# Machine Learning Concepts

Professor Marie Roch
San Diego State University

SAN DIEGO STATE UNIVERSITY

# Basic definitions & concepts

- Task – What the system is supposed to do

  e.g. ASR:  f(input speech) → list of words

- Performance measure – How well does it work?
- Experience – How does the machine learn the task?

# Types of experiences; how a learner learns…

- Supervised learning – Learn class conditional distributions: implies class labels are known
  $P(\omega|x)$ : probability of class ω given evidence $x$

- Unsupervised learning – No labels are provided, learn P(x) and possibly group x's into clusters

- Reinforcement learning – Learner actions are associated with payouts for actions in environment.

# Experience data set: Design matrix

label (supervised problems)

feature vector

$$
\begin{bmatrix}
x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,D} & y_1 \\
x_{2,1} & \cdots & & & & y_2 \\
x_{3,1} & \cdots & & & & y_3 \\
\vdots & & & & & \vdots \\
x_{N,1} & x_{N,2} & x_{N,3} & \cdots & x_{N,D} & y_N
\end{bmatrix}
$$

Learning sets of functions may be used if some features are missing. E.g. fn $f_0$ for all features, $f_1$ if feature 1 is missing, etc.

# A Cardinal Rule

OF MACHINE LEARNING

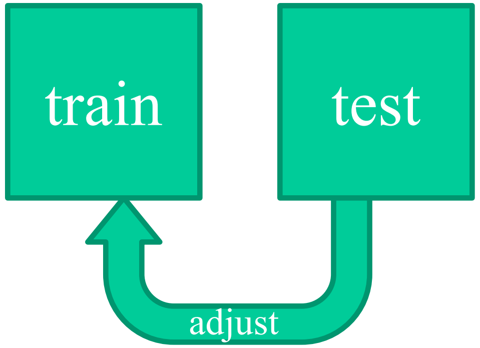
THOU SHALT NOT TEST ON THY TRAINING DATA

# Performance

- A metric that measures how well a learner is able to accomplish the task
- Metrics can vary significantly (more on these later), examples:
  - loss functions such as squared error
  - cross entropy

# Partitioning data

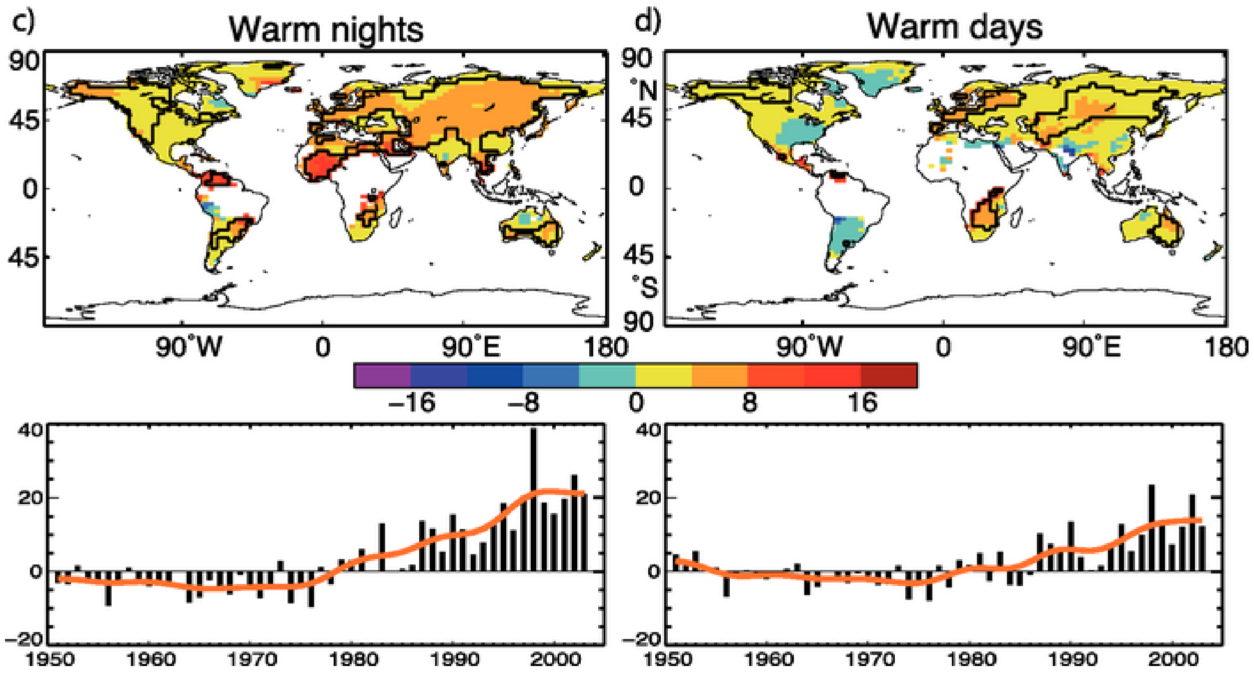
- Training data – experience for learner
- Test data – performance measurement
- Evaluation data
  - Only used once all adjustments are made
  - It is common to:




- Adjustments are a form of training (see 5.3)
- Evaluation provides an independent test

# Regression

Given a set of features and response, predict a response




IPCC 4th Assessment Report
Climate Change 2007

Observed changes in number of warm days/night with extreme temperatures (normative 1961-1990)

# Linear regression
# A simple learning algorithm

Predict response from data

$$\hat{y} = w^T x \qquad w, x \in \Re^N, \hat{y} \in \Re$$

- *w* is the weight vector
- Goal: Maximize performance on test set.

  Learn *w* to minimize some criterion, e.g. mean squared error (MSE)

$$MSE_{\text{test}} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}_i - y_i\right)^2 \equiv \frac{1}{m}\left\|\hat{y}^{\text{test}} - y^{\text{test}}\right\|_2^2$$

SAN DIEGO STATE UNIVERSITY

$\|x\|_p$ denotes the $L^p$ norm $\left(\sum_i |x_i|^p\right)^{1/p}$ read Goodfellow et al. 2.5

# Linear regression

- Cannot estimate *w* from test data
- Use the training data
- Minimize

$$MSE_{\text{train}} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}_i - y_i\right)^2 \equiv \frac{1}{m}\left\|\hat{y}^{\text{train}} - y^{\text{train}}\right\|_2^2$$

For convenience, we will usually omit the variable descriptor train when describing training.

# Linear regression

MSE minimized when gradient is zero

$$\nabla_w MSE = 0$$

$$\nabla_w \frac{1}{m}\left\|\hat{y} - y\right\|_2^2 = 0$$

$$\nabla_w \left\|\hat{y} - y\right\|_2^2 = 0$$

$$\nabla_w \left\|Xw - y\right\|_2^2 = 0 \text{ as } Xw = \hat{y}$$

# Linear regression

$$\nabla_w \left\| Xw - y \right\|_2^2 = 0$$

$$\nabla_w \left( Xw - y \right)^T \left( Xw - y \right) = 0 \qquad L_2^2 \text{ norm in matrix notation}$$

$$\nabla_w \left( \left( Xw \right)^T - y^T \right) \left( Xw - y \right) = 0 \qquad \text{transpose distributive over addition}$$

$$\nabla_w \left( w^T X^T - y^T \right) \left( Xw - y \right) = 0 \quad \text{as } (\mathrm{AB})^T = B^T A^T \qquad \text{Goodfellow et al. eqn 2.9}$$

$$\nabla_w \left( w^T X^T X w - w^T X^T y - y^T X w + y^T y \right) = 0$$

$$\nabla_w \left( w^T X^T X w - y^T X w - y^T X w + y^T y \right) = 0 \quad \text{as } w^T X^T y = y^T \left( w^T X^T \right)^T = y^T X w$$

$$\nabla_w \left( w^T X^T X w - 2 y^T X w + y^T y \right) = 0$$

# Linear regression

$$\nabla_w \left( w^T X^T Xw - 2y^T Xw + y^T y \right) = 0$$

$$\nabla_w \left( w^T X^T Xw - 2y^T Xw \right) = 0$$ as $y^T y$ independent of $w$

$$\nabla_w \left( X^T Xw^2 - 2y^T Xw \right) = 0$$ as $w^T X^T X = X^T Xw$

$$2X^T Xw - 2y^T X = 0$$ derivative

$$X^T Xw = y^T X$$

$$w = \left( X^T X \right)^{-1} y^T X$$ matrix inverse $A^{-1} A = I$

These are referred to as the normal equations

# Linear regression

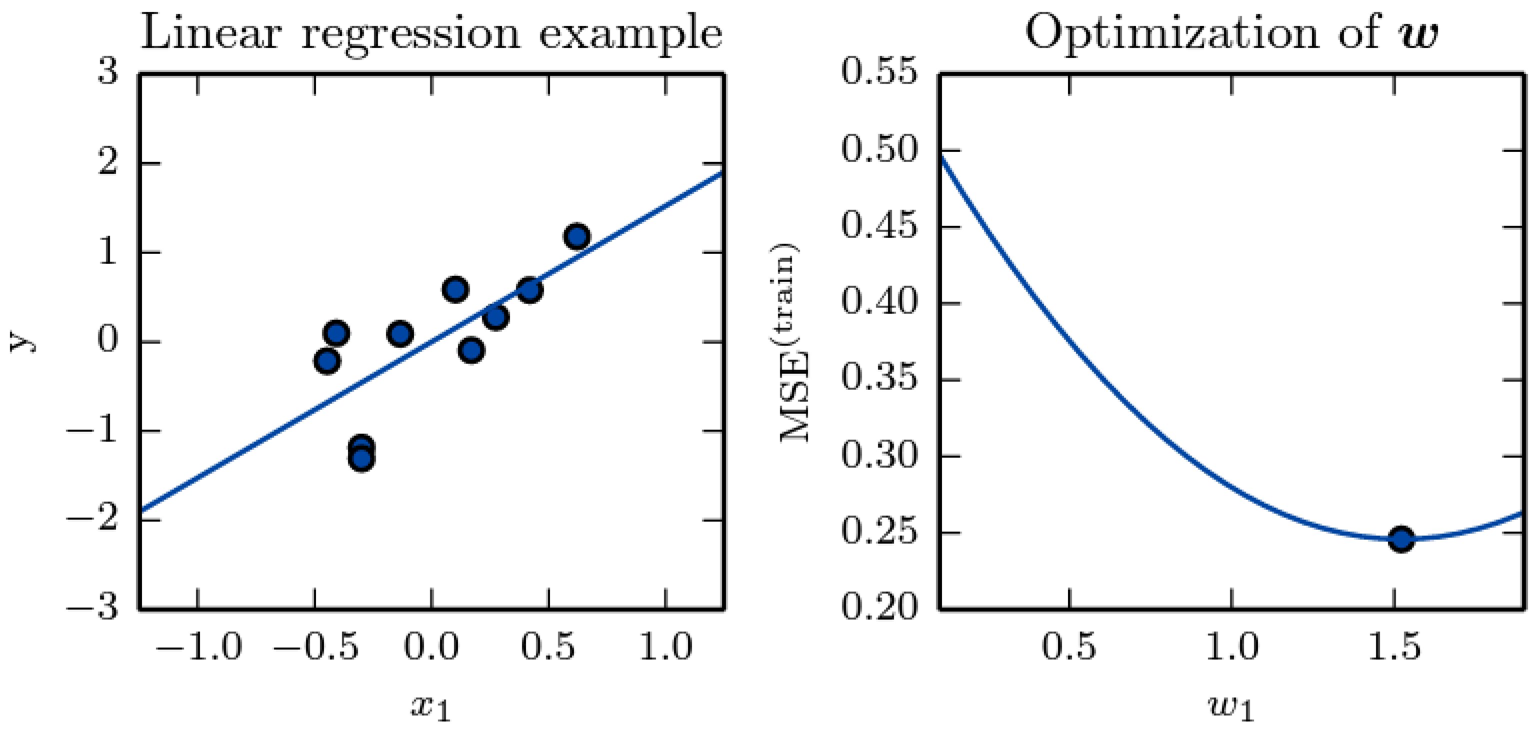



Fig. 5.1 Goodfellow et al.

Normal equations optimize w

# Linear regression

Regression formula forces curve to pass through origin

Remove restriction:

- Add bias term $\hat{y} = w^T x + b$
- To use normal equations, use modified $x$
- Last term of new weight vector $w$ is bias

$$x_{mod} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \\ 1 \end{bmatrix}$$

# Notes on learning

- A learner that performs well on unseen data is said to *generalize* well.
- When will learning on training data produce good generalization?
  - Training and test data drawn from the same distribution
  - Large enough training set to learn the distribution

# Underfitting & Overfitting

- Underfit
  Model cannot learn training data well

- Overfit
  Model does not generalize well

- These properties are related to model *capacity*

# Capacity

What kind of functions can we learn?

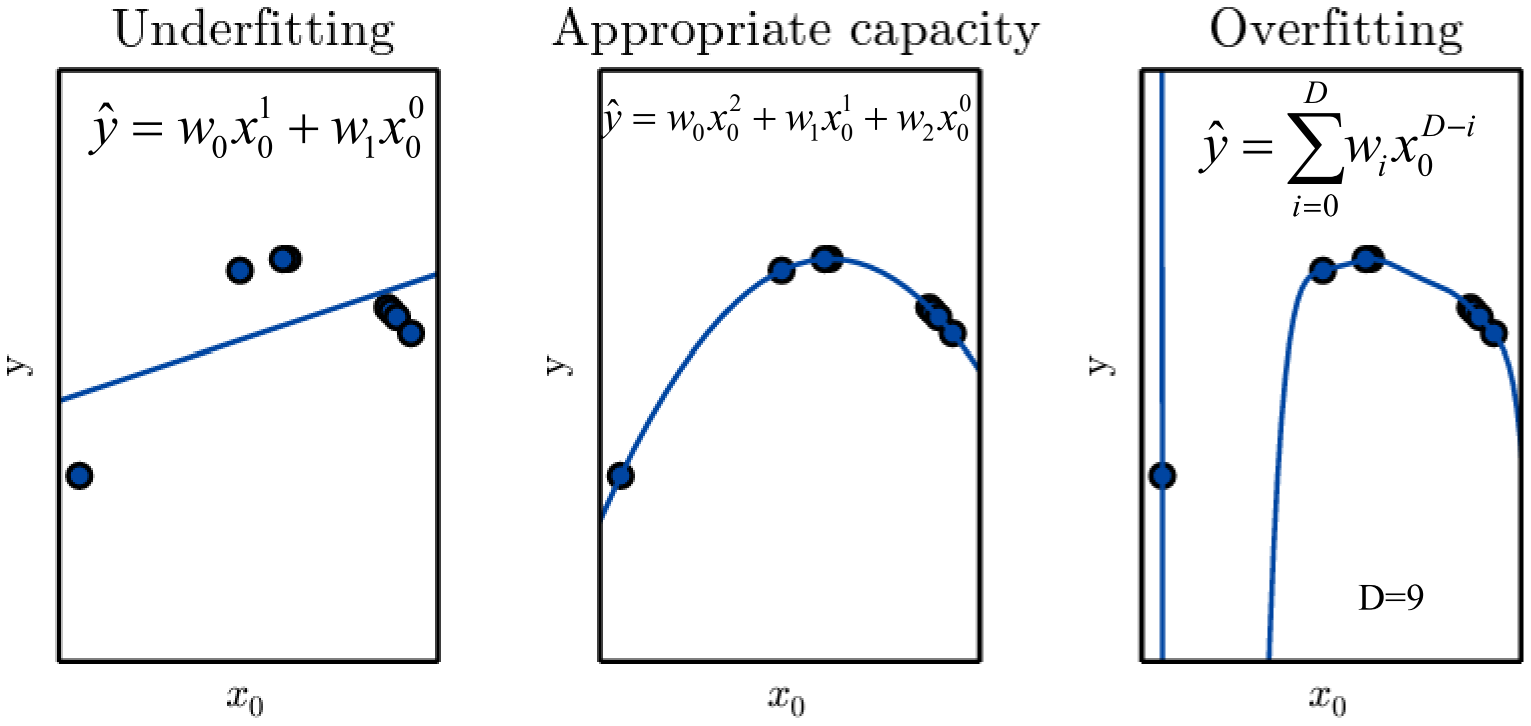
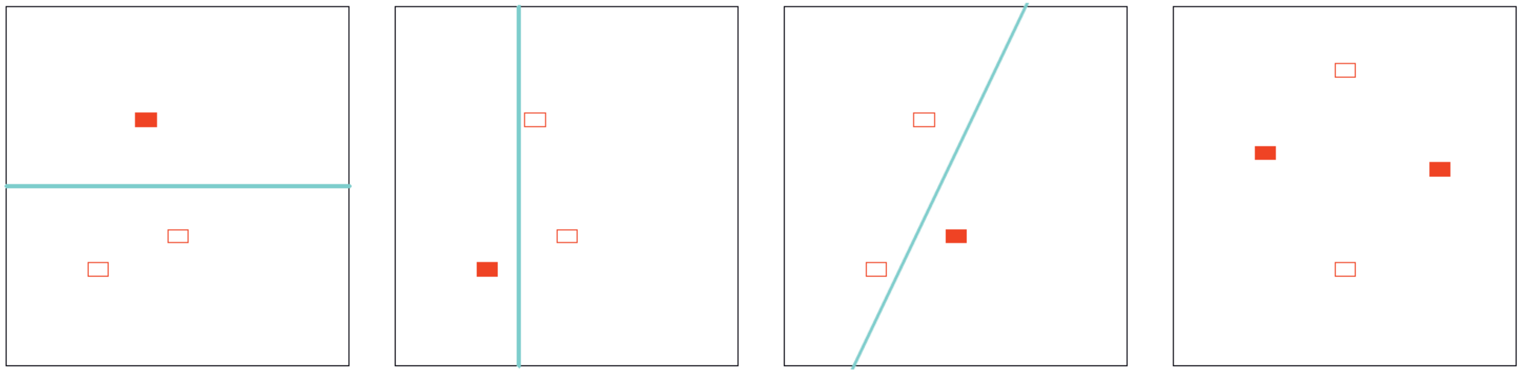



Fig. 5.2 Goodfellow et al.

Here, model order affects capacity

# Shattering points

Points are *shattered* if a classifier can separate them regardless of their binary label



Hastie et al., 2009, Fig 7.6

We can shatter the three points, but not four with a linear classifier

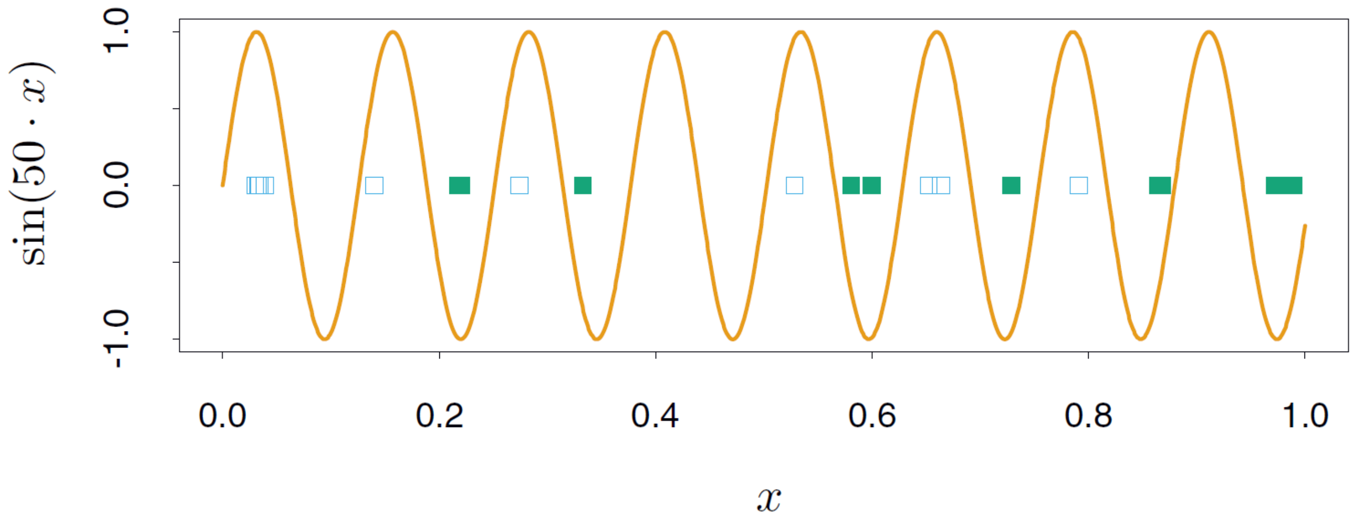
# Capacity

- Representational capacity – best function that can be learned within a set of learnable functions

- Frequently a difficult optimization problem
  - We might learn a suboptimal solution
  - This is called *effective capacity*

# Measuring capacity

- Model order is not always a good predictor of capacity




Hastie et al., 2009, Fig .7.5

Label determined by sign of function. Increasing frequency of sinusoid enables ever finer distinctions…

SAN DIEGO STATE UNIVERSITY

# Measuring capacity

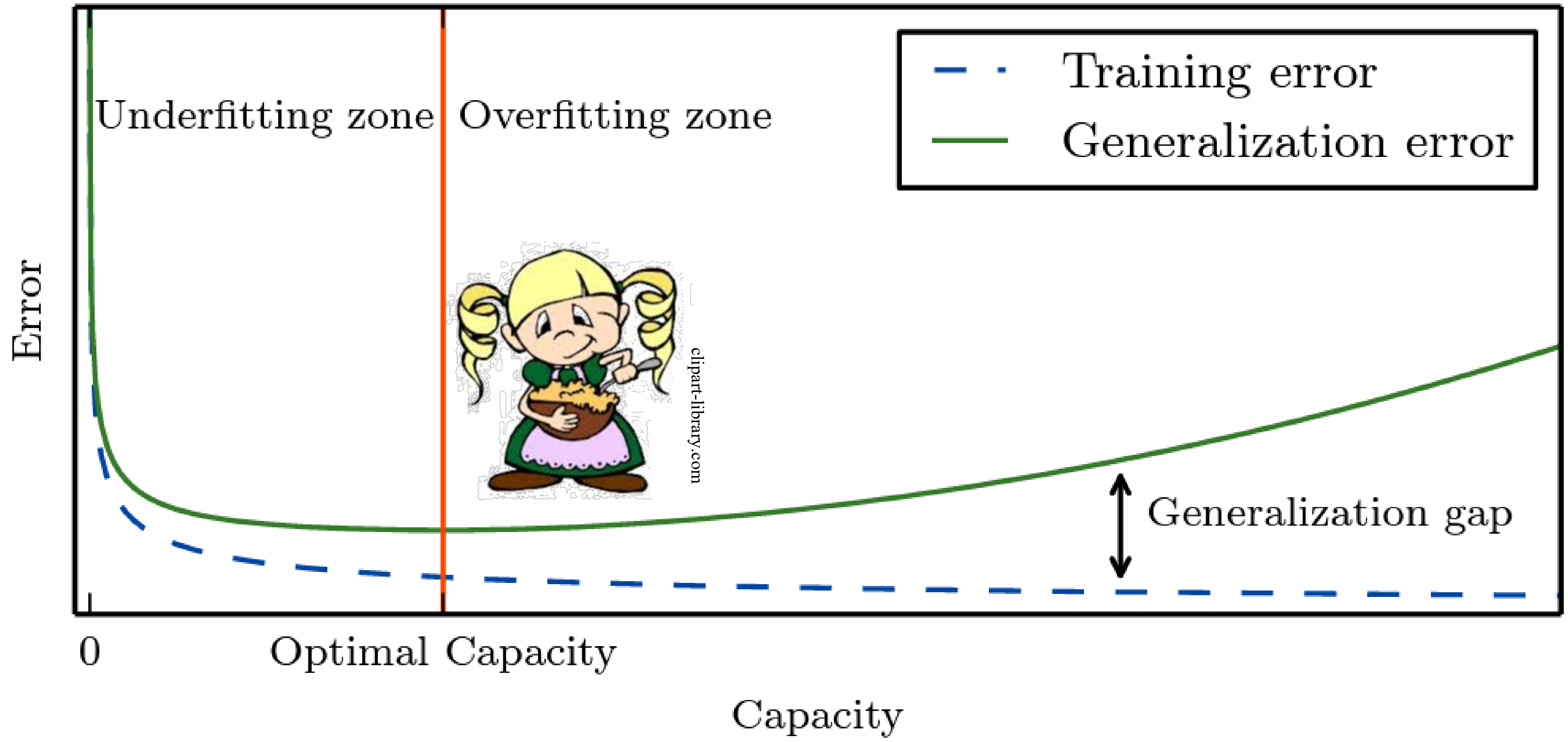
- Vapnik Chervonikis (VC) dimension
  - for binary classifiers
  - Largest # of points that can be shattered by a family of classifiers.

- In practice, hard to estimate for deep learners… so why do we care?

# Predictions about capacity

- Goal: Minimize the generalization error
- Hard; perhaps minimize difference Δ
  Δ = |training error - generalization error|
- Learning theory
  - Models with higher capacity have higher upper bound on Δ
  - Increasing amount of training data decreases Δ's upper bound

# Recap on capacity




Hastie et al., 2009, Fig 7.6

# The No Free Lunch Theorem

Expected performance of *any* classifier across all possible generalization tasks is no better than any other classifier.


http://elsalvavidas.mx/lifehacking/inicia-tu-aventura-para-estudiar-en-el-extranjero-con-estos-tips/1182/

A classifier might be better for some tasks, but no classifier is universally better than others.

# N-fold cross validation

- Problem:
  - More data yields better training
  - Getting more data can be expensive
- Workaround
  - Partition data into N different groups
  - Train on N-1 groups, test on last group
  - Rotate to have N different evaluations



Scikit learn's KFold and StratifiedKFold will split example indices for you: sklearn.model_selection.*

# Regularization

- Remember:  Learners select a solution function from a set of hypothesis functions.
- Optimization picks the function that minimizes some optimization criterion
- Regularization lets us express a preference for certain types of solutions

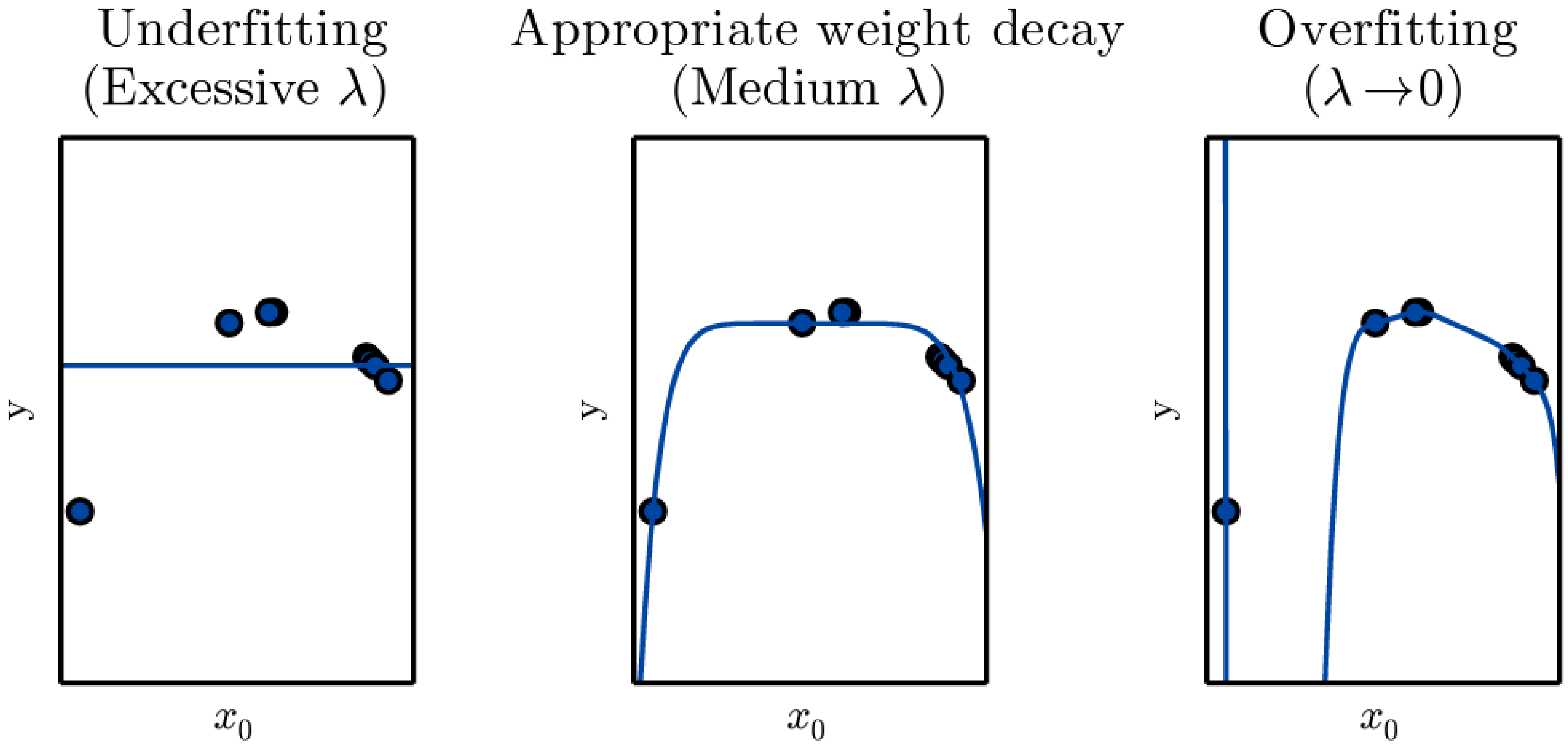
# Example:
# High Dimensional Polynomial Fit

- Suppose we want small coefficients
  Remember: $\hat{y} = w^T x$
- This happens when $\Omega(w) = w^T w$ is small and results in shallower slopes
- We can define a new criterion:

$$J(w) = MSE + \lambda\Omega(w) = MSE + \lambda w^T w$$

where λ controls regularization influence

# $9^{th}$ degree polynomial fit with regularization




Goodfellow et al. Fig. 5.5

# Point estimators

- An approximation of interest
- Examples:
  - a statistic of a distribution

    e.g. $\hat{\mu} = \frac{1}{N}\sum_i x^{(i)}$ is an approximation of $\mu$
  - a parameter of a classifier

    e.g. a mixture model weight or distribution parameter

# Point estimators

In general, it is a function of data

$$\hat{\Theta}_m = f(x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)})$$

and may even be a classifier function that maps data to a label (*function estimation*).

# Bias

How far from the true value is our estimator?

$$\text{bias}(\hat{\theta}_m) = E[\hat{\theta}_m] - \theta$$

Goodfellow et al. give an example with a Bernoulli distribution that we have not yet covered (read 3.9.1). Bernoulli distributions are good for estimating the number of times that a binary event occurs (e.g., 42 head in 100 coin tosses).

# Bias of sample mean

$$\text{bias}(\hat{\mu}_m) = E[\hat{\mu}_m] - \mu$$

$$= E\left[\frac{1}{N}\sum_i x^{(i)}\right] - \mu$$

$$= \frac{1}{N} E\left[\sum_i x^{(i)}\right] - \mu$$

$$= \frac{1}{N} N \cdot E[X] - \mu \quad x^{(i)} \text{ is a random var}$$

$$= \mu - \mu = 0 \quad \text{unbiased estimator}$$

# Bias

- Read more examples in Goodfellow et al.
- Bias of classifier functions?
  - We are trying to estimate the Bayes classifier.
  - Bias is amount of error over that

# Variance

- Already defined: $Var(X) = E[(X-\mu)^2]$
- Variance of classifier functions
  – Variance of a mean classification result, e.g., error rate

$$Var(\hat{\mu}_m) = Var\left(\frac{1}{m}\left(X_1 + X_2 + \ldots + X_m\right)\right)$$

$$= \frac{1}{m^2} Var\left(X_1 + X_2 + \ldots + X_m\right) \text{ as } Var(kX) = k^2 Var(X) \; ^{\dagger}$$

$$= \frac{1}{m^2} m\sigma^2 = \frac{1}{m}\sigma^2 \text{ or equivalantly, standard error } \mathrm{SE}(\hat{\mu}_m) = \frac{\sigma}{\sqrt{m}}$$

† it can be shown that $Var(X) = E[X^2] - E[X]^2, Var(kX) = k^2 Var(X)$ follows from this

# Bias & Variance

- Variance gives us an idea of classifier sensitivity to different data
- Distribution of mean approaches a normal distribution (central limit theorem)
- Together can estimate with 95% confidence that the real mean lies within:

$$\hat{\mu}_m - 1.96 SE(\hat{\mu}_m) \leq \mu_m \leq \hat{\mu}_m + 1.96 SE(\hat{\mu}_m)$$

# Information Theory

A quick trip down the rabbit hole...

- Details in Goodfellow 3.13
- Needed for maximum likelihood estimators

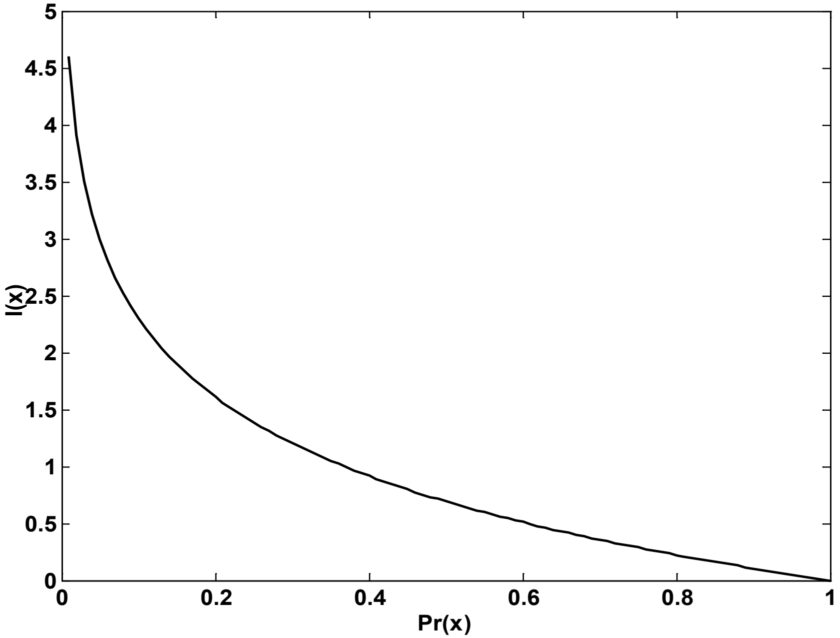

British Postal Service, Graham Baker-Smith 2015

# Quantity of information

- Amount of surprise that one sees when observing an event.

$$I(x_i) = \log \frac{1}{\mathrm{P}(x_i)}$$

- If an event is rare, we can derive a large quantity of information from it.



# Quantity of information

- Why use log?
  - Suppose we want to know the information in two independent events:

$$
\begin{aligned}
I(x_1, x_2) &= \log \frac{1}{\mathrm{P}(x_1, x_2)} \\
&= \log \frac{1}{\mathrm{P}(x_1)\,\mathrm{P}(x_2)} \qquad x_1, x_2 \text{ independent} \\
&= \log \frac{1}{\mathrm{P}(x_1)} + \log \frac{1}{\mathrm{P}(x_2)} \\
&= I(x_1) + I(x_2)
\end{aligned}
$$

San Diego State University

# Entropy

- Entropy is defined as the expected amount of information (average amount of surprise) and is usually denoted by the symbol H.

$$
\begin{aligned}
H(X) &= E[I(X)] \\
&= \sum_{x_i \in S} \mathrm{P}(x_i) I(x_i) && S \text{ is all possible symbols} \\
&= \sum_{x_i \in S} \mathrm{P}(x_i) \log \frac{1}{\mathrm{P}(x_i)} && \text{definition } I(x_i) \\
&= E[-\log \mathrm{P}(X)]
\end{aligned}
$$

# Discrete vs continuous

**Discrete**

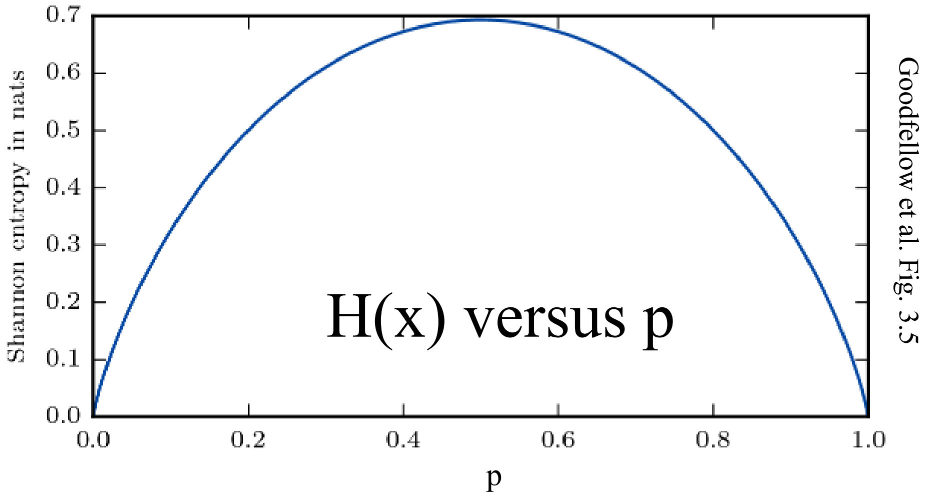
- Shannon Entropy
- Use $\log_2$
- Units called
  - bits, or sometimes
  - Shannons

**Continuous**

- Differential entropy
- Use $\log_e$
- Units called nats

# Example

- Assume
  - X = {0, 1}
  - $$P(X) = \begin{cases} p & X = 0 \\ 1-p & X = 1 \end{cases}$$




- Then

$$\begin{aligned} H(X) &= E[I(X)] \\ &= -p\log p - (1-p)\log(1-p) \end{aligned}$$

# Comparing distributions

- How similar are distributions P and Q?
  Recall
  - X~P means that X has distribution P
  - we denote its probability as $P_P$ and information as $I_P$
- How much more information do we need to represent P's distribution using Q:

$$
\begin{aligned}
E_{X\sim P}\left[I_Q - I_P\right] &= E_{X\sim P}\left[-\log P_Q - (-\log P_P)\right] \\
&= E_{X\sim P}\left[\log \frac{P_P}{P_Q}\right] \\
&= \sum_x P_P\,(x) \log \frac{P_P(x)}{P_Q(x)}
\end{aligned}
$$

# Comparing distributions

- This is known as the Kullback-Leibler (KL) divergence from Q to P:

$$D_{KL}(P||Q) = E_{X\sim P}\left[\log\frac{P_P(X)}{P_Q(X)}\right] = E_{X\sim P}[-\log P_Q\,(X) + \log P_P\,(X)\,]$$

$$D_{KL}(P\,||\,Q) = 0 \text{ iff } P \equiv Q, \text{ otherwise } D_{KL}(P\,||\,Q) > 0$$

note: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ $\rightarrow$ not a distance measure

# Cross entropy H(P,Q)

- Calculates total entropy in two distributions

$$H(P,Q) = E_{X\sim P}\left[-\log P_Q\,(X)\right]$$

- Can be shown to have the entropy of P plus the KL divergence from Q to P.

$$H(P,Q) = H(P) + D_{KL}(P||Q)$$

- Interesting as we sometimes want to minimize KL divergence…

# Cross entropy

Minimizing the KL divergence minimizes the cross entropy:

$$
\begin{aligned}
&H(P,Q) = E_{X\sim P}\left[-\log P_Q\,(X)\right] = H(P) + D_{KL}(P||Q) \\
&= H(P) + E_{X\sim P}\left[\log P_P\,(X) - \log P_Q\,(X)\right] \\
&= -E_{X\sim P}[\log P_P\,(X)] + E_{X\sim P}[\log P_P\,(X)] + E_{X\sim P}\left[-\log P_Q\,(X)\right] \\
&= E_{X\sim P}\left[-\log P_Q\,(X)\right]
\end{aligned}
$$

Suggests that if we are trying to fit a distribution to data, minimizing the cross entropy H(ActualDist, ModelDist) may be appropriate.

# Maximum Likelihood Estimation (MLE)

- Method to estimate model parameters
- Suppose we have *m* independent samples drawn from a distribution
- Can we fit a model with parameters ϴ?

$$\mathsf{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$$

$$\theta_{ML} = \arg\max_{\theta} P(\mathsf{X} \mid \theta)$$

# MLE

- The $x^{(i)}$'s are independent, so

$$\theta_{ML} = \arg\max_{\theta} P(\mathrm{X} \mid \theta) = \arg\max_{\theta} \prod_{i} P(x^{(i)} \mid \theta)$$

- Log transform for numerical stability

$$\theta_{ML} = \arg\max_{\theta} \prod_{i} P(x^{(i)} \mid \theta) = \arg\max_{\theta} \sum_{i} \log P(x^{(i)} \mid \theta)$$

- Traditional to use derivatives and solve for the maximum value.

# MLE through optimization

- Let us reframe this problem:

$$\theta_{ML} = \arg\max_{\theta} \sum_{i} \log P_{model}\,(x^{(i)}|\theta)$$
$$= \arg\max_{\theta} E_{X\sim\hat{P}_{data}}[\log P_{model}\,(x|\theta)]$$

Why can we think of this as $E[\cdot]$?

- Maximized when $P_{\text{model}}$ is most like $\hat{P}_{\text{data}}$

- What does this remind us of?

# MLE through optimization

- $D_{KL}(\hat{P}_{data}||P_{model})$ minimized as $P_{model}$ becomes more like $\hat{P}_{data}$
- Recall

$$D_{KL}(\hat{P}_{data}||P_{model}) = E_{X\sim\hat{P}_{data}}[\log \hat{P}_{data}(X) - \log P_{model}(X)]$$

- $E_{X\sim\hat{P}_{data}}[\log \hat{P}_{data}(X)]$ constant with the same X, so we only need minimize the second term

$$-E_{X\sim\hat{P}_{data}}[\log P_{model}(X)]$$

# MLE through optimization

- We now have a good framework to estimate a model $\Theta$ even when do not have a good parametric model
- We know that we can maximize the likelihood of the data with respect to model $\Theta$ by minimizing the cross entropy between the data and model

# Conditional log-likelihood & MSE

- Instead of thinking of predicting value $\hat{y} = wx$, what if we predicted a conditional probability?

$$P(\hat{y}|x, \Theta)$$

- Regression: only one possible output.
- MLE estimates a distribution: support for multiple outcomes, can think of this as a noisy prediction.

*Theory behind conditional log-likelihood & its relationship to mean squared error will not be tested*

# Conditional log-likelihood & MSE

- Assume $Y|X \sim n(\mu, \sigma^2)$
  - learn $\mu$ such that $E\left[Y \mid x^{(i)}\right] = y^{(i)}$
  - $\sigma^2$ fixed (noise)
- Can formulate as an MLE problem

$$\theta_{\mathrm{ML}} = \arg\max_{\theta} P(Y \mid X; \theta)$$

where parameter $\Theta$ is our weights $w$.

# Conditional log-likelihood & MSE

$\theta_{\mathrm{ML}} = \arg\max_{\theta} P(Y|X;\theta)$

$= \arg\max_{\theta} \prod_i P(y^{(i)}|x^{(i)};\theta)$ if $x^{(i)}$'s independent & identically distributed

$\log \theta_{\mathrm{ML}} = \arg\max_{\theta} \sum_i \log P\,(y^{(i)}|x^{(i)};\theta)$

$= \arg\max_{\theta} \sum_i \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\hat{y}^{(i)}-\mu)^2}{2\sigma^2}}$ prediction of $y^{(i)}$ is $\hat{y}^{(i)}$

$= \arg\max_{\theta} \sum_i \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\hat{y}^{(i)}-y^{(i)})^2}{2\sigma^2}}$ as we want $E[Y|x^{(i)}] = y^{(i)}$

# Conditional log-likelihood & MSE

$$\log \theta_{\text{ML}} = \arg\max_{\theta} \sum_i \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\hat{y}^{(i)} - y^{(i)})^2}{2\sigma^2}}$$

$$= \arg\max_{\theta} \sum_i \left( -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{\left(\hat{y}^{(i)} - y^{(i)}\right)^2}{2\sigma^2} \right)$$

$$= \arg\max_{\theta} \left( -\frac{m}{2}\log(2\pi) - m\log\sigma - \sum_i \frac{\left(\hat{y}^{(i)} - y^{(i)}\right)^2}{2\sigma^2} \right)$$

$$= \arg\max_{\theta} \left( -\sum_i \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \right) \text{ as } \sigma \text{ is constant}$$

Equivalent to maximizing $\sum_i \left(\hat{y}^{(i)} - y^{(i)}\right)^2$ which has the same form as or MSE optimization:

$$MSE_{\text{train}} = \frac{1}{m}\left\|\hat{y}^{\text{train}} - y^{\text{train}}\right\|_2^2$$

# MLE limitations

- MLE can only recover the distribution if the parametric distribution is appropriate for the data.
- If data drawn from multiple distributions with varying parameters, MLE can still estimate distribution, but information about underlying distributions is lost.

# Optimization

- We have seen closed form (single equation) optimization with linear regression.
- Not always so lucky… how do we optimize more complicated things?

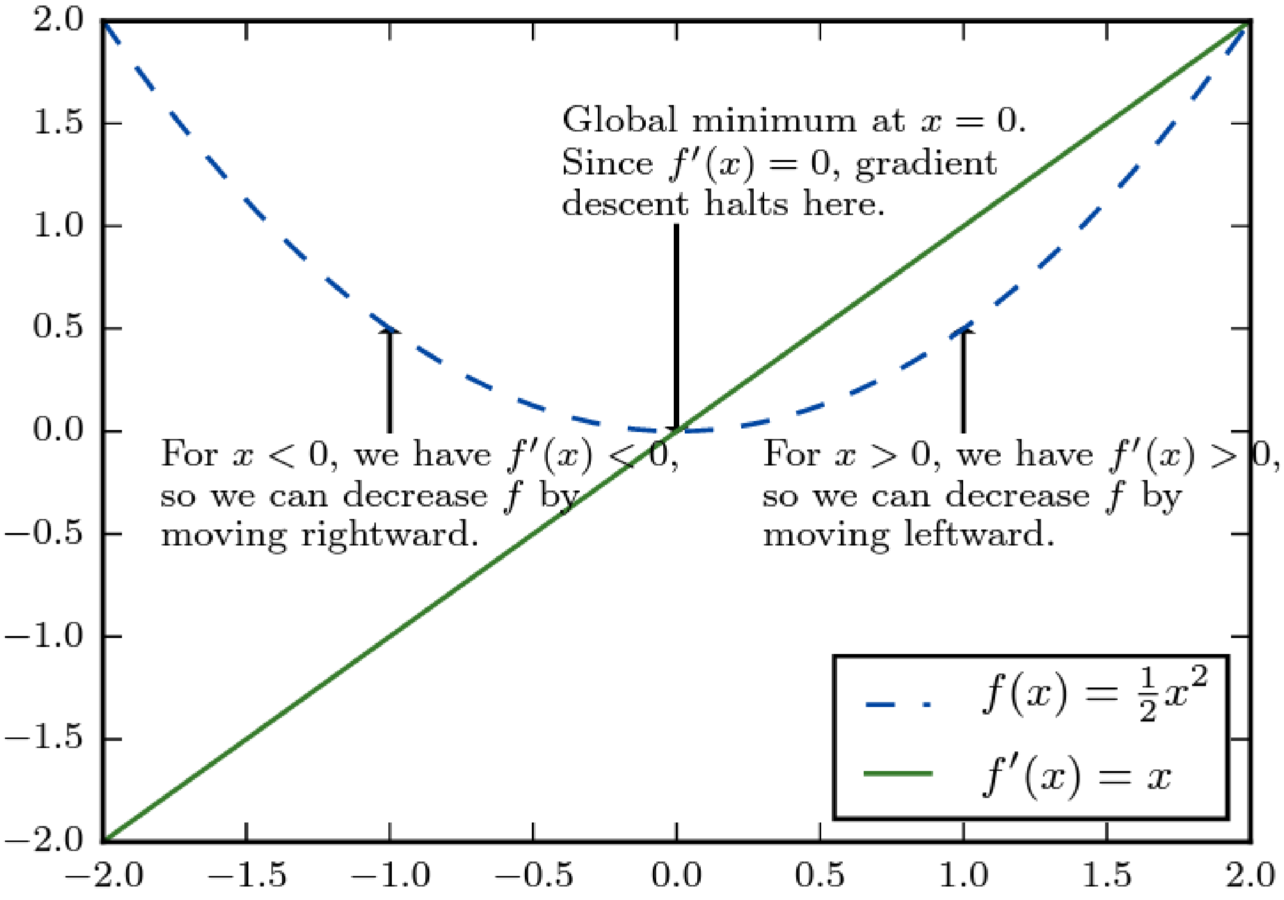
# Optimization

- Select an *objective* function *f(x)* to optimize
  e.g. MSE, KL divergence

- Without loss of generality, we will always consider minimization.

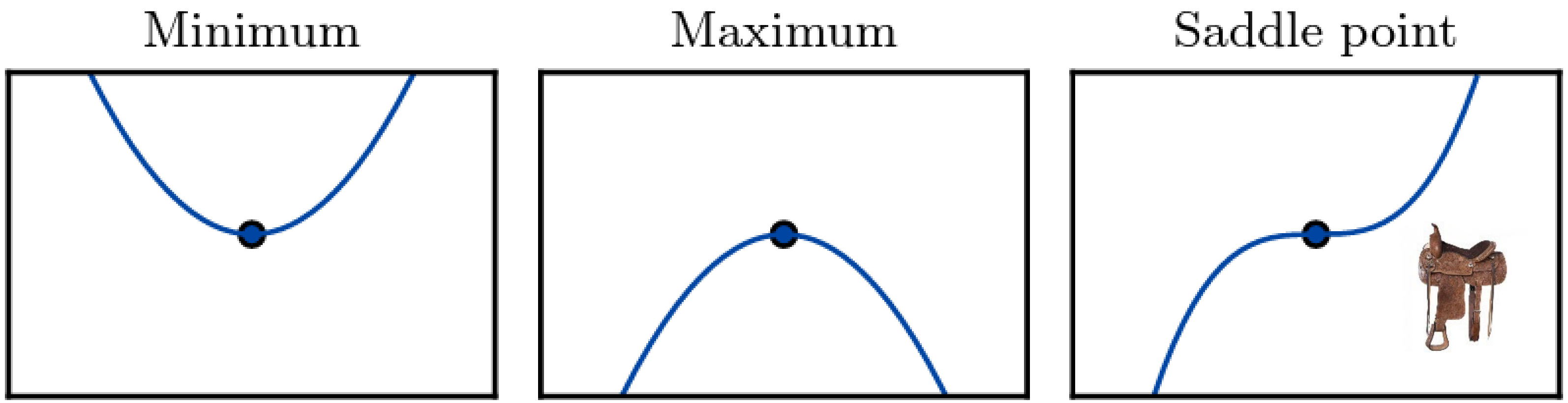
  *Why can we do this?*

# Gradient descent



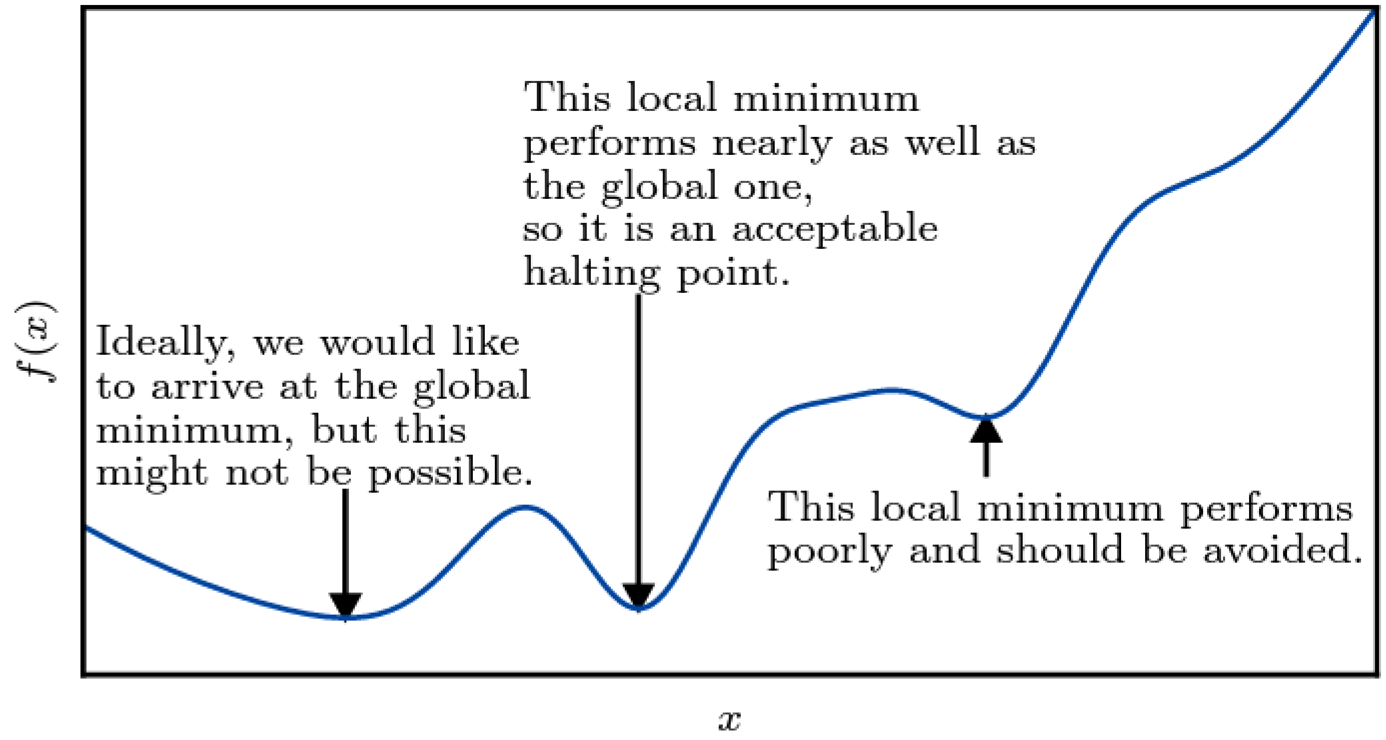

Goodfellow et al. Fig. 4.1

# Critical points




Goodfellow et al. Fig. 4.2

# Global vs. local minima




Goodfellow et al. Fig. 4.3

# Functions on $\mathbb{R}^m \to \mathbb{R}^1$

- Gradient vector of partial derivatives

$$\nabla f_x(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f_x(x) \\ \frac{\partial}{\partial x_2} f_x(x) \\ \vdots \\ \frac{\partial}{\partial x_m} f_x(x) \end{bmatrix}$$

- Moving in gradient direction will increase f(x) if we don't go too far…
- To move to an x with a smaller f(x)

$$x' = x - \epsilon \nabla f_x(x)$$

what we pick for ε will make a difference



IMPORTANT: f is loss fn to be optimized, e.g. MSE, not learner

# Putting this all together

note change in notation, objective is J()

- Want to learn: $f_{\Theta}(\mathrm{x}_\mathrm{i}) = y_i$
- To improve $f_{\Theta}(\mathrm{x}_\mathrm{i})$, define objective $J(\Theta)$
- Optimize $J(\Theta)$
  - Gradients defined for each sample
  - Average over data set (e.g. mini-batch) and update $\Theta$

# Putting this all together

- Sample *J(Θ)*, a loss function:

$$J(\theta) = E_{x,y\sim\hat{p}_{data}}[L(x,y,\theta)] = \frac{1}{m}\sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$$

where $L(x,y,\theta) = -\log P\,(y|x,\theta)$ remember $\mathrm{D}_{KL}$?

implies $J(\theta) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} -\log P\,(y|x,\theta)$

- which is like our effort to get a MLE by minimizing KL divergence:

$$H(P||Q) = E_{X\sim P}\left[-\log P_Q\,(X)\right] = E_{X\sim P}\left[-\log P_Q\,(X)\right]$$



Loss (aka cost) is a measurement of how far we are from the desired result.

# “There’s the rub”

Shakespeare’s Hamlet


- Computing J(Θ) is expensive

  One example not so bad…

  massive data sets…


- Sample expectation relies on having enough samples that the $1/m$ term estimates P(X).
- What if we only evaluated some of them…

# Stochastic gradient descent (SGD)

while not converged:

    pick minibatch of samples (x,y)

    compute gradient

    update estimate

minibatch is usually up to a 100 samples