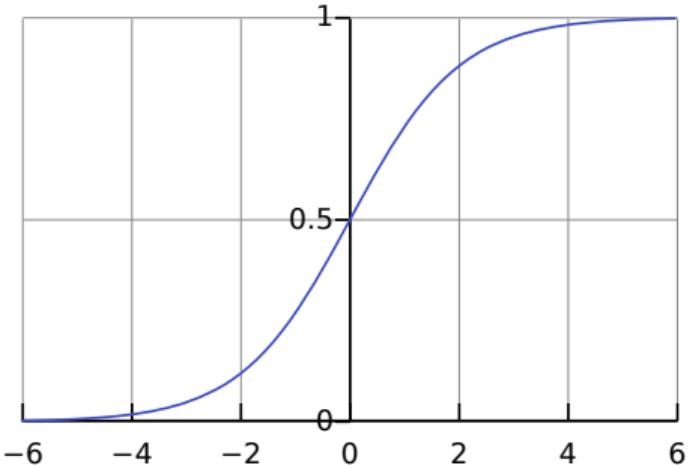
where $\sigma(x) \in [0, 1]$

- › Typical choice: the logistic function

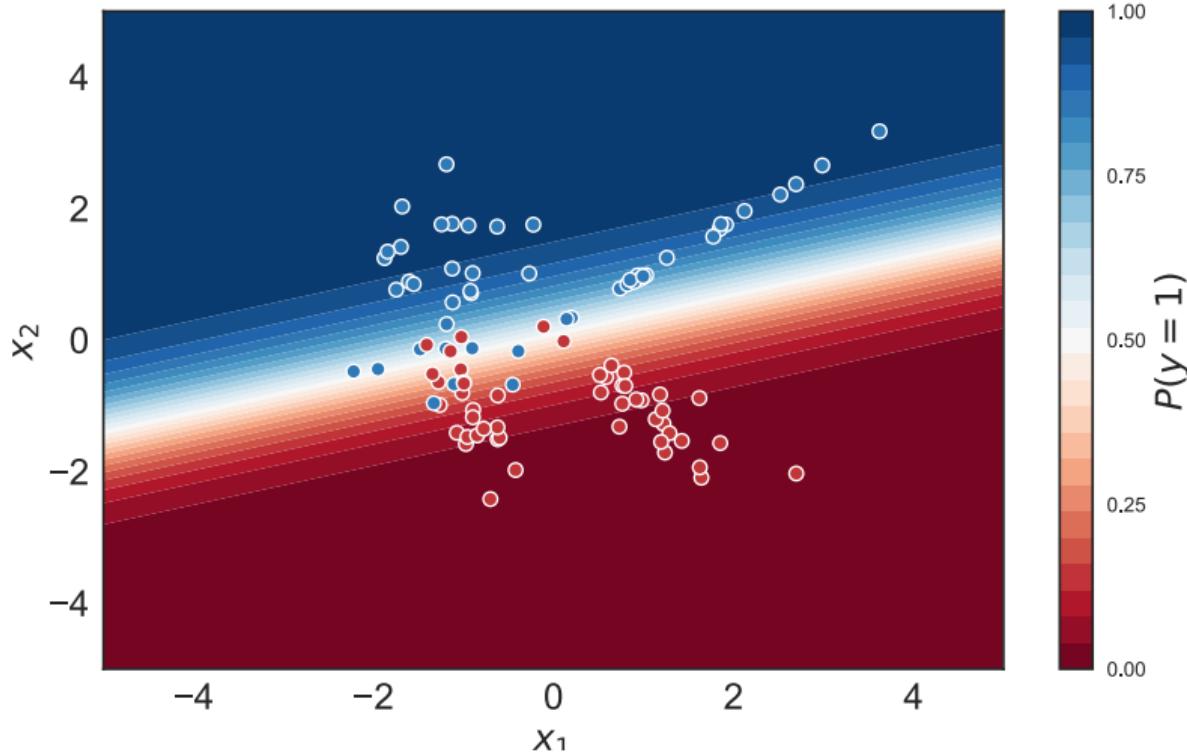
$$\sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

- › Plugging the logistic function into the loss yields an approximation of accuracy

$$\sum_{i=1}^{\ell} (1 + \exp(\mathbf{w}^\top \mathbf{x})) \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$



The logistic regression model



Figures of merits

Classification quality evaluation: accuracy

- › Given a labeled sample $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$, $y_i \in \{-1, +1\}$, and some candidate h , **how well does h perform on X^ℓ ?**
- › Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t : hyperparameter)
- › Obvious choice: **accuracy**

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(\mathbf{x}_i) = y_i]$$

- › Example: Higgs challenge – selection of the **interesting signal** $H \rightarrow \tau\tau$ decay against the **already known background**
- › 164,333 background, 85,667 signal events (66% background)



Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

- › **Rates** are often more informative:

$$\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

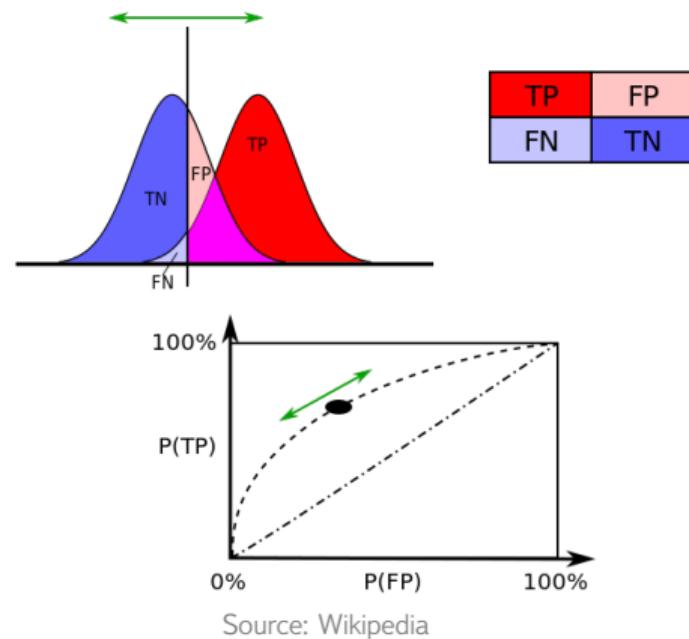
$$\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

- › While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Classification quality: the receiver operating curve

- › Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- › Consider two-dimensional space with coordinates $(\text{TPR}(t), \text{FPR}(t))$, corresponding to various choices of the threshold t
- › The plot $\text{TPR}(t)$ vs. $\text{FPR}(t)$ is called the **receiver operating characteristic (ROC)** curve
- › Area under curve (ROC-AUC) reflects classification quality



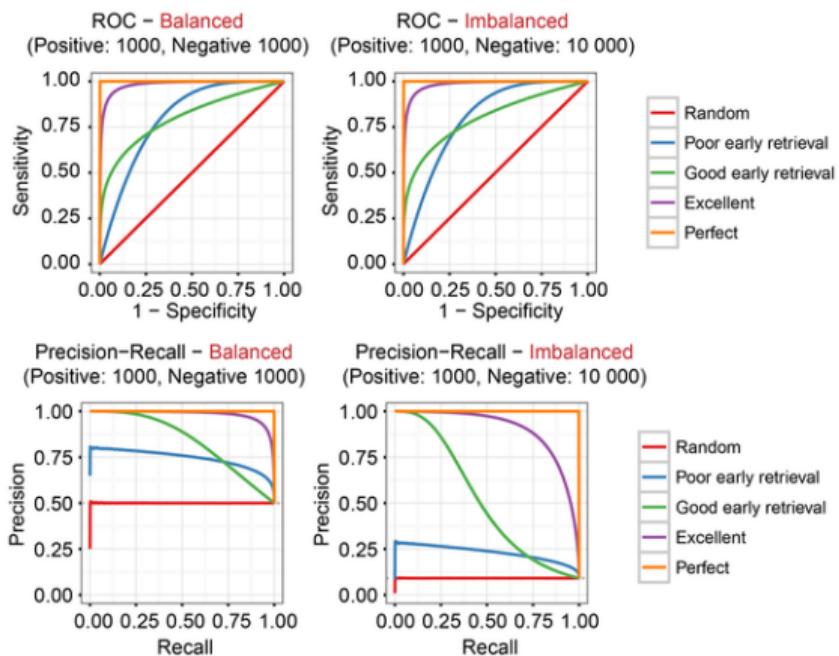
Classification quality: imbalanced data

- › Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- › TPR(t) vs. FPR(t) / ROC is **bad for imbalanced data**: for $\ell = 1000$,
 $n_- = 950$ (high background noise), $n_+ = 50$ (low signal),
a trivial rule $h(\mathbf{x}) = -1$ (“treat everything as background”) would yield:
 - › accuracy(a, X^ℓ) = 0.95 (**bad**)
 - › TPR(a, X^ℓ) = 0. (**OK**)
 - › FPR(a, X^ℓ) = 0. (**bad**)
- › Criteria better suited for imbalanced problems:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Classification quality: imbalanced data

- > The plot recall vs. precision is called the **precision-recall (PR) curve**
- > Recall(t) vs. Precision(t) is **good for imbalanced data**: for $\ell = 1000$,
 $n_- = 950$ (high background noise),
 $n_+ = 50$ (low signal),
a trivial rule $h(\mathbf{x}) = -1$ would yield:
 - > $\text{Recall}(a, X^\ell) = 0.$ (**OK**)
 - > $\text{Precision}(a, X^\ell) = 0.$ (**OK**)



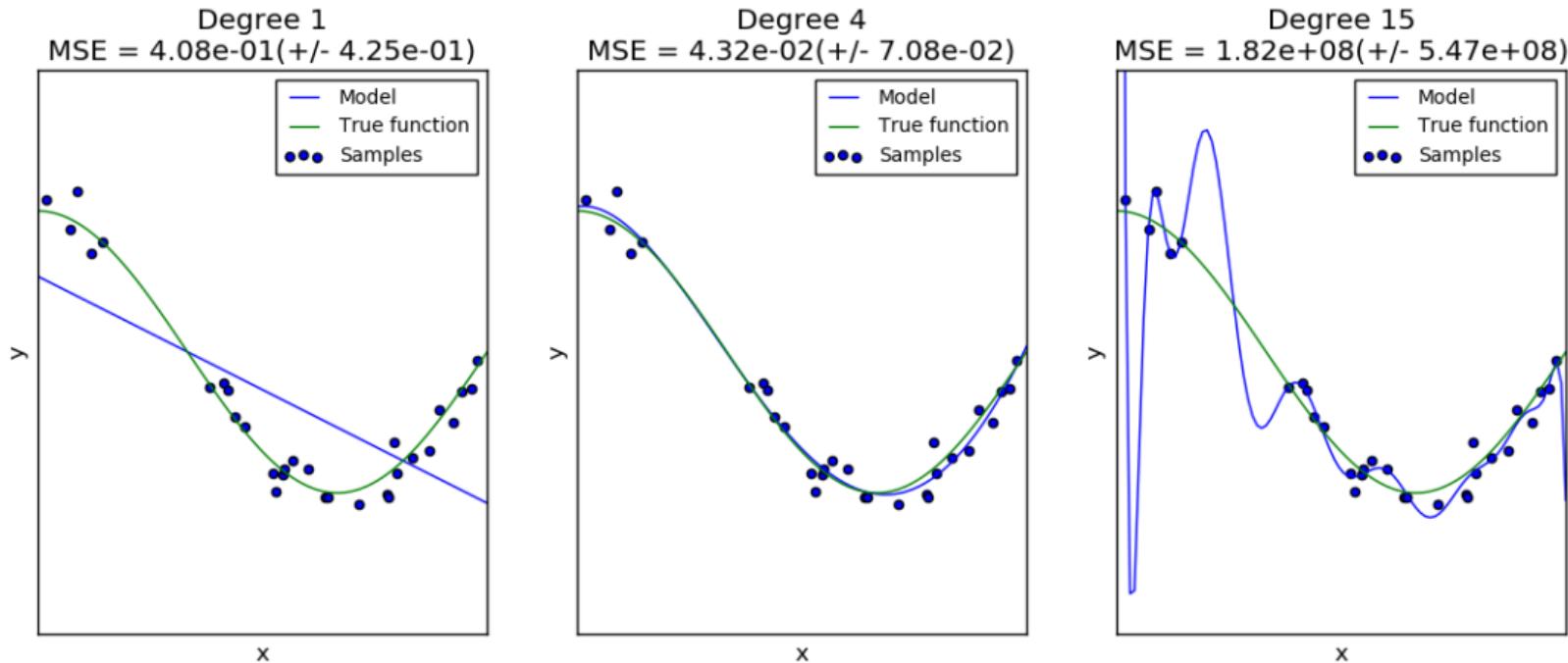
Source: classeval.wordpress.com

Overfitting: how to fool
the linear regression

Generalization and overfitting

- › Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- › Generalization: equally good performance on both new and seen instances
- › How to assess model's generalization ability?
- › Consider an example:
 - › $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \text{Uniform}[0, 1]$
 - › Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \dots, x^{15}\}$
 - › The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$
- › How well do the regression models perform?

Polynomial fits of different degrees



Model validation and selection

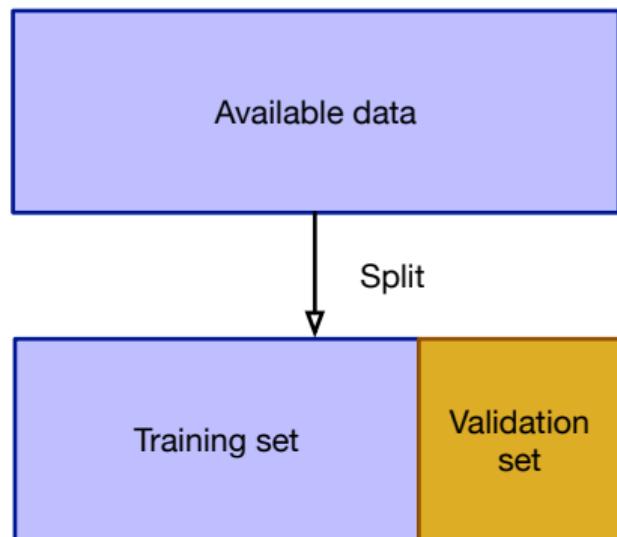
- › We have free parameters in models:
 - › polynomial degree d , subset of features in multivariate regression, kernel width in kernel density estimates, ...
- › **Model selection:** how to select optimal hyperparameters for a given classification problem?
- › **Validation:** how to estimate true model performance?
- › Can we use entire dataset to fit the model?
- › **Yes, but** we will likely get overly optimistic performance estimate
- › **The solution:** rely on held-out data to assess model performance

Assessing generalization ability: train/validation

- › Split training set into two subsets:

$$X^\ell = X_{\text{TRAIN}}^\ell \cup X_{\text{VAL}}^\ell$$

- › Train a model h on X_{TRAIN}^ℓ
- › Evaluate model h on X_{VAL}^ℓ
- › Assess quality using $Q(h, X_{\text{VAL}}^\ell)$
- › **Data-hungry:** can we afford the "luxury" of setting aside a portion of the data for testing?
- › **May be imprecise:** the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split



Assessing generalization ability: cross-validation

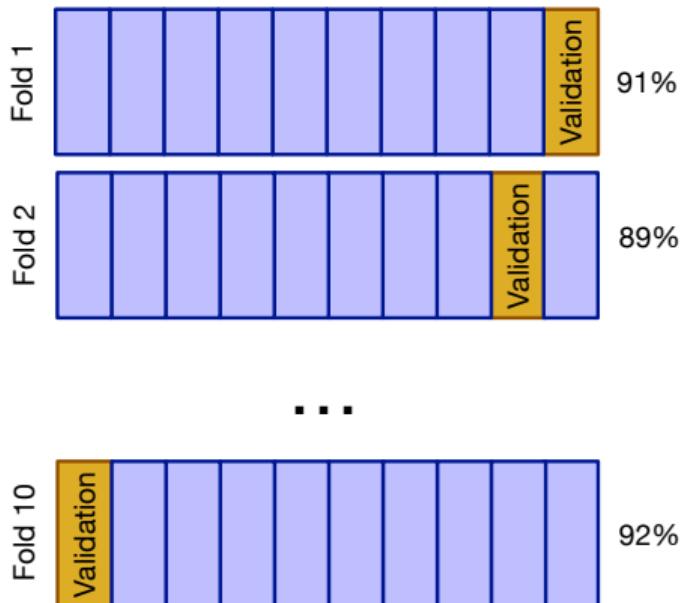
- › Split training set into subsets of equal size

$$X^\ell = X_1^\ell \cup \dots \cup X_K^\ell$$

- › Train K models h_1, \dots, h_K where each model h_k is trained on all subsets **but** X_k^ℓ
- › Assess quality using

$$\text{CV} = \frac{1}{K} \sum_{k=1}^K Q(h_k, X_k^\ell) \text{ (K -fold)}$$

- › Leave-one-out cross-validation:
 $X_k^\ell = \{(\mathbf{x}_k, y_k)\}$ (yes, train ℓ models!)



Cross-validation method: drawbacks

$$\text{CV} = \frac{1}{K} \sum_{k=1}^K Q(h_k, X_k^\ell)$$

Many folds:

- › **Small bias:** the estimator will be very accurate
- › **Large variance:** due to small split sizes
- › **Costly:** many experiments, large computational time

Few folds:

- › **Cheap, computationally effective:** few experiments
- › **Small variance:** average over many samples
- › **Large bias:** estimated error rate conservative or smaller than the true error rate

Regularization

Ad-hoc regularization: motivation

- › Consider the multivariate linear regression problem with $\mathbf{X} \in \mathbb{R}^{d \times d}$

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › Analytic solution involves computing the product $\mathbf{R} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$
- › If $\mathbf{X} = \text{diag}(\lambda_1, \dots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \dots > \lambda_d \rightarrow 0$
(meaning we're in eigenbasis of \mathbf{X}) then

$$\begin{aligned}\mathbf{R} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top = \\ &= \left(\text{diag}(\lambda_1, \dots, \lambda_d) \text{diag}(\lambda_1, \dots, \lambda_d) \right)^{-1} \text{diag}(\lambda_1, \dots, \lambda_d) = \\ &= \text{diag} \left(\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_d} \right), \quad \text{leading to huge diagonal values in } \mathbf{R}\end{aligned}$$

Ad-hoc regularization: L2

- › Regularization: replace **fit** with **fit + penalty** as in

$$Q(\mathbf{w}) \rightarrow Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

- › $R(\mathbf{w})$ is called the regularizer, $\alpha > 0$ – the regularization constant
- › Regularized multivariate linear regression problem

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_2^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › Regularized analytic solution available

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

Why L2 regularization works

- › Analytic solution: compute the regularized operator

$$\mathbf{R} = (\mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^\top$$

- › If $\mathbf{X} = \text{diag}(\lambda_1, \dots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \dots > \lambda_d \rightarrow 0$ (meaning we're in eigenbasis of \mathbf{X}) then

$$\begin{aligned}\mathbf{R} &= (\mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^\top = \\ &= (\text{diag}(\lambda_1, \dots, \lambda_d) \text{diag}(\lambda_1, \dots, \lambda_d) + \text{diag}(\alpha, \dots, \alpha))^{-1} \text{diag}(\lambda_1, \dots, \lambda_d) = \\ &= \text{diag}\left(\frac{\lambda_1}{\lambda_1^2 + \alpha}, \dots, \frac{\lambda_d}{\lambda_d^2 + \alpha}\right),\end{aligned}$$

smoothing diagonal values in \mathbf{R}

More regularizers!

- › L2 regularized multivariate linear regression problem

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › L1 regularized regression (LASSO)

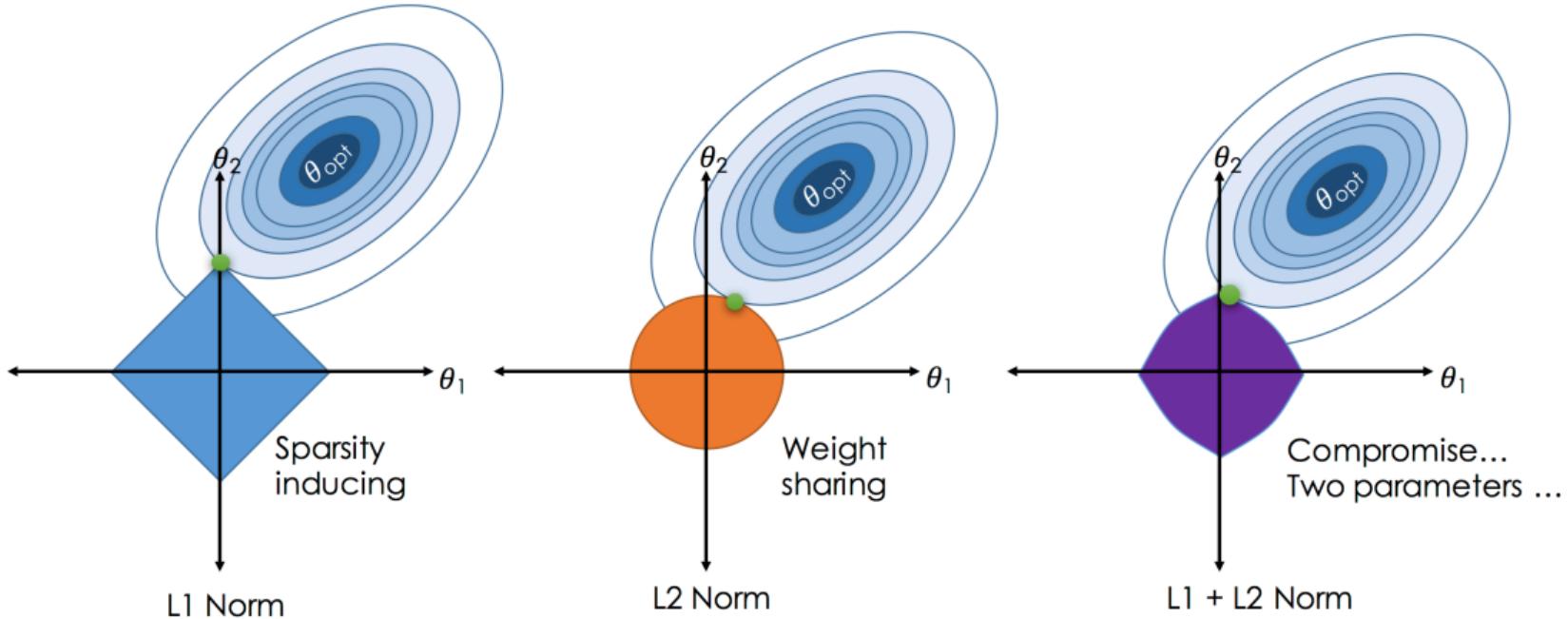
$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_1 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › L1/L2 regularized regression (Elastic Net)

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \alpha_1\|\mathbf{w}\|_1 + \alpha_2\|\mathbf{w}\|_2^2 \rightarrow \min_{\mathbf{w} \in \mathbb{R}^d}$$

- › Convex $Q(\mathbf{w})$: **unconstrained** optimization $Q(\mathbf{w}) + \alpha\|\mathbf{w}\|_1$
is equivalent to **constrained** problem $Q(\mathbf{w})$ s.t. $\|\mathbf{w}\|_1 \leq C$

Geometric interpretation of regularizers



Picture credit: http://www.ds100.org/sp17/assets/notebooks/linear_regression/Regularization.html

Another interpretation of regularizers



Figure: Large parameter space



Figure: Regularized models

Why machine learning works:
a bit of learning theory

Empirical Risk Minimization framework

- › Given a labeled sample $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$ and some candidate h
- › Define the **empirical error** of h as $Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$
(the proportion of sample points on which h errs)
- › $\text{ERM}(X^\ell)$ — find h the minimizes $Q(h, X^\ell)$
- › **No learning is possible without applying prior knowledge**
- › A hypothesis class \mathbb{H} is a set of hypotheses
- › Re-define the ERM rule by searching only inside such a prescribed \mathbb{H}
- › $\text{ERM}_{\mathbb{H}}(X^\ell)$ picks a classifier $h \in \mathbb{H}$ that minimizes the empirical error over members of \mathbb{H}

Guarantees for ERM

Theorem (Guaranteed success for $\text{ERM}_{\mathbb{H}}$)

- › Let \mathbb{H} be a finite class.
- › Let the unknown labeling rule, f , be a member of \mathbb{H} .
- › Then
 - › for every $\varepsilon, \delta > 0$, if $m > (\log(|\mathbb{H}|) + \log(1/\delta))/\varepsilon$,
 - › with probability $> 1 - \delta$ (over the choice of X^ℓ),
 - › any $\text{ERM}_{\mathbb{H}}$ hypothesis has error below ε .

Bonus track

Structural Risk Minimization framework

- › Balancing the model's complexity against its success at fitting the training data
- › Given \mathbb{H} , a decompositon of \mathbb{H} to 'good' \mathbb{H}_n 's and a weight functon w ,
- › On a labeled training sample S of size m ,
- › Find $h \in \mathbb{H}$ that minimizes

$$Q(h, X^\ell) + \min\{n : h \in \mathbb{H}_n\} \varepsilon_n(m, w(n)\delta)$$

where $\varepsilon_n(m, \delta) = \min\{\varepsilon : m_{\mathbb{H}_n}(\varepsilon, \delta) < m\}$
(namely, the minimal error that an m -size sample can guarantee)

Structural Risk Minimization framework

- › SRM has many applications, usually referred to as “ERM with regularization”
- › Adding to the empirical error a “penalty” on complex (or otherwise, undesirable) h 's.
- › Examples include:
 1. Norm of a linear classifier
 2. Description length
 3. Small margins
 4. Low prior likelihood

Linear discriminant analysis [R. Fisher, 1936]

- › Training set $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$ where $y_i \in \{-1, +1\}$

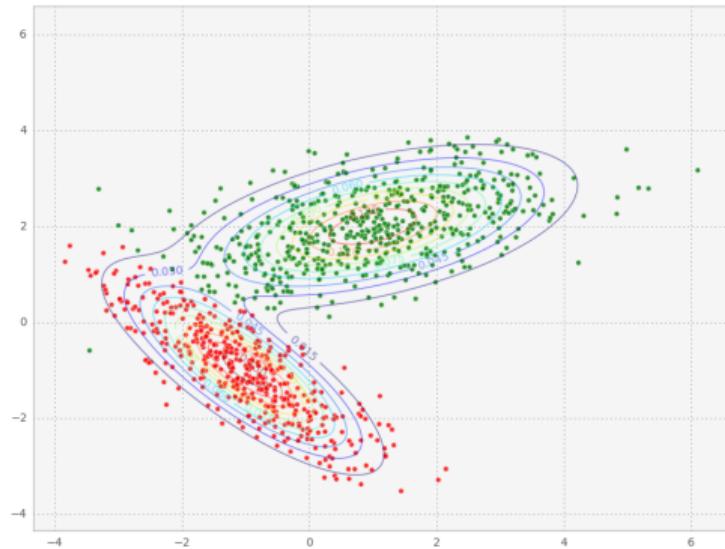
- › The model:

$$P(\mathbf{x}_i | y_i = -1) = \mathcal{N}(\boldsymbol{\mu}_-, \boldsymbol{\Sigma}_-),$$

$$P(\mathbf{x}_i | y_i = +1) = \mathcal{N}(\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$$

- › Minimize error probability via Neyman-Pearson lemma

$$\frac{P(\mathbf{x} | y_i = +1)}{P(\mathbf{x} | y_i = -1)} > t$$



Linear discriminant analysis [R. Fisher, 1936]

- › Log-likelihood ratio

$$\log P(\mathbf{x}|y_i = +1) - \log P(\mathbf{x}|y_i = -1) > T$$

- › Recall that $P(\mathbf{x}|y_i = \pm 1) = \frac{1}{\sqrt{2\pi|\Sigma_{\pm}|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{\pm})^T \boldsymbol{\Sigma}_{\pm}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\pm})\right\}$

- › When $\Sigma_+ = \Sigma_- = \Sigma$, the optimal solution is

$$\mathbf{x}^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) > \frac{1}{2}(T - (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-))$$

which has the form $\mathbf{w}^T \mathbf{x} > c$

- › Homoscedastic case: linear discriminant hyperplane

Linear discriminant analysis

